Cisco Security Agent and Incident Handling

Greg Farnham

Cisco Security Agent and Incident Handling

GIAC (GCIH) Gold Certification

Author: Student Greg Farnham
Advisor: Don Weber

Accepted: September 30, 2009

Abstract

Deploying new security solutions requires consideration of the incident handling process. New solutions can impact the various phases of incident handling: Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned. An implementation of Cisco Security Agent (CSA) is analyzed and impacts to the incident handling process are identified. In addition, guidance is provided to configure CSA to support the incident handling process. An example malware incident is used to demonstrate the differences in incident handling with CSA and without CSA. Conficker is used as the malware sample. In order to understand the incident and required response, Conficker behaviors and how to test for them are documented.

1. Introduction

The goal of this project is to review how the deployment of a host intrusion prevention security solution impacts the incident handling process. The Cisco Security Agent (CSA) is the selected host intrusion protection systems (HIPS) (Cisco-3, 2009). This paper will review how a CSA deployment impacts the main phases of incident response: Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned (Skoudis, 2005). It will also provide guidance for configuring CSA to support the incident handling process. The Conficker/Downadup worm (F-Secure-1, 2009) will be used as a case study for incident response. An overview is provided for CSA, Incident Handling and Conficker.

1.1. CSA Overview

Cisco Security Agent is a host intrusion protection system that provides protection to servers and clients. CSA protects hosts by controlling operating system activities such as network access, file access, registry access, data access and memory access. By intercepting these activities, CSA can control them based on a centralized policy (Cisco-3, 2009).

The two main components of CSA are the Management Center and the host Agent (Cisco-3, 2009). The Management Center provides centralized policy management and centralized event logging. Administration is performed by connecting to the Management Center with a web browser. The Management Center delivers the centralized policies to each host Agent. The host Agent is installed on each protected endpoint. The Agent provides the enforcement at the endpoint as well as a graphical user interface (GUI) for user interaction (Sullivan, 2005). One of the CSA actions is to query the user to allow or deny an action. The GUI provides the Query response as well as some configuration options. CSA provides a wide array of policies for controlling host activity (Sullivan, 2005).

1.2. CSA Policies

CSA policies have a hierarchical structure. At the top of the hierarchy is the Group object. The group relates a list of hosts to a list of policies that will be applied

with them. A host can be in multiple groups and all policies from all groups will be applied to that host. The next level down is the Policy object. The Policy object's main purpose is to provide a list of Rule Modules to be included in that Policy. The Rule Module is the next level down in the hierarchy. The Rule Module provides a list of Rules for that Rule Module. The Rule object is the lowest level in the hierarchy. There are a number of different Rule types that can be defined. The Rule provides the details of the host protection. A simple example is a "Network Shield" type that blocks TCP SYN flood.

To summarize, a given host will get all the Policies from all the groups for which it is a member. It will get all the Rule Modules in all the Policies and all the Rules in all the Rule Modules. All the rules are combined to make up the overall policy applied to the host.

In addition to the policy hierarchy, CSA also tracks System State and User State. Rules can be used to set state values and other rules can be dependent on previously set state values. For example, one rule could be created to detect if a host is on the corporate network. This could be done by checking access to a Corporate Domain Controller. If a host is on the corporate network,, the location state can be set to "Corporate". Other rules can then be selectively applied based on the location state. This provides the capability to have more restrictive policies when a host is in an un-trusted network.

1.3. CSA Implementation Plan

Implementation of CSA requires a methodical approach to avoid impacting important services. CSA implementation typically involves several phases including Training, Planning, Testing, Pilot, Implementation and Continuation (Sullivan, 2005). The Training phase is required to bring administrators and operators up to speed on CSA. During the planning phase, the overall implementation plan is developed. Protected hosts are identified and base policies are defined. During the testing phase, CSA is installed and tested in a test environment that represents the production environment. The pilot phase is used to test CSA Policies on a small number of production hosts. A pilot allows the policies to be tested and refined before broad deployment. A pilot should include all categories of hosts that will be in the final deployment. The implementation phase is the

full deployment of CSA. Policies are first configured in Audit mode to identify any remaining false positives. Audit mode generates events for policy violations without enforcing blocking. After sufficient time in Audit mode, the hosts are removed from Audit mode which enables enforcement of policies. The final phase is Continuation where CSA is operated on a continual basis. Policies are modified as required by changes to the overall environment.

1.4. Conficker Overview

In order to analyze how a CSA deployment impacts the incident response process, a sample malware incident is used as a case study. The malware selected is Conficker also known as Downadup and Kido. There are many variants of Conficker this project looks only at Conficker.B referred to generically as Conficker for the rest of this paper. Conficker exploits a vulnerability (MS08-067) in the Windows Server Service. MS08-067 was announced by Microsoft in October, 2008 (Microsoft, 2008). Conficker first appeared in November 2008. Conficker.B (a.ka. Downadup.AD) first appeared in December 2008 (Bernadino, 2008). Infection estimates vary, but some place the number of infections in the 9-10 million range (F-Secure-2, 2009), (Porras, 2009), (Goodin, 2009). Conficker received a large amount of press coverage in late March of 2009 due to the fact that researchers determined that it was scheduled to update itself on April 1st, 2009. After much hype by the main stream media, there was little impact on April 1st. Conficker did begin receiving updates on April 7, 2009 (Ferguson, 2009), (Macalintal, 2009). Conficker began updating itself with a new version that supports a peer to peer (P2P) update mechanism. It also exhibited connections with the Waledec botnet and FakeAV spyware.

1.4.1. Conficker Basics

Conficker is a complex malware example. It has a comprehensive set of features and advanced capability. It has multiple attack vectors, built in defenses, virtual machine detection and advanced cryptography.

Conficker behavior described in this paper is based on several references from Security companies and other researches. While these summaries share much in common, there are a few subtle things that are mentioned in some and not others.

This paper covers behaviors relevant to how a Host Intrusion Prevention System (HIPS) will detect and block Conficker. There are many analyses which describe Conficker in great detail. There are minor variations in the analyses with some providing additional detail. For example, some state that Conficker copies itself to multiple locations. The ISS analysis clarifies that it only copies to the "Movie Maker" and other secondary locations if it fails to copy itself to %System% (Yason, 2009). For the purpose of this paper, some specific Conficker behaviors are discussed. For additional details on Conficker refer to the references. Conficker behaviors are categorized into Infection, Protection, Propagation and Update.

1.4.2. Infection

Conficker has multiple attack vectors including the exploit of the MS08-067 RPC vulnerability, autorun from USB Flash drive and guessing of weak administrative passwords. Once a host is compromised several steps are taken as part of the infection.

Conficker actions (Sophos, 2008)
malware copy placed in C:\windows\system32
malware attached to itself to system processes svchost.exe, explorer.exe and services.exe.
Creates a service: %System%\svchost.exe –k netsvcs
Adds registry key: HKLM\SYSTEM\CurrentControlSet\Services\<random filename>\Parameters\ServiceDll = "%System%\<random filename>". The display name is made up of 2 words from a list of over 20 common system words.
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run {Random characters} = rundll32.exe {System folder}\{Malware file name}.dll, {Parameters}"
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\

Windows NT\CurrentVersion\SvcHost
It modifies registry keys to ensure that its files stay hidden even if explorer is configured to display them. Note that dir / ah from cmd will show them. HKEY_CURRENT_USER\Software\Microsoft\Windows\ CurrentVersion\Explorer\Advanced Hidden = "2" HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\ CurrentVersion\Explorer\Advanced\Folder\Hidden\SHOWALL CheckedValue = "0"
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ Windows\CurrentVersion\Applets dl = "0" ds = "0"
It attempts to identify the internet IP address of the infected machine by connecting to checkip.dyndns.org, www.getmyip.org, www.whatismyip.org, www.whatsmyipaddress.com. This was verified by monitoring firewall logs.

Table 1: Highlights of Conficker Infection Actions

1.4.3. Protection

Conficker takes a number of steps to protect itself from detection and removal. It terminates processes associated with several security related tools. It will terminate process associated with matching strings such as tcpview and Wireshark. A test was conducted running Wireshark and infecting the host. As expected, the Wireshark application was terminated. Included in the strings for process names to terminate are confick, downad and kido. After the security industry named the malware, it included those names in order to kill cleanup tools.

Conficker will hook several DNS related APIs. It will intercept DNS queries to several security web sites (Sophos, 2009). This has the affect of blocking access to AV updates and other security tools. No CSA process names are included in the list.

Another step Conficker takes for protection is to disable security and update services. The following services are disabled according to most Conficker analyses (Ferrer, 2009):

- wscsvc - Security Center
- wuauserv - Automatic updates
- BITS - Background Intelligent Transfer Service
- WinDefend - Windows Defender
- ERSvc - Error Reporting Service
- WerSvc - Windows Error Reporting Service

Conficker will also disable the Windows Security Center notifications. This is checked via the Security Center control panel and/or the registry key:

```
HKLM\Software\Microsoft\Windows\CurrentVersion\explorer\ShellServiceObj
ects\{FD6905CE-952F-41F1-9A6F-135D9C6622CC}
```

It resets and deletes system restore points. This step makes it impossible to roll back the configuration with a system restore.

Conficker sets a registry key that will ensure its files are hidden. The following key is set:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\A
dvanced\Folder\Hidden\SHOWALL CheckedValue = dword:00000000
```

Conficker defines a service so that it can be restarted on reboot. The following registry key is created:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SvcHost, netsvcs = %Previous data% and %Random%
```

Finally, Conficker will inject itself in to svchost.exe to hook NetpwPathCanonicalize. This closes the MS08-067 vulnerability and prevents it from being re-infected. However, since it is an in memory patch if the host is cleaned and rebooted, it would still be vulnerable to MS08-067 and could be re-infected. This has the affect of protecting a Conficker infected system from other malware. Also, since the system is not patched Conficker has the opportunity to re-infect it if the malware is removed and the patch is not applied.

Greg Farnham

1.4.4. Propagation

Conficker has several attack vectors for propagation. Primarily, it will spread via the network to hosts vulnerable to MS08-067, via weak passwords and via removable media. It will infect any USB Flash drives inserted into an infected system.

The exploit of the MS08-067 vulnerability occurs in three steps. First, it creates an http listener which can deliver the full Conficker program. The file is delivered with an extension of bmp, gif, png or jpg

Second, it bypasses the firewall, by changing the following registry key.

```
HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\Firewall
Policy\StandardProfile\GloballyOpenPorts\List, [PortNumber]:TCP =
"[PortNumber]:TCP:*Enabled:[random]"
```

Third, it exploits the MS08-067 vulnerability over the network and pulls the Conficker program from the http listener.

It will attempt to access available network share (IP\ADMIN$\system32) using weak administrator password (Ferrer, 2009). If successful, it will copy itself to that system and create a scheduled job to run it with rundll32.exe.

It increases the max tcp connections by modifying the following registry key:

```
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpNumConnectio
ns = 0x00FFFFFE
```

It uses PnP functionality of the upstream Internet Gateway to configure a port forward to allow it to be accessed from the internet.

It will keep a count of the number of times it was downloaded in the following registry key.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
Windows\CurrentVersion\Applets
gip = "dword:%count%"
```

Once a system has been infected and Conficker has its protections in place, it then can attempt to update itself.

1.4.5. Update

Conficker attempts to connect to 50 domains from a unique daily list of 250 based on a predicable algorithm. Conficker.B contacts 250 domains once very 2 hours (Fitzgibbon 2009). Note the Conficker sample used in the lab tests attempts to phone home every 3 hours. An example URL is shown on the following line:

```
http://example.com/search?q=0
```

Later versions of Conficker use peer to peer networking to deploy updates to infected hosts.

2. Incident Handling and CSA Implementation

The incident handling process is made up of the Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned phases. Each of these phases can be impacted by the implementation of a new security solution. Implementation of CSA will require changes to the incident handling process. The incident handling process will also need to be taken into consideration when planning a CSA implementation. The first phase in the incident handling process is preparation.

2.1. Preparation

The preparation phase includes defining the personnel, tools and procedures that are used in all the other phases. The CSA implementation as a whole is part of preparation. As a Host Intrusion Prevention (HIPS) solution, CSA will detect and prevent attacks from infecting protected hosts. CSA day zero protection blocks most worms with default policies (Cisco-2, 2008). By blocking attacks as part of preparation, a potentially major incident will be reduced to simply an event or a minor incident. For example, if an infected USB drive is inserted into a CSA protected host, the host will not be infected. An incident response will still be required to investigate the infected USB drive. The infected USB drive will need to be cleaned and the response should include checking any other hosts it was used on. Part of preparation is defining what happens in other phases. CSA preparation impacts on specific phases are included in the section for that phase. The next phase in the incident handling process is identification.

2.2. Identification

The Identification phase is used to identify incidents. Events are analyzed to determine if an incident needs to be declared. CSA will be a helpful asset in this phase. CSA will contribute to the identification phase by adding additional event reporting and alerting. CSA will generate Alerts and Event Logs based on the defined policies.

2.2.1. CSA Alerts and Logs

CSA generates Events based on rule configuration (Cisco-3, 2009). These events can be viewed interactively through the CSA Management Center (MC) browser interface. Alerts are given a severity of Information, Notice, Warning, Error, Alert, Critical or Emergency. The CSA Implementation plan will need to include configuration of Alerts to support the Incident Response process. In addition, CSA can be configured to deliver alerts for events based on severity. CSA can send alerts using five different methods, Email, SNMP, Log File, Custom Program or a Named Pipe. For the example deployment CSA will be configured to send alerts to the security team's alert email list (security@example.com). The default event set "Significant events of all types" will be used. This event set will generate alerts for severity warning and above for all events on all hosts. The configuration is shown in Figure 1.

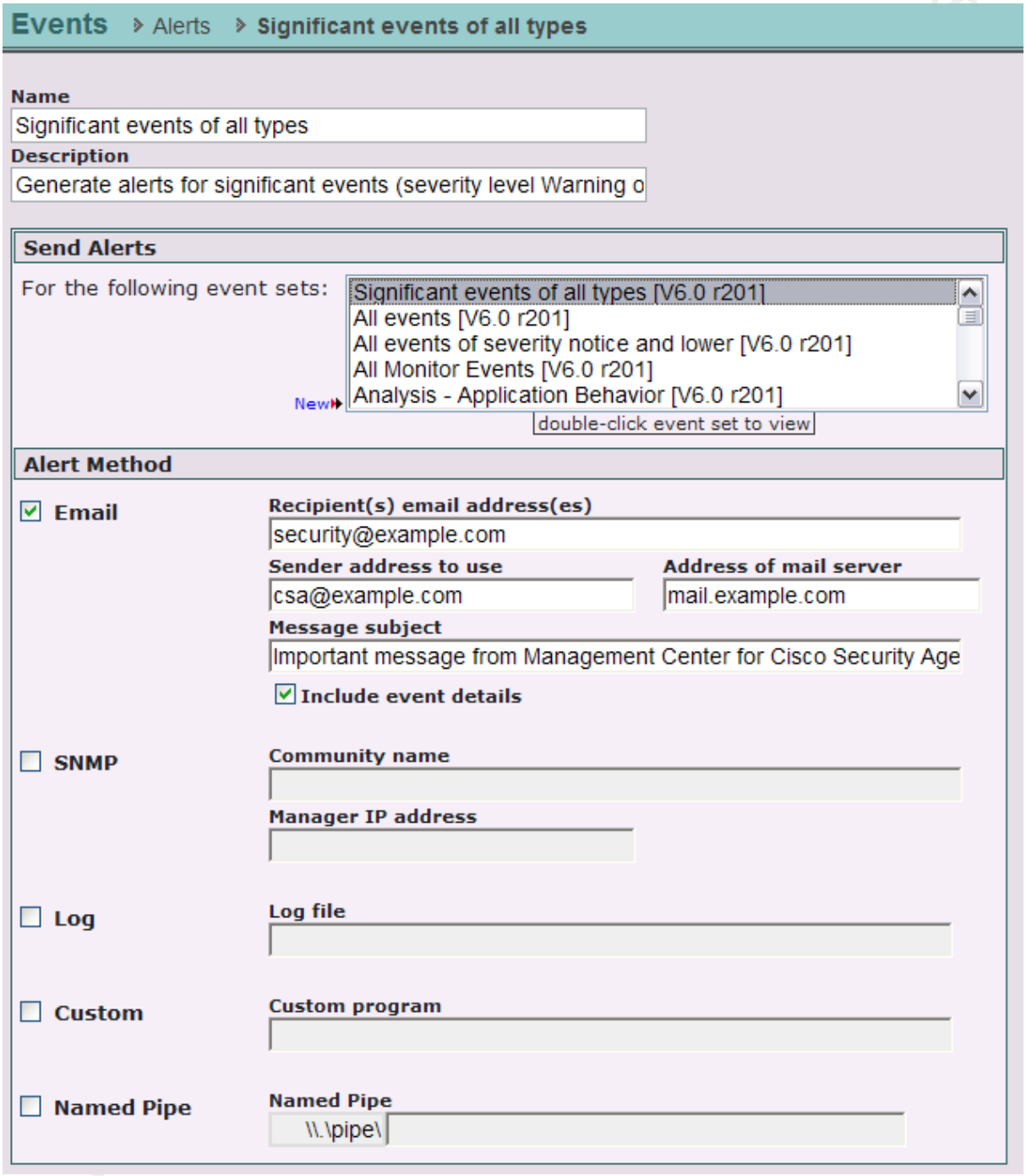

Figure 1: CSA alert configuration

The security email list is monitored by security staff and is used for all security alerts. The incident response plan will be modified to include CSA alerts and reviewing the CSA event log as inputs to the identification process. The CSA event logs will be used to interactively investigate possible incidents.

Greg Farnham

Another feature of CSA is the ability to collect Windows event logs from CSA protected hosts. This is useful for investigating potential incidents (Mandia, 2001). The CSA implementation plan will be modified to include implementation of a Windows event logs policy. This will also require endpoints to be configured to Audit events in the Local Security Settings. CSA includes a sample Rule Module for Windows event logging that is not part of the default configuration. A custom Policy was created using a custom configured version of this Rule Module. CSA has 22 pre-defined Rules for capturing various Windows (a.k.a. NT) event logs. Rules were selectively enabled for the example deployment. The event log rule configuration is shown in Figure 2.

ID	Type	Events (Last 24hr)	Status	Action	Log	Description
800	NT Event log		Enabled	-	-	Add SID History
801	NT Event log		Enabled	-	-	Encrypted Data Recovery Policy Change
802	NT Event log		Enabled	-	-	Global Group related
803	NT Event log	6 (0)	Disabled	-	-	User Account/Rights related
804	NT Event log	1938 (496)	Enabled	-	-	Successful Network Logon
805	NT Event log		Enabled	-	-	Audit Log change
806	NT Event log		Enabled	-	-	QoS Policy Change
807	NT Event log	3 (0)	Enabled	-	-	Computer Account related
808	NT Event log	222 (0)	Enabled	-	-	Successful Logon
809	NT Event log	11 (0)	Disabled	-	-	Kerberos and Ticket related
810	NT Event log	2 (0)	Disabled	-	-	Windows Startup
811	NT Event log		Enabled	-	-	VPN (IKE and IPSec) related
812	NT Event log	3 (0)	Disabled	-	-	Windows Shutdown
813	NT Event log	14 (0)	Enabled	-	-	Logon security package loaded
814	NT Event log		Enabled	-	-	Group Type Changed
815	NT Event log		Enabled	-	-	User Account Locked Out
816	NT Event log		Enabled	-	-	Universal Group related
817	NT Event log	94 (0)	Disabled	-	-	Privileged access related
818	NT Event log	1659 (400)	Enabled	-	-	User Logoff
819	NT Event log		Enabled	-	-	Trusted Domain related
820	NT Event log		Enabled	-	-	Local Group related
821	NT Event log	154 (0)	Enabled	-	-	Logon Failure

Figure 2: CSA Event log rule configuration

For event monitoring, all successful and failed logon events can be logged on the hosts. Logon failure event logs can be vary useful in identifying an attack. Conficker

and other malware that attempts password guessing will result in a large number of logon failures. This will also result in account lockouts if that is configured in the operating system. A sample event from an Account lock out is shown in Figure 3.

930	8/23/2009 11:53:01 PM	dc.lab.local	Notice	-

The 'Security' service logged event code 539 into the security event log: Logon Failure:
Reason: Account locked out
User Name: user1
Domain: C2
Logon Type: 3
Logon Process: NtLmSsp
Authentication Package: NTLM
Workstation Name: C2
Caller User Name: -
Caller Domain: -
Caller Logon ID: -
Caller Process ID: -
Transited Services: -
Source Network Address: 192.168.0.102
Source Port: 0

Figure 3: Example event log for Account lock out

In addition to logging individual events for failed login CSA can perform event correlation across multiple hosts. If an infected host is causing failed logons to multiple hosts CSA will generate a Warning level alert. The default configuration is to generate a warning if two or more systems report failed logons within 30 minutes.

For the identification phase, the impact to the CSA implementation is to include configuration Windows Event logging policies, configuration of alerting via email and configuration of Event Correlation across all hosts. The impact to the Incident Response plan is to document how various events and alerts are handled. After identification, the next phase is containment.

2.3. Containment

During containment, efforts can be taken to keep infected hosts from spreading the infection. CSA can help in this phase by creating a special lockdown policy for containing infected hosts.

2.3.1. CSA Containment Policy

For containment a CSA Network Lockdown policy can be created. This policy will block all TCP and UDP traffic to and from the host. CSA management traffic is exempted, so CSA policy changes can still be made to the host. The predefined "Security

– Network Lockdown" Module was used as a starting point. For the "Network access control" rule, the action was changed from deny to "priority deny". This was done to ensure that any "priority allow" rules do not preempt the deny action.

To better understand this configuration decision, it is useful to explain how CSA combines rules from multiple Policies. CSA rules have six actions for allowing or denying activities. These rule actions are shown in Figure 4.

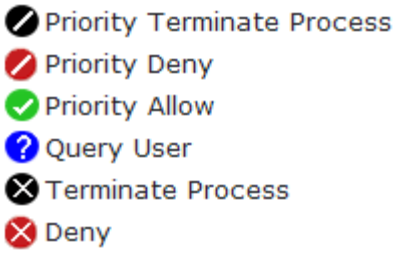

Figure 4: Rule Actions

When CSA combines Rules from multiple Policies it will sort them based on the action. They are sorted in the same order shown in Figure 4. All Priority Terminate Process rules come first followed by all Priority Deny rules followed by the remaining actions in Figure 4. Since Priority Allow comes before Deny, it is possible that another rule with Priority Allow would preempt a Deny. Therefore, by using a Priority Deny for lockdown, it ensures the activity will be denied. Note also that if Query User is used it can have one of three options for the default action if the user does not respond within the time limit (default of 5 minutes). The default query options are Terminate Process, Deny and Allow.

In addition to the main actions, there are five other actions that can be used which do not explicitly allow or deny traffic. The other actions are shown in Figure 5.

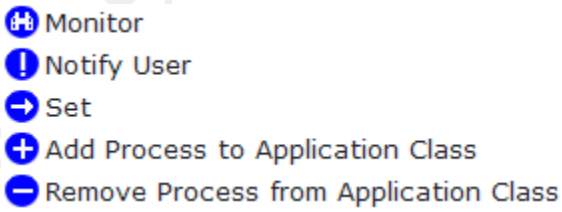

Figure 5: Other Rule Actions

One of the resulting Rules for the Network Lockdown Policy is shown in Figure 6.

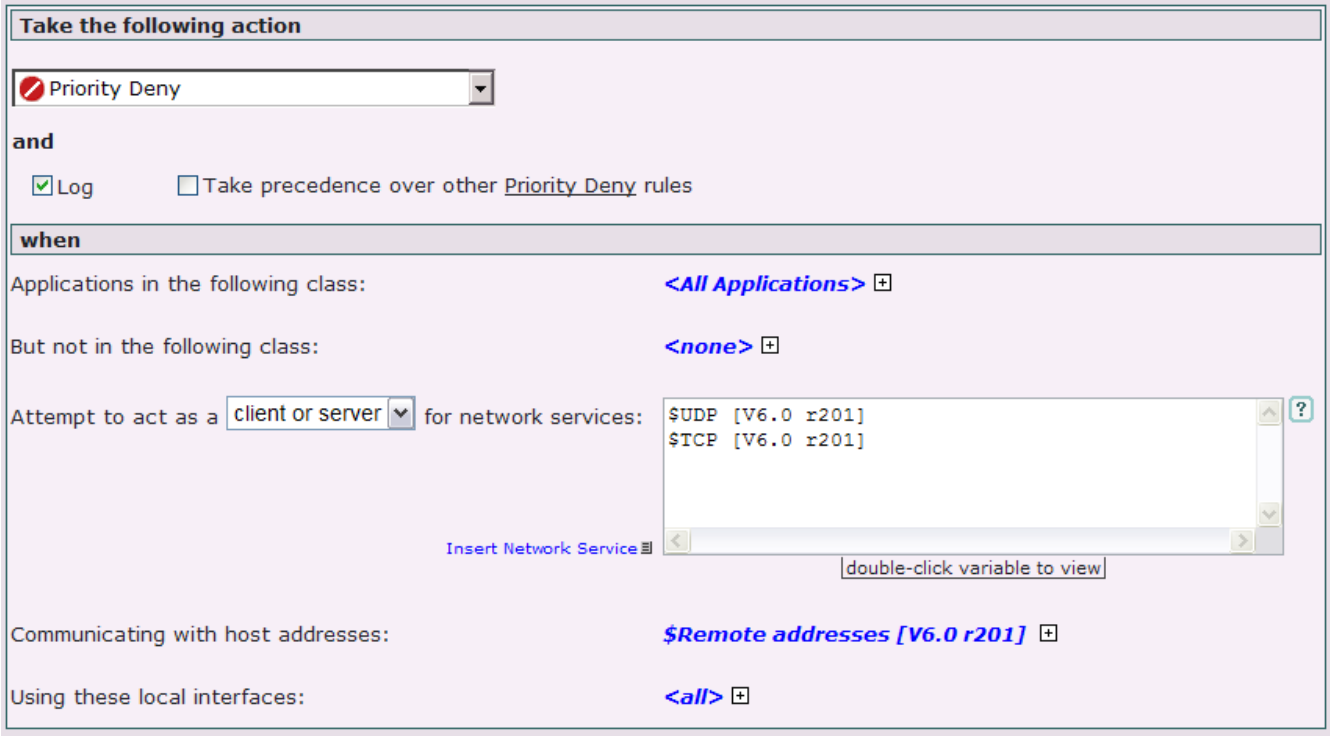

Figure 6: Network Lockdown Rule for TCP and UDP traffic

This rule will deny all inbound and outbound TCP and UDP traffic. However, it does not terminate existing connections.

This policy does not block any activity that occurs locally on the host. In other cases, an incident specific lockdown Policy could be deployed. For example, a policy could be defined to block specific activities such as registry changes or keyboard interception.

In normal operations, no hosts would have the Network Lockdown Policy assigned. There is a Network Lockdown Group created that assigns the Network Lockdown Policy to any hosts that are a member. If a host is suspected of being infected it would be added to this group and the policy changes would be deployed. This would stop the host from doing any network based attacks on the rest of the network. It would also stop it from being able to do any legitimate network activity such as web browsing. A technician would need to physically go to the host to remediate it.

Greg Farnham

The Network Lockdown Policy can only be applied to a host that has the CSA agent installed. If a host has CSA deployed with the default protection, it likely would not be infected anyway. To use the Network Lockdown on other hosts a method for remotely installing CSA would be required. Many companies have a software deployment system in place that could be used. Alternatively, for hosts where full CSA protection is not desired, CSA could be deployed in Audit mode or with some minimal protection policy. Then, the Network Lockdown policy could quickly be implemented if necessary for an incident.

The impact to the CSA implementation is to create a Network Lockdown policy that could be used for containment. The impact to the Incident Response plan is to document the process for using the Network Lockdown policy to contain infected hosts. After containment, the next phase is eradication.

2.4. Eradication

During the eradication phase, infected or compromised hosts have the malware removed. Eradication would typically be done using an Anti-Virus solution. While the latest version of CSA does offer Clam Signature Based AV, it is not included in the scope of this project. Infected hosts could be cleaned using malware removal tools or standard Anti-Virus (AV) solutions. Most organizations have a standard AV solution deployed. Depending on the severity of the infection, the host may require a clean Operating System installation.

For the Eradication phase, no major changes will be required to the CSA Implementation plan or the Incident Response plan. The next phase in the incident handling process is recovery.

2.5. Recovery

During recovery, all systems are returned to a normal operating state (Van Wyk, 2001). Once a host is cleaned or rebuilt, it can be returned to service. If the CSA Network Lockdown policy was applied, it would need to be removed from the host for it to be returned to service.

For the recovery phase, there are no major changes required to the CSA implementation plan. For the Incident Handling plan, the removal of the CSA lockdown policy will need to be included in the recovery phase. The last phase in the incident handling process is lessons learned.

2.6. Lessons Learned

CSA will be useful for forensics analysis of the incident. The CSA Event Log includes any CSA related events and also Windows Event Logs. They can be reviewed to help build a timeline of the incident. Additionally, some events include details such as a stack recovery dump. Figure 7 shows a disassembly from a Conficker infection.

Disassembly

Address	Code	Instruction
7c81f042	e9b442ffff	jmp 0x7c8132fb
7c81f047	90	nop
7c81f048	90	nop
7c81f049	90	nop
7c81f04a	90	nop
7c81f04b	90	nop
7c81f04c	6a10	push dword(0x10)
7c81f04e	6888f0817c	push dword(0x7c81f088)
7c81f053	e87334feff	call 0x7c8024cb
7c81f058	8b4d0c	mov ecx,dword[ss:ebp+0xc]
7c81f05b	85c9	test ecx,ecx
7c81f05d	7439	je 0x7c81f098
7c81f05f	8b4508	mov eax,dword[ss:ebp+0x8]
7c81f062	85c0	test eax,eax
7c81f064	743c	je 0x7c81f0a2
7c81f066	8bd0	mov edx,eax
7c81f068	8d4c08ff	lea ecx,[eax+ecx*1+0xffffffff]
7c81f06c	8365fc00	and dword[ss:ebp+0xfffffffc],0x0
7c81f070**	8a00	mov al,byte[eax]
7c81f072	8845e7	mov byte[ss:ebp+0xffffffe7],al
7c81f075	84c0	test al,al
7c81f077	741b	je 0x7c81f094
7c81f079	3bd1	cmp edx,ecx
7c81f07b	7417	je 0x7c81f094
7c81f07d	42	inc edx
7c81f07e	8955e0	mov dword[ss:ebp+0xffffffe0],edx
7c81f081	8a02	mov al,byte[edx]
7c81f083	ebed	jmp 0x7c81f072
7c81f085	90	nop
7c81f086	90	nop
7c81f087	90	nop
7c81f088	ff	undefined: bad opcode
7c81f089	ff	undefined: bad opcode
7c81f08a	ff	undefined: bad opcode
7c81f08b	ffc7	inc edi

Figure 7: Disassembly of Stack Recovery

For a Lessons Learned activity, CSA Policies should be reviewed. If a host had CSA protection and was infected, more restrictive policies may be required. If a host did

not have CSA protection, consideration should be given to adding CSA protection to that host.

For the Lessons Learned phase, CSA will be useful for forensic analysis and a CSA policy review should be included. A lab environment was created to demonstrate CSA functionality using Conficker as a case study.

3. Conficker Incident lab

In order to demonstrate the concepts in this project, a malware incident lab was used. The goal of the lab was to demonstrate the difference in incident response for an environment without CSA versus an environment with CSA. In this lab, a network was created with a small number of hosts. Most hosts are directly connected to a hub. A network diagram is shown in Figure 8. The version of CSA (6.0) and the policies used were released prior to MS08-067 being published and prior to the appearance of Conficker. The following scenarios are used in the lab tests.

Scenario 1 – Conficker attack without CSA

Scenario 2a – Conficker attack with CSA (on all hosts)

Scenario 2b – Conficker attack with CSA (on all hosts except one)

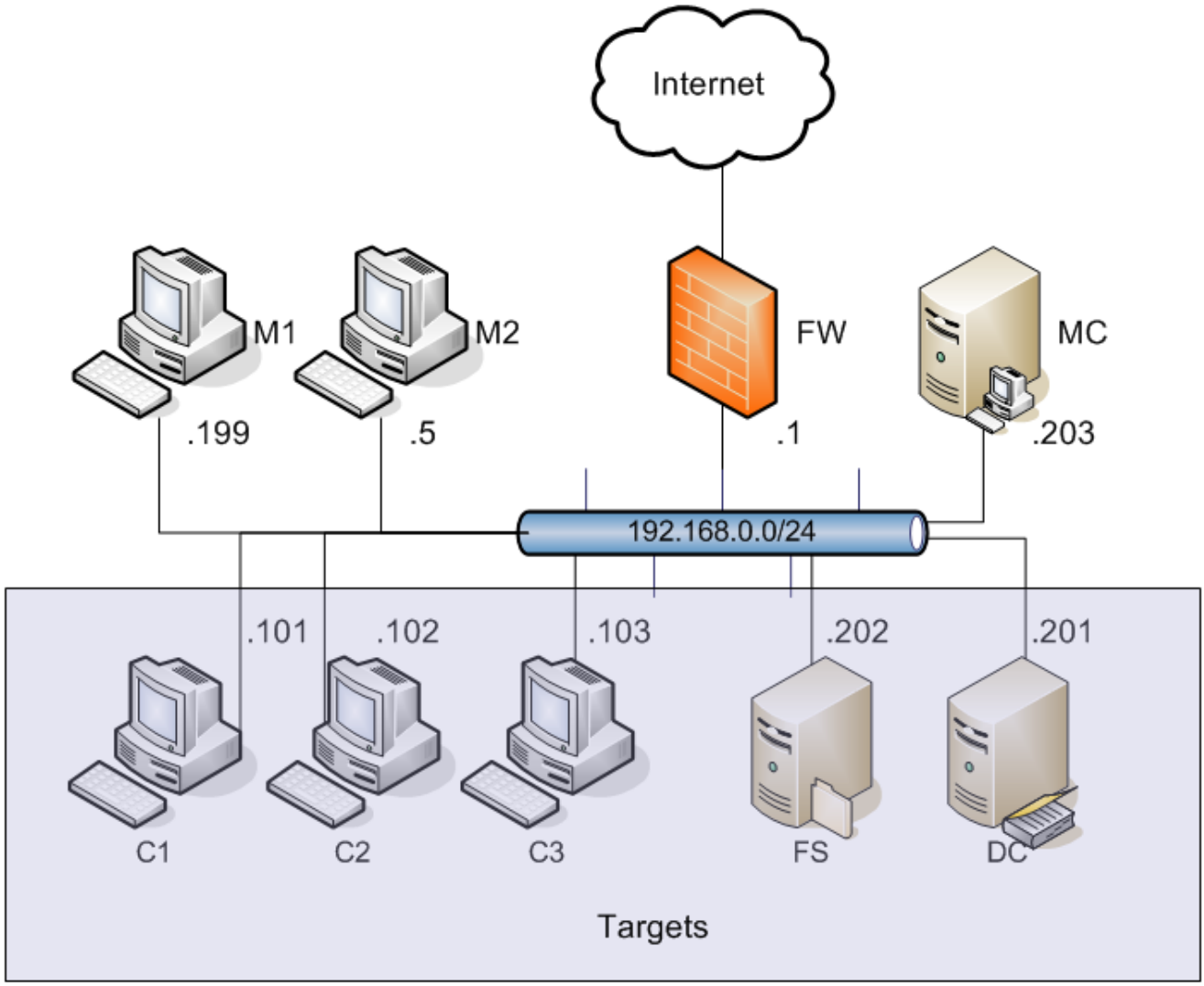

Figure 8: Lab network diagram

The target environment consists of a domain controller, a file server and three clients. The servers are running Windows 2003 SP2. The clients are running Windows XP SP2 or SP3. Physical machines were used for all target hosts since Conficker has virtual machine detection. In addition to the target hosts, there is the CSA Management Center (MC) and two hosts for monitoring. One monitoring host (M1) is used to collect syslog data from the Firewall and the other (M2) is used to capture network traffic.

3.1. Lab Configuration

The environment was setup with hosts that would be vulnerable to Conficker attack vectors when CSA is not installed. Hosts C1 and C2 are setup vulnerable to the

MS08-067 RPC vulnerability. C1, C2, FS and DC are all domain members. C1 and C2 mount files shares data and pub from FS. In some scenarios, hosts are configured with a weak administrator password. All device clocks are synchronized.

All scenarios will start with a USB Flash drive infected with Conficker as an autorun. The flash drive is inserted into C1 and the reaction is monitored. A Cisco firewall is used to control traffic attempting to leave the lab network.

Hostname	IP	OS	Description
DC	.201	W2K3SP2	Domain Controller
FS	.202	W2K3SP2	File Server
C1	.101	WXPSP2	Client, patient 0
C2	.102	WXPSP2	Client, vulnerable to MS08-067
C3	.103	WXPSP3	Client, non vulnerable to MS08-067
MC	.203	W2KSP2	CSA Management Center
M1	.199	WXPSP3	Monitor 1, Syslog
M2	.5	Ubunto 8.0.4	Monitor 2, Network Capture, tcpdump
FW	.1	PIX	Firewall

Table 2: Lab Configuration

The host firewall is turned off on the target hosts. The target hosts are configured to use the domain controller for DNS queries. The firewall is configured to allow outbound DNS queries, but block any malicious traffic to Windows SMB ports.

The Conficker sample used in the lab was checked using virustotal.com. The results are shown in Appendix B. The security vendors all have different ways of naming malware as shown in Appendix B. The behavior of the sample used in the lab testing is consistent with Conficker.B (Cisco-1, 2008).

Several different tests were performed for the various lab scenarios. The results for a given scenario may come from separate test instances. Public IP addresses are typically sanitized or blacked out. For sanitizing the first octet is changed to 5 which is a

class A network currently reserved by IANA. The second octet is changed to some other arbitrary number.

3.1.1. Logging and Monitoring

In order to collect data regarding Conficker worm propagation several sources of logging and monitoring were used. All outbound firewall rules are logged to the Syslog server. The Domain Controller also provides DNS services to target hosts. All DNS requests are logged locally on the Domain Controller. The M2 monitor host runs tcpdump to do a full network capture of traffic on the hub. When CSA is used, CSA events are logged to the Management Center. Target clients are configured to send important Windows Events to the CSA MC. A batch file was used to export Conficker related registry keys to a text file before and after infection. A second batch file was used to log service status and scheduled jobs to a text file at a regular interval (1-30 seconds depending on the test). This was done via a modified version of a WMIC script from a SANS Technology Institute project (Proffitt, 2009). A third batch file was used to scan for infected hosts using nmap. It uses the nmap script engine for smb-check-vulns and smb-os-discovery (Fyodor, 2009). This third batch file was later dropped from the testing. In some cases the hosts stopped responding to the nmap scans. These scripts are shown in Appendix A. Hosts were also checked manually for Conficker infection using the information in section 1.5.

3.2. Conficker attack without CSA

3.2.1. Overview

Scenario 1 represents a Conficker attack without CSA. It was carried out with the lab configuration described above. The goal was to demonstrate worm propagation in an environment without HIPS and identify the required incident response. In this scenario an employee brings an infected USB drive into the office and inserts it in their workstation. The employee's workstation and other hosts are unprotected from Conficker. This section includes data from several different tests of this scenario. The different incidents had some slight variations in configuration. The target hosts did not have CSA installed.

3.2.2. Attack Vector

The initial attack begins when the user inserts an infected USB flash drive into host C1. The USB flash drive contains an autorun.inf which directs the operating system to run the Conficker executable using rundll32.exe. Once infected, C1 will attempt to infect other hosts using multiple attack vectors.

3.2.3. Attack Results

In an environment without protection, the Conficker worm quickly spreads to additional hosts. Given the variety of actions taken by Conficker, several methods were used to check and verify the behavior. The attack results showed infections across multiple hosts. Other infection symptoms were checked as well based on section 1.5. To analyze the results, data from the infected machines is reviewed. The methods used to monitor Conficker behavior are also included.

Services

One simple and reliable way to check for Conficker infection is to check the services running on a host. For the Windows XP SP2 client used, Conficker disabled the Automatic Update and Background Intelligent Transfer (BITS) services. The Error Reporting Service and the Security Center were not disabled in any of the lab tests. The other services mentioned in Conficker Analyses (Ferrer, 2009) were not present on the lab hosts. A WMI script was used to output the status of the Automatic Update and BITS Services. The script output is time stamped and can be used to identify the time of infection.

HTTP listener and Host Firewall

Another way to check for Conficker infection is to look at the Windows Firewall exceptions. Conficker creates an http listener and creates an exception in the Windows Firewall for it. Conficker injects itself in svchost.exe and creates an http listener. The listener is used to serve files to additional victims infected via the MS08-067 vulnerability. Figure 9 shows a Conficker listener on port 4238 with a PID of 856 based on the netstat command. It also shows that this PID is for svchost.exe based on the tasklist command.

```
C:\>netstat -nao

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       792
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       728
  TCP    0.0.0.0:4238           0.0.0.0:0              LISTENING       856
  TCP    127.0.0.1:1029         0.0.0.0:0              LISTENING       1876
  TCP    192.168.0.101:139      0.0.0.0:0              LISTENING       4
  TCP    192.168.0.101:3389     192.168.0.251:33593    ESTABLISHED     728
  UDP    0.0.0.0:445            *:*                                    4
  UDP    0.0.0.0:500            *:*                                    576
  UDP    0.0.0.0:1026           *:*                                    912
  UDP    0.0.0.0:4500           *:*                                    576
  UDP    127.0.0.1:123          *:*                                    856
  UDP    127.0.0.1:1034         *:*                                    856
  UDP    127.0.0.1:1900         *:*                                    960
  UDP    192.168.0.101:123      *:*                                    856
  UDP    192.168.0.101:137      *:*                                    4
  UDP    192.168.0.101:138      *:*                                    4
  UDP    192.168.0.101:1900     *:*                                    960

C:\>tasklist

Image Name                     PID Session Name     Session#    Mem Usage
========================= ====== ================ ======== ============
System Idle Process              0                        0         16 K
System                           4                        0        212 K
smss.exe                       420                        0        388 K
csrss.exe                      496                        0      3,112 K
winlogon.exe                   520                        0      3,324 K
services.exe                   564                        0      4,968 K
lsass.exe                      576                        0      1,948 K
svchost.exe                    728                        0      4,680 K
svchost.exe                    792                        0      3,748 K
svchost.exe                    856                        0     19,728 K
svchost.exe                    912                        0      3,592 K
svchost.exe                    960                        0      4,192 K
```

Figure 9: Conficker listener on port 4238 injected into svchost.exe

To allow access to the http listener, Conficker creates an exception rule in the Windows Firewall. An example exception rule is shown in Figure 10.

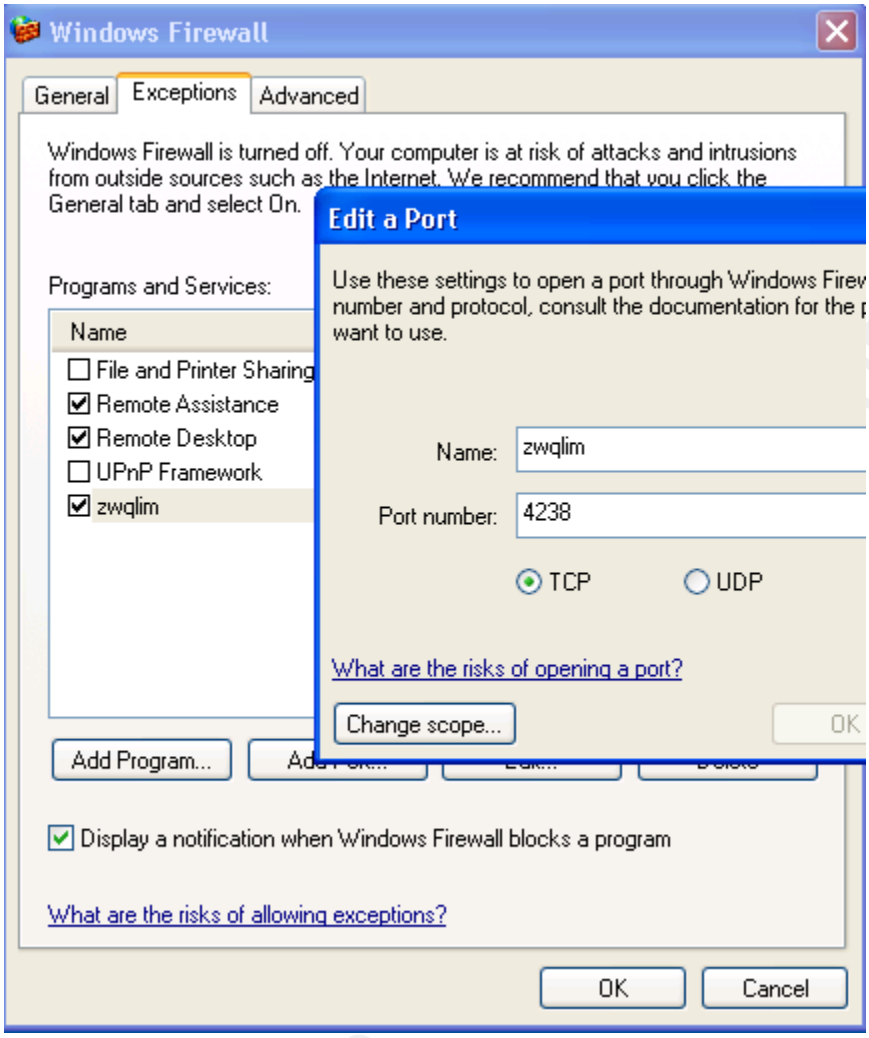

Figure 10: Conficker Firewall exception

Another method for verifying Conficker infections is by looking at actual captured traffic. For the lab tests network traffic was captured using tcpdump and later analyzed using Wireshark and NetWitness Investigator. Figure 11 shows an example capture of a recently infected host (C2) connecting back to the http listener on patient zero (C1).

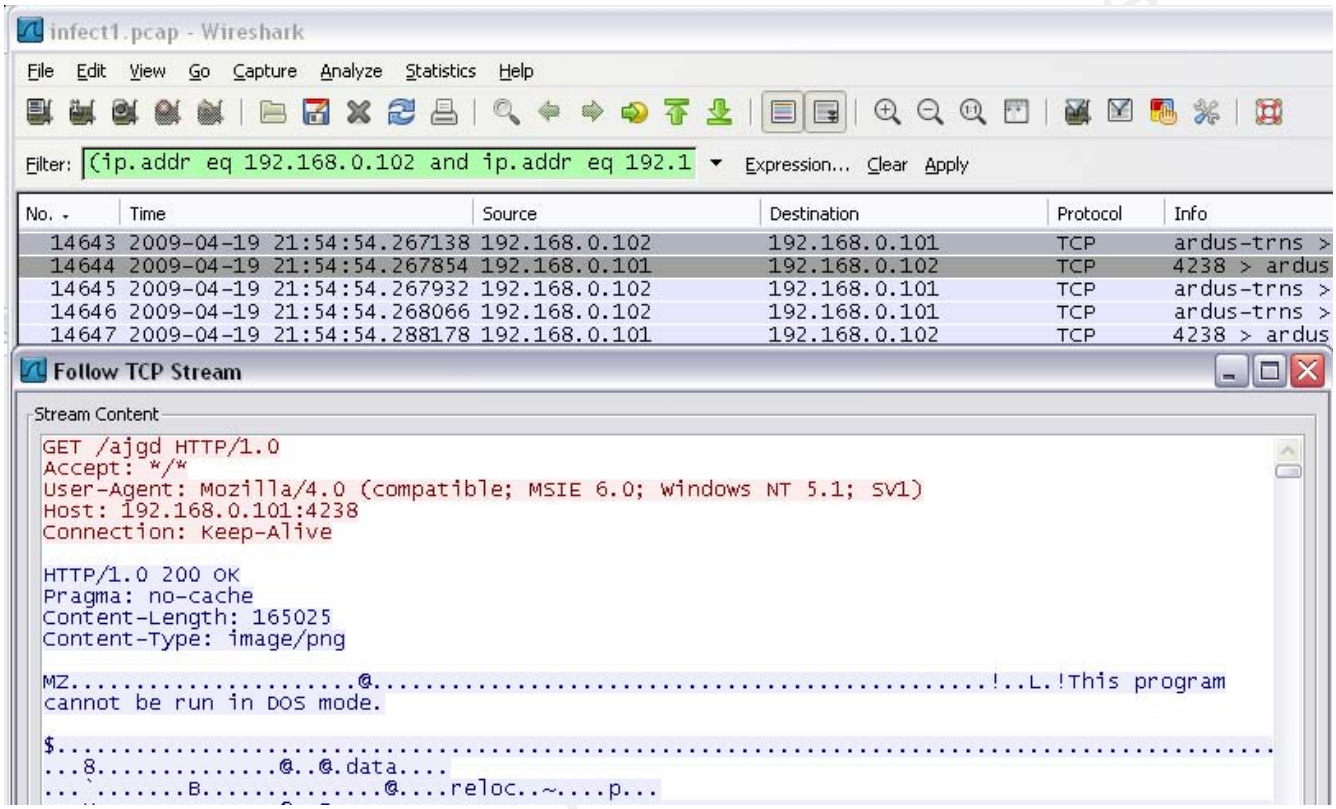

Figure 11: Infected host (.102) connecting to Conficker http listener (.101).

The tools available for analyzing network traffic captures are invaluable for analyzing Conficker behavior. The full packet capture can be use to show high level view of events, query for event types or view specific packets. Additional examples are shown for time history, application layer analysis and SMB decoding.

Time History

An example of a high level view is shown in Figure 12. This time history diagram shows SMB login attempts that occur approximately every 40 minutes at peaks A, C and D. Peak B shows attempts to phone home to multiple domain names. These domain names are unique for each day and generated by a specific algorithm.

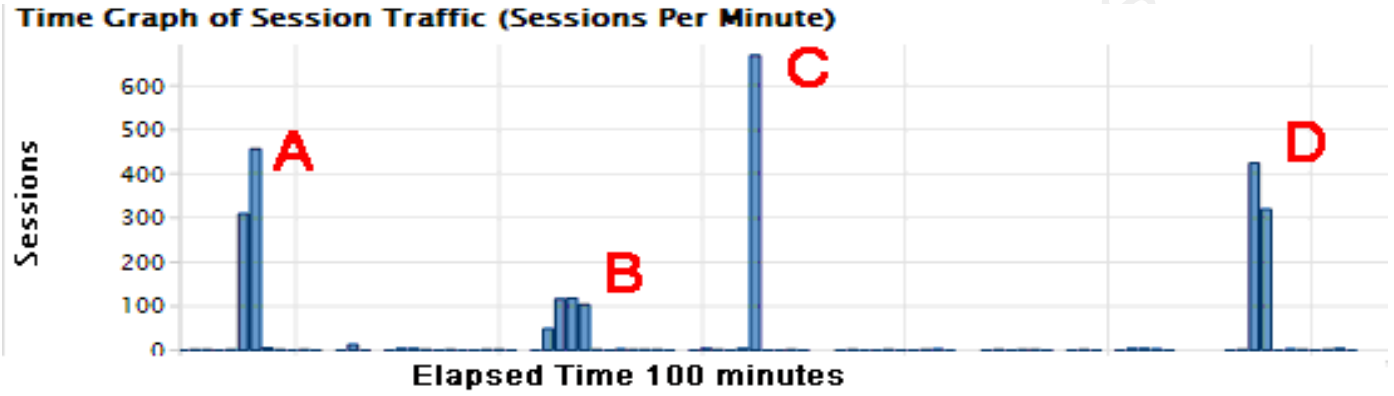

Figure 12: Time History of Network Traffic

Phone Home Query String

With a full network capture the data can be queried to display data that matches specific criterion. A useful application for analyzing network captures is NetWitness Investigator. This application was used to analyze the Conficker data. NetWitness analyzes multiple layers of the data including the application layer. One of the ways this can be used to display Conficker infected hosts is to show all sessions that used the query string "q=0" when requesting a web page. The source ip addresses can be checked to see which hosts are infected. That string is included when Conficker attempts to phone home. An example of this is shown in Figure 13.

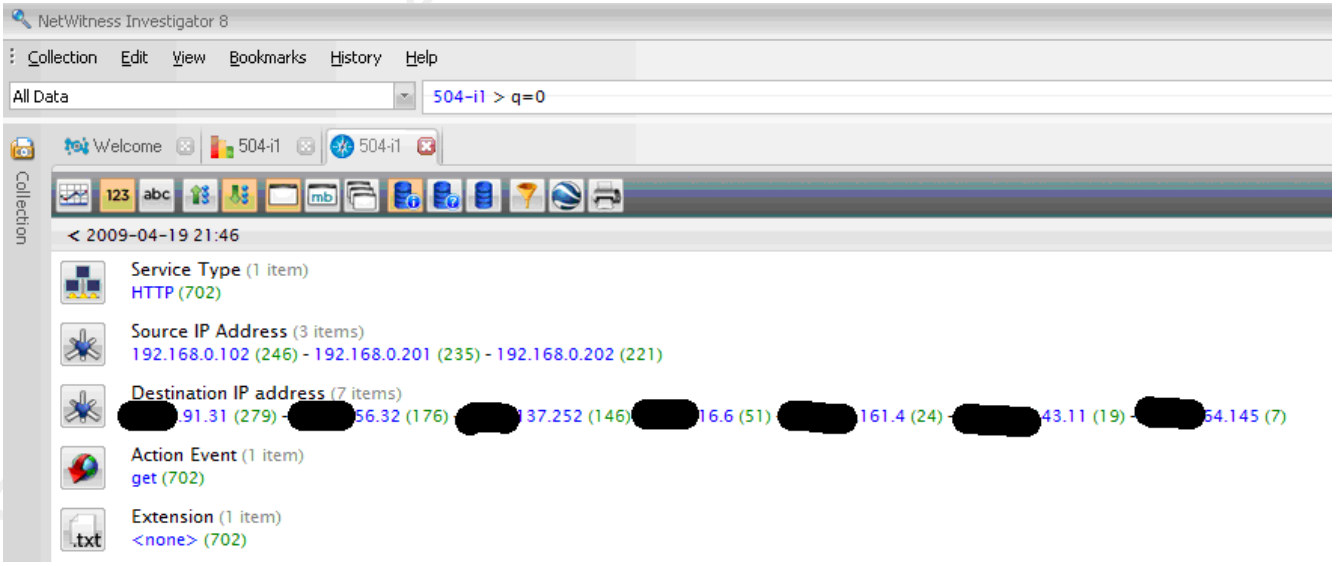

Figure 13: Traffic filtered to show only Conficker query string q=0

Note that in Figure 13 while Conficker is attempting to connect to many unique domains, only 7 unique ip addresses were returned. This is likely due to the Conficker Working Group pre-registering all of the Conficker phone home domain names (Conficker Working Group, 2009). A spot check of a few domain names showed them registered to the "Conficker Cabal", the informal name for the Conficker Working Group.

SMB Protocol Details

The full network capture can also be used to drill into the details of network events. The Wireshark application was the primary tool for detailed network traffic review. Figure 14 shows decoded SMB traffic of bvwwtbp.dll being copied from host C1 (.101) to host FS (.202). This is part of the exploit of a weak admin password.

192.168.0.101	SMB	Tree Connect AndX Response
192.168.0.202	SMB	Trans2 Request, QUERY_PATH_INFO, Query File Basic Info, Path: \System32
192.168.0.101	SMB	Trans2 Response, QUERY_PATH_INFO
192.168.0.202	SMB	Trans2 Request, FIND_FIRST2, Pattern: \System32\bvwwtbp.dll
192.168.0.101	SMB	Trans2 Response, FIND_FIRST2, Error: STATUS_NO_SUCH_FILE
192.168.0.202	SMB	NT Create AndX Request, Path: \System32\bvwwtbp.fid
192.168.0.101	SMB	NT Create AndX Response, FID: 0xc002
192.168.0.202	SMB	Trans2 Request, QUERY_FILE_INFO, FID: 0xc002, Query File Internal Info
192.168.0.101	SMB	Trans2 Response, FID: 0xc002, QUERY_FILE_INFO
192.168.0.202	SMB	Trans2 Request, QUERY_FS_INFO, Info Allocation
192.168.0.101	SMB	Trans2 Response, QUERY_FS_INFO
192.168.0.202	SMB	Write AndX Request, FID: 0xc002, 1 byte at offset 165024
192.168.0.101	SMB	Write AndX Response, FID: 0xc002, 1 byte
192.168.0.202	SMB	Trans2 Request, QUERY_FILE_INFO, FID: 0xc002, Query File Standard Info
192.168.0.101	SMB	Trans2 Response, FID: 0xc002, QUERY_FILE_INFO

Figure 14: Decoded SMB packets of Conficker attack

After the copying itself to another host, Conficker creates a Scheduled job to run the new copy. Figure 15 shows the decoded packets where the job is created to run bvwwtbp.fid using rundll32.exe at 22:00 (Job Time of 79,200,000 milli-seconds) on host 192.168.0.202.

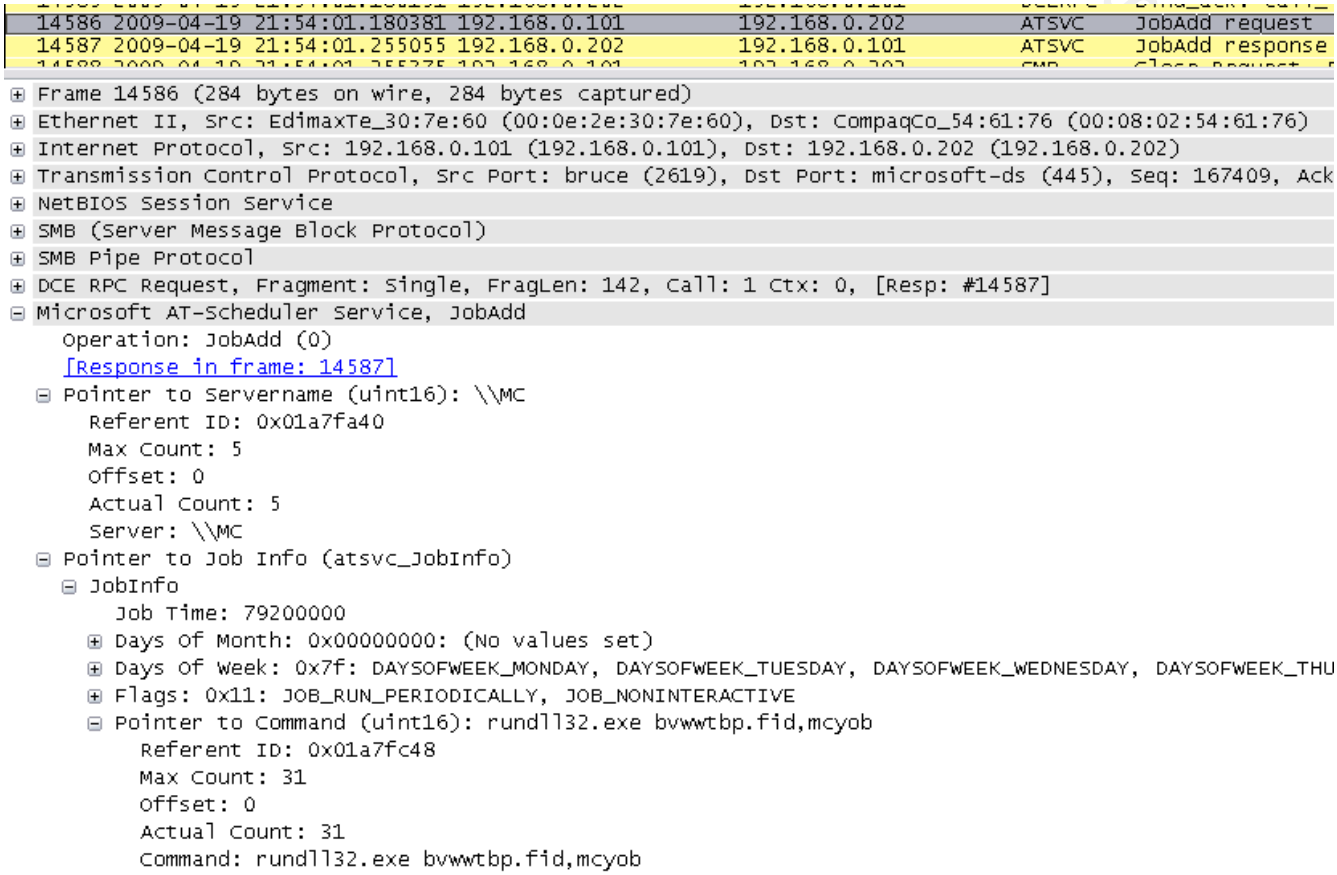

Figure 15: Create scheduled job to run bvwwtbp.fid

Conficker Eye Chart

The Conficker Eye Chart (Conficker Working Group, 2009) is another method that can be used to interactively check if a host is infected. The Conficker Eye Chart is a web page that displays logos from security and operating system organizations. If some images are blocked, it is likely a Conficker infection. Figure 16 shows an example Conficker Eye Chart of a host infected with Conficker.B. The lab tests are consistent with a Conficker.B infection. F-Secure and TrendMicro logos are blocked however, the SecureWorks logo is not.

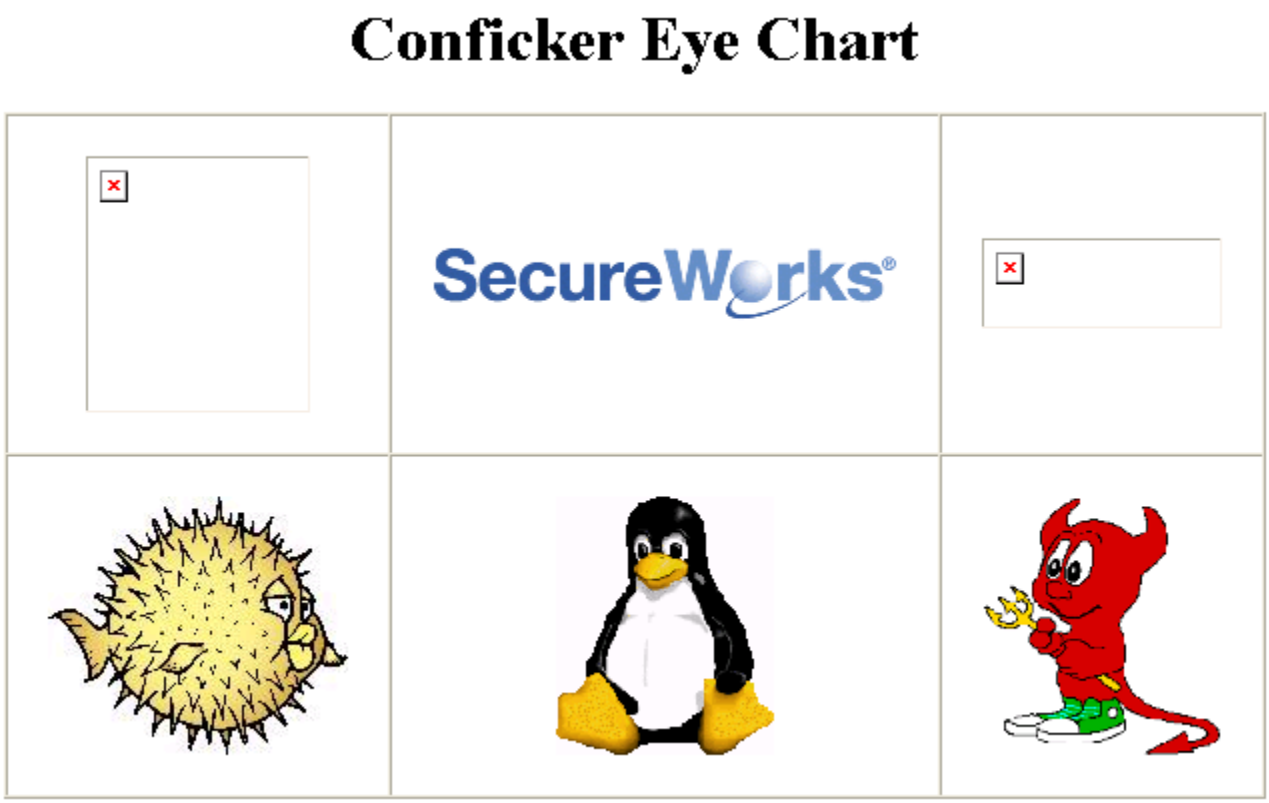

Figure 16: Conficker Eye chart example of Conficker.B infection.

Logging and Monitoring

The logging and monitoring data is also used to check and verify Conficker behavior. One of the first things Conficker does is a DNS request to ip address checking sites such as www.whatsmyipaddress.com. An example query from the Active Directory DNS log shown on the following line:

```
02014BF0 UDP Rcv 192.168.0.101    3e72    Q [0001    D    NOERROR] A
(3)www(16)whatsmyipaddress(3)com(0)
```

The request for the web page is also logged by the external firewall and captured on the syslog server. An example firewall log is shown below:

```
permitted tcp inside/192.168.0.101(1275) ->
<010><009>outside/5.9.87.13(80)
```

Windows Event Logs

The Windows Event logs on target hosts are also used to check on Conficker behavior. Successful and Failed logon events were logged to the Windows Event Log.

Registry Changes

The last method to check a host for a Conficker infection is to use a registry dump of selected keys. A script that sends registry output to a text file is run before and after infection. The diff command was used to check for differences before and after an infection. Registry values that changed are shown in Figure 17 and Figure 18. Figure 17 shows the values before infection. Figure 18 shows the values after infection.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\
Advanced\Folder\Hidden\SHOWALL]
"RegPath"="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Adva
nced"
"Text"="@shell32.dll,-30500"
"Type"="radio"
"CheckedValue"=dword:00000001
"ValueName"="Hidden"
"DefaultValue"=dword:00000002

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\BITS]
"Type"=dword:00000020
"Start"=dword:00000003

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauserv]
"Type"=dword:00000020
"Start"=dword:00000002

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SvcHost]
"LocalService"=hex(7):41,00,6c,00,65,00,72,00,74,00,65,00,72,00,00,00,5
7,00,65,\
00,62,00,43,00,6c,00,69,00,65,00,6e,00,74,00,00,00,4c,00,6d,00,48,00,6f
,00,\
73,00,74,00,73,00,00,00,52,00,65,00,6d,00,6f,00,74,00,65,00,52,00,65,00
,67,\
00,69,00,73,00,74,00,72,00,79,00,00,00,75,00,70,00,6e,00,70,00,68,00,6f
,00,\
73,00,74,00,00,00,53,00,53,00,44,00,50,00,53,00,52,00,56,00,00,00,00,00
"NetworkService"=hex(7):44,00,6e,00,73,00,43,00,61,00,63,00,68,00,65,00
,00,00,\
  00,00
"netsvcs"=hex(7):36,00,74,00,6f,00,34,00,00,00,41,00,70,00,70,00,4d,00,
67,00,\
```

```
6d,00,74,00,00,00,41,00,75,00,64,00,69,00,6f,00,53,00,72,00,76,00,00,00
,42,\
... lines removed ...
00,70,00,72,00,6f,00,76,00,00,00,77,00,73,00,63,00,73,00,76,00,63,00,00
,00,\
  57,00,6d,00,64,00,6d,00,50,00,6d,00,53,00,4e,00,00,00,00,00
```

Figure 17. Selected Registry values before infection.

In comparing the before and after output, changes made are consistent with Conficker analyses (Porras, 2009). There are 5 registry keys that are changed from those selected for monitoring. First the Explorer\Advanced key is changed to hide files from view. This allows Conficker files to be hidden from the user. Next, the Applets key is added when a host is infected. The next difference is the BITS and Update services being disabled. Lastly, a service is added to netsvcs. The service name is represented by 17 hexadecimal bytes which added to the end of the netsvcs key. For the example in Figure 18, after removing 00s, the result is "72:74:71:61:62:75:6a:74" which converts to the service name "rtqabujt".

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\
Advanced\Folder\Hidden\SHOWALL]
"RegPath"="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Adva
nced"
"Text"="@shell32.dll,-30500"
"Type"="radio"
"CheckedValue"=dword:00000000
"ValueName"="Hidden"
"DefaultValue"=dword:00000002

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Applets]
@=dword:00000001
"dl"=hex:02,00,00,00,c0,a8,00,ca,00,66,a9,e8,75,c1,c9,01,c0,a8,00,66,90
,5f,ac,\
  2f,75,c1,c9,01
"ds"=hex:00,00,00,00,00,00,00,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\BITS]
"Type"=dword:00000020
"Start"=dword:00000004

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\wuauserv]
"Type"=dword:00000020
"Start"=dword:00000004

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SvcHost]
```

```
"LocalService"=hex(7):41,00,6c,00,65,00,72,00,74,00,65,00,72,00,00,00,5
7,00,65,\
00,62,00,43,00,6c,00,69,00,65,00,6e,00,74,00,00,00,4c,00,6d,00,48,00,6f
,00,\
73,00,74,00,73,00,00,00,52,00,65,00,6d,00,6f,00,74,00,65,00,52,00,65,00
,67,\
00,69,00,73,00,74,00,72,00,79,00,00,00,75,00,70,00,6e,00,70,00,68,00,6f
,00,\
73,00,74,00,00,00,53,00,53,00,44,00,50,00,53,00,52,00,56,00,00,00,00,00
"NetworkService"=hex(7):44,00,6e,00,73,00,43,00,61,00,63,00,68,00,65,00
,00,00,\
  00,00
"netsvcs"=hex(7):36,00,74,00,6f,00,34,00,00,00,41,00,70,00,70,00,4d,00,
67,00,\
6d,00,74,00,00,00,41,00,75,00,64,00,69,00,6f,00,53,00,72,00,76,00,00,00
,42,\
... lines removed ...
00,70,00,72,00,6f,00,76,00,00,00,77,00,73,00,63,00,73,00,76,00,63,00,00
,00,\
57,00,6d,00,64,00,6d,00,50,00,6d,00,53,00,4e,00,00,00,72,00,74,00,71,00
,61,\
  00,62,00,75,00,6a,00,74,00,00,00,00,00
```

Figure 18. Selected Registry values after infection.

Attack Results Summary

The Conficker behaviors from the lab testing were consistent with those reported by security researchers. Some additional in depth understanding was obtained by conducting an infection in a lab network. The first step of the infection was inserting an infected USB Flash drive into host C1. The autorun file caused Conficker to run and C1 was immediately infected. C1 began scanning the local network and infected any hosts vulnerable to MS08-067. In the lab example C2 was vulnerable and was infected via exploit of MS08-067. Infected hosts also attempted password guessing. Lab host FS had a weak administrator password that was on the list of Conficker passwords (see Appendix C). It was infected via the weak password. Any USB Flash drive inserted into an infected host was also infected. Some additional observations are:

- Even though the Firewall was Off, Conficker added an exception for its listener.
- Conficker attempted port 445 connections for the MS08-067 exploit on the local network, but no attempts were observed for ip addresses outside the local network 192.168.0.0/24.

In summary, in an unpatched and unprotected network Conficker will quickly compromise multiple hosts in the network. Once this occurs, the Incident Handling process will be required to address the incident.

3.2.4. Attack Incident Handling

In scenario 1, the network was unprotected and that resulted in many infected hosts. The result was a major incident which would likely interrupt important services. In this scenario, any existing monitoring would be used for identification. For Incident Handling in this scenario, the Identification, Containment, Eradication, Recovery and Lessons Learned. phases will be discussed. Identification would likely come from domain account lockouts or possibly an alert Engineer checking the firewall logs. Once identified, the incident must be contained.

Given that this malware actively attacks hosts over the network, physically removing hosts from the network is a likely containment step. In addition, USB flash drives and file shares must also be quarantined for clean up. After containing the incident, the infection must be eradicated.

Eradication is best accomplished with commercial anti-virus (AV) software once a signature is available. Given the hosts in this scenario do not have AV software installed, deployment of an AV solution is a likely approach. An alternative would be to use one of the many free removal tools offered by security vendors. Note that these tools remove the threat, but do not completely restore the system to the pre-infected state. For example, services remain disabled and the firewall exception is not removed. After the infection is eradicated, the recovery phase can begin.

Once infected hosts are cleaned and all related shares and USB flash drives are tracked down. The systems can be patched to prevent re-infection and returned to service as part of the Recovery phase.

For Lessons Learned policies should be reviewed regarding the use of USB flash drives and applying operating system patches. Additionally, addition protection measures could be considered such as deploying a Host Intrusion Prevention system like CSA.

3.3. Conficker attack with CSA

3.3.1. Overview

The malware attack with CSA scenario was carried out with the lab configuration described previously. The goal was to demonstrate CSA protection from worm propagation in an environment with HIPS and identify the required incident handling. Two scenarios are used to analyze how CSA protects from Conficker. In scenario 2a all hosts are protected by CSA. In scenario 2b all hosts are protected with CSA except host C1, patient zero.

3.3.2. Attack Vector

The two scenarios represent two of Conficker's attack vectors. In scenario 2a, host C1 is protected with CSA. In this scenario, CSA stops the attack when the USB Flash drive is inserted. In scenario 2b, C1 is not protected, but other hosts in the network are. Scenario 2b represents the case where an unmanaged host without CSA is on the local network. In scenario 2b, C1 becomes infected with the USB Flash drive. The infected host (C1) then attempts to exploit the MS08-067 vulnerability over the network. The attacks are stopped from propagating to other hosts by CSA.

3.3.3. Attack Results

Scenario 2a

In scenario 2a, CSA protects the client by stopping the malware on the infected USB Flash drive from running. Figure 19 shows an alert generated in CSA when the malware attempts to run the dll file on the USB Flash drive. The red X icon indicates that the operation was denied.

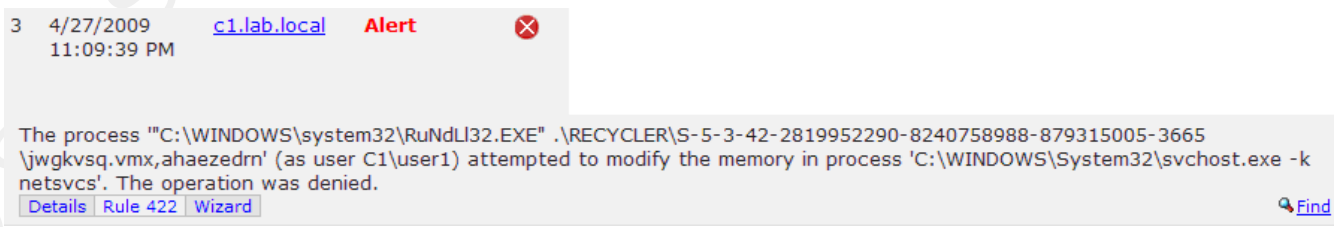

Figure 19: CSA Alert blocking Conficker.

The malware is stopped by Rule 422 which is part of the default Desktop protection. Rule 422 [All Applications (not White List), Write memory owned by system application] is part of the "Security –Policy Violation –Modify another process" Rule Module. As the name implies it stops a process from modifying another process. This Rule Module is part of the "Anti-Virus – Behavior based (desktops)" Policy. This Policy is attached to the Desktops Group and is therefore applied to all Desktops in the group.

Because the initial attack is stopped, the Incident Response required is relatively minor. No hosts were infected, but there is an infected USB Flash Drive in the environment. The infected USB Flash Drive should be quarantined. An investigation should determine any other hosts it has been attached to and those hosts should be checked for infection.

Scenario 2b

In scenario 2b, the unprotect patient zero C1 was infected as expected from the infected USB Drive. The infected host then attempts all the propagation attacks described in Section 1.5.4. Two of the attacks are of interest the guessing weak Administrator passwords and the MS08-067 exploit.

The attack against weak passwords needs to be mitigated using a password policy. Strong passwords and account lockout will effectively block password guessing attacks. CSA cannot block password guessing attacks, but it can alert on a correlation of account lockout events from the Windows Event log.

As part of scenario 2a and 2b, strong passwords and account lockout was configured on the target hosts. This mitigated Conficker's password guessing attack vector. An example Windows event log showing an account lockout event is shown in Figure 20. An example of an account lockout event displayed in CSA is shown in Figure 3.

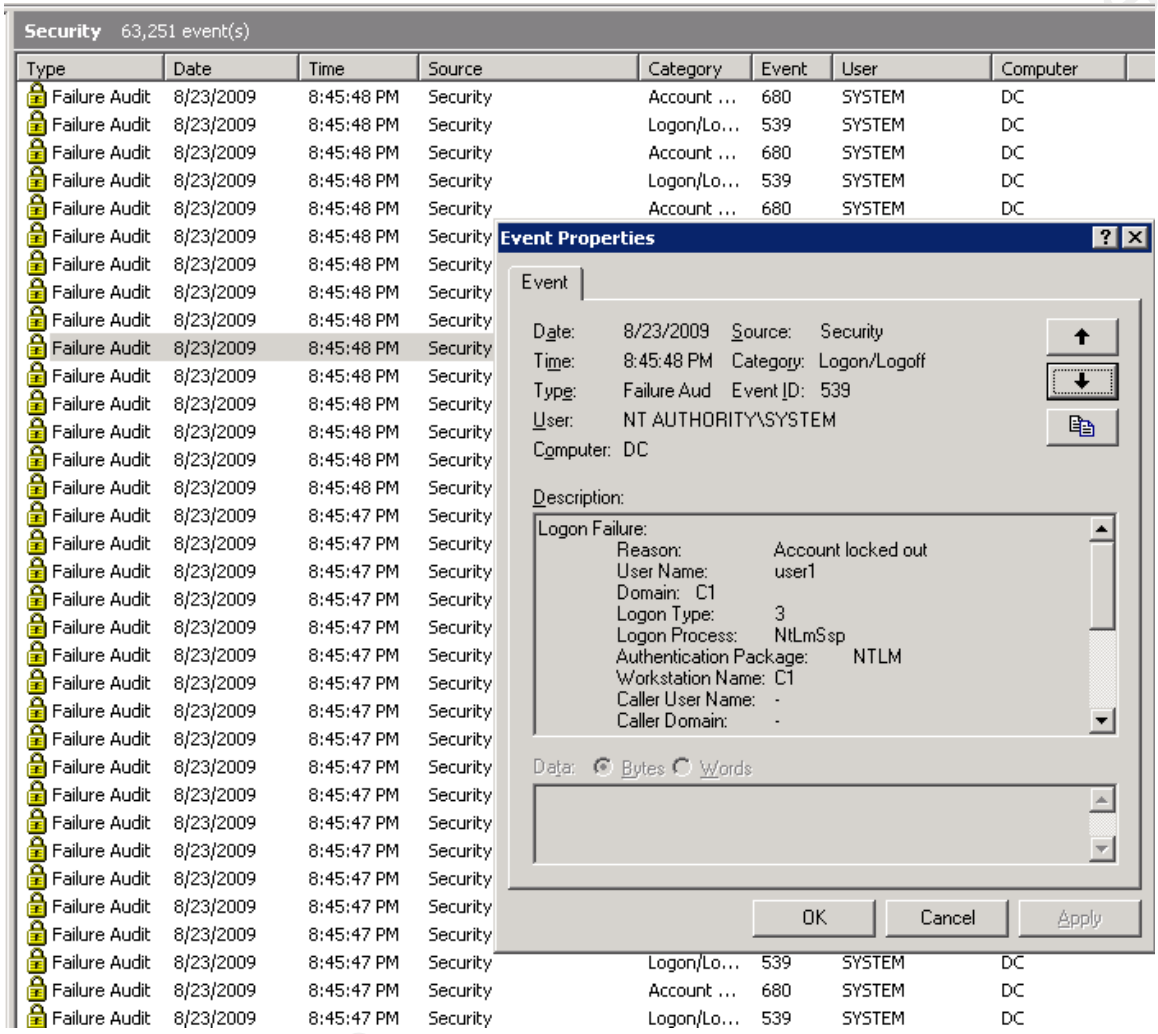

Figure 20: Account locked out event log

The other attack of interest is Conficker's exploit of the MS08-067 vulnerability. In the lab test, Conficker began scanning the local network for other Windows hosts immediately after infection. Once it finds a Windows host, it attempts the exploit. The other hosts which were protected with CSA were not infected.

Figure 21 shows CSA blocking a network based attack to exploit the MS08-067 vulnerability.

Greg Farnham

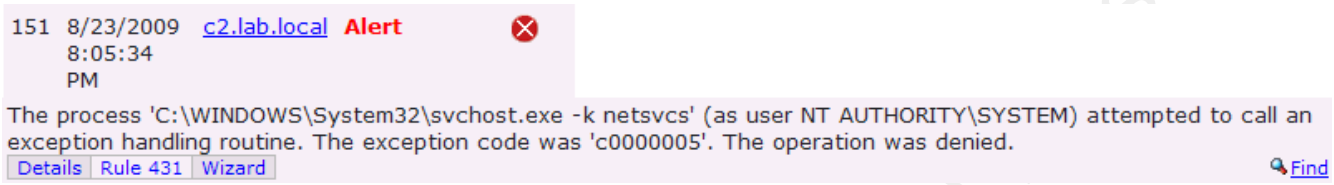

Figure 20: CSA Alert Rule 431

The attack was blocked by Rule 431. Rule 431 [Windows services requiring stack recovery, Handle Exceptions] is part of the "Security - Stack recovery for critical services" Rule Module. As the name implies it blocks Windows Services requiring stack recovery exception handling. Rule 431 applies to services in the system folder matching one of the following names: (lsass.exe, mstask.exe, services.exe, smss.exe, spoolss.exe, spoolsv.exe, svchost.exe, tcpsvcs.exe, winmgmt.exe). This Rule Module is part of the "Firewall - Centrally Managed (desktops)" Policy. This Policy is attached to the Desktops Group and is therefore applied to all Desktops in the group.

With CSA protection in place and a strong password policy, the infection is limited to the single un-managed host. Therefore, the incident handling is limited to this host (C1).

3.3.4. Attack Incident Handling

Because all of the targets except C1 were protected, the incident was contained to one host. This greatly reduced the scope of this incident and the resources required for incident handling. As part of the preparation phase, CSA was deployed to protect hosts on the network. CSA was useful in the identification phase with alerts being generated when Conficker attacked other hosts via the MS08-067 vulnerability. This attack triggered alerts from Rule 431. It also generated CSA events from the Windows Event log showing account lockouts. The Event logging is configured in Rule 821. The eradication and recovery phases are executed as previously described. For Lessons Learned, CSA is used for forensic analysis and to plan deployment of CSA to previously unprotected hosts.

3.4. Comparison

Three scenarios were tested. Scenario 1 where there was no CSA protection. Scenario 2a where all hosts had CSA protection and Scenario 2b where all hosts except C1 had CSA protection. The results of a Conficker incident in an unprotected environment versus one protected with CSA show value in having behavior based Anti Virus protection. In an unpatched unprotected environment, the Conficker worm rapidly spreads using multiple attack vectors. This results in a major incident which would disrupt business and require a major incident response. With CSA in place and used in multiple phases of the incident handling process the attack is largely mitigated. In scenario 2a, all hosts are protected and the incident is contained to just investigating the infected USB Flash drive. In scenario 2b, one un-managed host is infected, but the rest of the network is protected. In both scenario 2a and 2b, the attack is largely mitigated and the scope of the resulting incident response is greatly reduced.

4. Conclusions

Conficker is a sophisticated and dangerous worm. It has multiple attack vectors and takes many measures to protect itself from detection and removal. Conficker behaviors were analyzed and methods for identifying Conficker infections using scripts, traffic analysis and manual methods were documented (section 3.2.3).

CSA is a behavior based endpoint protection system that provides zero day protection. The version used in the lab was released before the MS08-067 vulnerability was discovered and before Conficker was released. By blocking behaviors such as "modifying memory of system processes, Rule 422 and "calling exception handling", Rule 431 CSA is able to protect hosts from Conficker and similar malware. This protection is available before Anti-virus vendors have signatures to detect the malware. CSA includes many other policies to stop malware such as blocking key loggers. With behavior based policies CSA can protect hosts without using signatures. CSA has successfully blocked a long list of malware (Cisco-2, 2008) with zero day protection.

Guidance was provided for implementing CSA to support the incident handling process. The preparation phase is augmented by including the CSA implementation in

the incident handling process. The identification process is enhanced by configuring CSA to send alerts to the existing alert email list, configuring Windows Events to be centrally collected and leveraging the default event correlation (section 2.2). The containment phase is improved by providing a custom CSA lockdown policy (section 2.3). The recovery phase is also affected by removing the CSA lockdown policy. CSA is also included in the lessons learned phase. These concepts were demonstrated in a lab environment by initiating a Conficker incident in three different scenarios.

Organizations that implement behavior based malware protection will benefit by having increased protection over standard signature based anti-virus. Behavior based anit-virus solutions are capable of providing zero day protection. By considering the incident handling process when deploying new security solutions, organization can increase their effectiveness in incident handling. The primary impact will be in the identification, containment and lessons learned phases.

5. Appendices

5.1. Appendix A

A few simple batch scripts were used in monitoring the Conficker lab tests. The first is a batch file which is used to repeatedly loop and output service status and any scheduled jobs. This is used to time stamp when a Conficker infection occurs. WMI commands based on previous SANS student work (Proffitt 2009).

```
rem check conficker
set outfile=%computername%-wmi.txt
date /t  > %outfile%

del /f /q sleep.vbs

> "sleep.vbs" ECHO WScript.Sleep 1 * 1000

rem wmic qfe where hotfixid="KB958644" list brief  >>
%outfile%

:Label1

time/t  >> %outfile%

wmic job >> %outfile%
echo - >> %outfile%
wmic service where (name="BITS" OR name="Wuauserv") get
name, state  >> %outfile%
echo - >> %outfile%

REM run delay script
CSCRIPT //NoLogo "sleep.vbs"
GOTO Label1
```

The second batch script was used to dump selected registry keys before and after a Conficker infection.

```
rem export conficker related reg keys
echo off

set ranfile=aaaaaa
set ranserv=aaaaaa
set sd=Parameters
```

```
rem tcp connections
reg export HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters a

rem remove rems and edit set values above if you know random values
rem reg export
HKLM\SYSTEM\CurrentControlSet\Services\%ranfile%\Parameters\ServiceDll
b
rem reg export HKLM\SYSTEM\CurrentControlSet\Services\%ranserv% c

rem run
reg export HKCU\Software\Microsoft\Windows\CurrentVersion\Run d

rem stay hidden
reg export
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced e
reg export
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder
\Hidden\SHOWALL f

rem applets
reg export HKCU\Software\Microsoft\Windows\CurrentVersion\Applets g
reg export HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Applets h

rem services
reg export HKLM\SYSTEM\CurrentControlSet\Services\BITS i
reg export HKLM\SYSTEM\CurrentControlSet\Services\wscsvc j
reg export HKLM\SYSTEM\CurrentControlSet\Services\wuauserv k
rem reg export HKLM\SYSTEM\CurrentControlSet\Services\WinDefend l
reg export HKLM\SYSTEM\CurrentControlSet\Services\ERSvc m
rem reg export HKLM\SYSTEM\CurrentControlSet\Services\WerSvc n

rem disable windows security center notifications
reg export
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObj
ects o

rem service auto start
reg export "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost"
p

rem firewall bypass
reg export
HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\Firewall
Policy\StandardProfile\GloballyOpenPorts\List q

reg export HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Nls r
rem commented items removed
type a  d e f g h i j k  m  o p q r> conficker-%computername%.txt
```

The last script is running the nmap script engine to detect Conficker infections. It runs repeatedly in a loop and searches the output for the word "Likely" which indicates a Conficker infection.

```
rem check conficker
date /t  > nmapc.out

:Label1
time/t  >> nmapc.out
nmap -PN -T4 -p139,445 -n -v --script smb-check-vulns,smb-os-discovery
--script-args safe=1 -oN nmap.out  192.168.0.101, 192.168.0.102,
192.168.0.201, 192.168.0.202
time/t >> nmap.full
type nmap.out >> nmap.full
type nmap.out | find "Likely" >> nmapc.out
REM delay
ping -n 300 127.0.0.1

GOTO Label1
```

5.2. Appendix B

Results from virustotal.com on Conficker sample used in lab.

Antivirus	Version	Last	Result
a-squared	4.5.0.24	2009.08.17	Net-Worm.Win32.Kido!IK
AhnLab-V3	5.0.0.2	2009.08.17	Win32/Conficker.worm.165025
AntiVir	7.9.1.1	2009.08.17	Worm/Conficker.Y.03
Antiy-AVL	2.0.3.7	2009.08.17	Worm/Win32.Kido.gen
Authentium	5.1.2.4	2009.08.17	W32/Conficker!Generic
Avast	4.8.1335.0	2009.08.17	Win32:Confi
AVG	8.5.0.406	2009.08.17	I-Worm/Generic.COL
BitDefender	7.2	2009.08.17	Worm.Generic.63560
CAT-QuickHeal	10.00	2009.08.17	Worm.Conficker.b
ClamAV	0.94.1	2009.08.17	Worm.Kido-34

Comodo	1964	2009.08.17	NetWorm.Win32.Kido.~A
DrWeb	5.0.0.12182	2009.08.17	Win32.HLLW.Shadow.based
eSafe	7.0.17.0	2009.08.17	-
eTrust-Vet	31.6.6681	2009.08.17	Win32/Conficker
F-Prot	4.4.4.56	2009.08.16	W32/Conficker!Generic
F-Secure	8.0.14470.0	2009.08.17	Worm:W32/Downadup.gen!A
Fortinet	3.120.0.0	2009.08.17	W32/Conficker.B!worm
GData	19	2009.08.17	Worm.Generic.63560
Ikarus	T3.1.1.68.0	2009.08.17	Net-Worm.Win32.Kido
Jiangmin	11.0.800	2009.08.17	I-Worm/Kido.c
K7AntiVirus	7.10.820	2009.08.17	Net-
Kaspersky	7.0.0.125	2009.08.17	Net-Worm.Win32.Kido.ih
McAfee	5712	2009.08.17	W32/Conficker.worm.gen.a
McAfee+Artemis	5712	2009.08.17	W32/Conficker.worm.gen.a
McAfee-GW-	6.8.5	2009.08.17	Worm.Conficker.W
Microsoft	1.4903	2009.08.17	Worm:Win32/Conficker.C
NOD32	4342	2009.08.17	a variant of
Norman		2009.08.17	W32/Conficker.CR
nProtect	2009.1.8.0	2009.08.17	Worm/W32.Kido.165025
Panda	10.0.0.14	2009.08.16	W32/Conficker.C.worm
PCTools	4.4.2.0	2009.08.17	Net-Worm.Kido!sd6
Prevx	3.0	2009.08.17	High Risk Worm
Rising	21.43.04.00	2009.08.17	Worm.Win32.Undef.dc
Sophos	4.44.0	2009.08.17	Mal/Conficker-A
Sunbelt	3.2.1858.2	2009.08.16	Worm.Win32.Downad.Gen (v)

Symantec	1.4.4.12	2009.08.17	W32.Downadup.B
TheHacker	6.3.4.3.383	2009.08.13	W32/Conficker.gen
TrendMicro	8.950.0.1094	2009.08.17	WORM_DOWNAD.AD
VBA32	3.12.10.9	2009.08.17	Worm.Win32.kido.110
ViRobot	2009.8.17.1887	2009.08.17	Worm.Win32.Conficker.165025
VirusBuster	4.6.5.0	2009.08.17	Worm.Kido.KL

5.3. Appendix C

Passwords used by Conficker for password guessing (F-Secure-1, 2009).

```
[username]                  333333                  Internet
[username][username         3333333                 Login
]                           33333333                Password
[reverse_of_usernam         44444                   a1b2c3
e]                          444444                  aaaaa
00000                       4444444                 abc123
0000000                     44444444                academia
00000000                    54321                   access
0987654321                  55555                   account
11111                       555555                  admin
111111                      5555555                 admin1
1111111                     55555555                admin12
11111111                    654321                  admin123
123123                      66666                   adminadmin
12321                       666666                  administrator
123321                      6666666                 anything
12345                       66666666                asddsa
123456                      7654321                 asdfgh
1234567                     77777                   asdsa
12345678                    777777                  asdzxc
123456789                   7777777                 backup
1234567890                  77777777                boss123
1234abcd                    87654321                business
1234qwer                    88888                   campus
123abc                      888888                  changeme
123asd                      8888888                 cluster
123qwe                      88888888                codename
1q2w3e                      987654321               codeword
22222                       99999                   coffee
222222                      999999                  computer
2222222                     9999999                 controller
22222222                    99999999                cookie
33333                       Admin                   customer
```

Greg Farnham

```
database
default
desktop
domain
example
exchange
explorer
files
foobar
foofoo
forever
freedom
games
home123
ihavenopass
internet
intranet
killer
letitbe
letmein
login
lotus
love123
manager
market
money
monitor
mypass
mypassword
mypc123
nimda
nobody
nopass
nopassword
nothing
office
oracle
owner
pass1
pass12
pass123
passwd
password
password1
password12
password123
private
public
pw123
q1w2e3
qazwsx
qazwsxedc
qqqqq
qwe123
qweasd
qweasdzxc
qweewq
qwerty
qwewq
root123
rootroot
sample
secret
secure
security
server
shadow
share
student
super
superuser
supervisor
system
temp123
temporary
temptemp
test123
testtest
unknown
windows
work123
xxxxx
zxccxz
zxcvb
zxcvbn
zxcxz
zzzzz
```

5.4. Appendix D

Conficker blocks access to domains which contain the following strings (Sophos, 2009).

```
agnitum
ahnlab
anti-
antivir
arcabit
avast
avgate
avira
bothunter
castlecops
ccollomb
centralcommand
clamav
comodo
computerassociates
conficker
cpsecure
cyber-ta
defender
downad
drweb
dslreports
emsisoft
esafe
eset
etrust
ewido
f-prot
f-secure
fortinet
free-av
freeav
gdata
```

Greg Farnham

```
grisoft
hackerwatch
hacksoft
hauri
ikarus
jotti
k7computing
kaspersky
kido
malware
mcafee
microsoft
mirage
msftncsi
msmvps
mtc.sri
networkassociates
nod32
norman
norton
onecare
panda
pctools
prevx
ptsecurity
quickheal
removal
rising
rootkit
safety.live
securecomputing
secureworks
sophos
spamhaus
spyware
sunbelt
symantec
technet
threat
threatexpert
trendmicro
trojan
virscan
virus
wilderssecurity
windowsupdate
avg.
avp.
bit9.
ca.
cert.
gmer.
kav.
llnw.
llnwd.
msdn.
msft.
nai.
sans.
vet.
```


Greg Farnham

5.5. Appendix E

Conficker uses a few lists of words with typical system terms for naming services and keys. By using these names, it makes Conficker harder to spot when reviewing system configurations (Yason, 2009).

Service names are created by using two strings from the list below.

Audit	Helper	Security
Backup	Image	Server
Boot	Installer	Shell
Browser	Logon	Storage
Center	Machine	Support
Component	Management	System
Config	Manager	Task
Control	Microsoft	Time
Discovery	Monitor	Trusted
Driver	Network	Universal
Event	Notify	Update
Framework	Policy	Windows
Hardware	Power	

Key names are created using two strings. The first is from the list below.

App	Lanman	Trk
Audio	Net	W32
DM	Ntms	win
ER	Ras	Wmdm
Event	Remote	Wmi
help	Sec	wsc
Ias	SR	wuau
Ir	Tapi	xml

The second string in the key is from the list below.

```
access
agent
auto
logon
man
mgmt
mon
prov
serv
Server
Service
srv
svc
System
Time
```

5.6. Appendix F

Conficker terminates any process that has the following substrings (Sophos, 2008)

```
autoruns
avenger
confick
downad
filemon
gmer
hotfix
kb890
kb958
kido
klwk
mbsa.
mrt.
mrtstub
ms08-06
procexp
procmon
regmon
scct_
sysclean
tcpview
unlocker
wireshark
```

Greg Farnham

6. References

Bernardino, Jeffrey F. (2008). WORM_DOWNAD.AD. Retrieved April 4, 2009, from TrendMicro Web site: http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_DOWNAD.AD&VSect=T

Cisco-1, (2008) Worm: W32/Conficker.worm. Retrieved August 26, 2009, from Cisco Web site:

http://tools.cisco.com/security/center/viewAlert.x?alertId=17121

Cisco-2, (2008) Cisco Security Agent and Microsoft Vulnerability MS08-067. Retrieved April 4, 2009, from Cisco Web site: http://www.cisco.com/en/US/prod/collateral/vpndevc/ps5707/ps5057/product_bulletin_c25-514332.html

Cisco-3, (2009) Installing Management Center for Cisco Security Agents 6.0. Retrieved September 28, 2009, from Cisco Web site: http://www.cisco.com/en/US/partner/docs/security/csa/csa60/install_guide/Installing_CSAMC60.html

Conficker Working Group, (2009). Conficker Working Group. Retrieved April 4, 2009, from Conficker Working Group Web site: http://www.confickerworkinggroup.org/wiki/pmwiki.php/Main/HomePage

F-Secure-1, F-Secure (2009). Worm:W32/Downadup.AL. Retrieved April 4, 2009, from F-Secure Web site: http://www.f-secure.com/v-descs/worm_w32_downadup_al.shtml

F-Secure-2, (2009). Calculating the Size of the Downadup Outbreak. Retrieved April 4, 2009, from F-Secure Web site: http://www.f-secure.com/weblog/archives/00001584.html

Fitzgibbon, Niall (2009, April 1). Conficker.C A Technical Analysis. Retrieved July 19, 2009, from Sophos Web site: http://www.sophos.com/sophos/docs/eng/marketing_material/conficker-analysis.pdf

Ferguson, Paul (2009, April, 14). The DOWNAD/Conficker Jigsaw Puzzle. Retrieved September 8, 2009, from TrendMicro Web site: http://blog.trendmicro.com/the-downadconficker-jigsaw-puzzle/

Ferrer, Zarestel (2009, January 5). Win32/Conficker.B. Retrieved April 4, 2009, from Computer Associates Web site: http://www.ca.com/securityadvisor/virusinfo/virus.aspx?id=76852

Fyodor (2009, March 30). map 4.85BETA5: Now with Conficker detection!. Retrieved September 8, 2009, from Insecure.org Web site: http://seclists.org/nmap-dev/2009/q1/0870.html

Greg Farnham

Goodin, Dan (2009, January, 16). Superworm seizes 9m PCs, 'stunned' researchers say. The Register, Retrieved April 4, 2009, from http://www.theregister.co.uk/2009/01/16/9m_downadup_infections/

Macalintal, Ivan (2009, April 8). DOWNAD/Conficker Watch: New Variant in The Mix?. Retrieved September 29, 2009, from TrendMicro Web Site: http://blog.trendmicro.com/downadconficker-watch-new-variant-in-the-mix/

Mandia, Kevin (2001). *Incident Response Investigating Computer Crime*. Berkeley, CA: Osbourne McGraw Hill.

Microsoft, (2008) Microsoft Security Bulletin MS08-067 – Critical. Retrieved April 4, 2009, from Microsoft Web site: http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx

Porras, Phillip (2009, February 4). An Analysis of Conficer's Logic and Rendezvous Points. Retrieved April 14, 2009, from SRI Web site: http://mtc.sri.com/Conficker/

Proffitt, Tim (2009, March 1). GIAC Enterprises Downadup Incident. Retrieved July 15, 2009, from SANS Web site: http://www.sans.edu/resources/student_projects/200903_01.pdf

Skoudis, E. (2005). *SANS Security 504 Course Books*. SANS Institute.

Sophos, (2008). Mal/Conficker-B. Retrieved April 4, 2009, from Sophos Web site: http://www.sophos.com/security/analyses/viruses-and-spyware/malconfickerb.html

Sullivan, Chad (2005). *Cisco Security Agent*. Indianapolis, IN: Cisco Press.

Van Wyk, Kenneth R. (2001). *Incident Response Planning & Management*. Sebastapol, CA: O'Reilly.

Yason, M (2009, January 22). Conficker. Retrieved August 18, 2009, from IBM Internet Security Systems Web site: http://www.iss.net/threats/conficker.html