Global Information Assurance Certification Paper

A No-Budget Approach to the Containment of Malware Traffic

GIAC (GSEC) Gold Certification

Author: Paul Ackerman PAckerman@VacciNetLLC.com
Advisor: Richard Carbone

Accepted: January 10th, 2015

Abstract

Many small/medium sized businesses have little budget for Information Security yet face the same malware threat as larger organizations. In Information Security, we say that prevention is ideal and detection is necessary but what comes after detection? Specifically, what should a small team do to contain a malware infection? This paper is for those readers that do not have expensive tools to defend against malware and are left wondering how to contain an infection. We will begin with a PowerShell script to respond to these email-borne threats by searching through Exchange logs to find recipients of a specific message and then proceeding to quarantine the messages to prevent additional infection. Next, a Windows virtual machine will be transformed into a crime-fighting sidekick with tricks like static ARP entries for DNS servers and gateways to enable the capture of network traffic using Wireshark. Once we know where the malware is going we will create DNS zones with an internal IP address so we can redirect the malware to our own server and intercept outbound connections from infected machines enabling us to track down any others that may have evaded detection. In short, what follows is a no-budget approach to analyzing and containing a malware infection using freeware and open-source tools. This paper and its various scripts and website-based approach are the independent work of the author. After conducting a Google-based literature review, it has become apparent that while others have examined the same issues as the author, his work is unique because he has proposed a series of useful scripts that will enable the reader to perform actual and useful analyses.

1. Introduction

There are many unique pieces of malware that can be categorized into a few different types. Adware creates unwanted popup advertisements to generate revenue for its creators (Adware, 2014). Spyware, which is often bundled with adware, tracks a user's Internet usage and can steal personal information, account passwords or change the configuration of a computer (Microsoft, 2014). Bots are malicious programs designed to listen for and execute commands provided by an attacker and are often used in attacks against other systems (Glossary of Terms, 2014). Ransomware is a relatively new form of malware that encrypts documents on a computer and forces users to pay a ransom in order to retrieve the key to decrypt their files. Trojans are malicious programs that are bundled with harmless applications and are known to create backdoors and install rootkits for their creators (Glossary of Terms, 2014). One thing that nearly all types of malware have in common is that they rely on a connection to the Internet in order to receive commands or send information back to the attacker. In the author's experience, some malware samples will even stop executing if they are unable to reach a server from which they are supposed to receive commands. These servers are known as Command and Control (C2 or CnC) servers. Ransomware that cannot reach a C2 server cannot encrypt a user's documents since the key is generated on or copied to the C2 server. Password-stealing malware would be unable to transmit the stolen data if it cannot reach the Internet. Blocking Internet access to these servers would be of great use. Additionally, what if it could be used to help prevent the infection in the first place?

A common way many machines become infected is through non-malicious programs being sent to the user through email or a drive-by attack from a website. This "downloader" program does not contain any malicious code itself; it just downloads and executes another program from another website. The downloader itself often makes it past antivirus and IPS systems since it is not malicious and gives the creator a chance to download and execute a Trojan or other malware directly. As of January 2012, there were more than eight million unique Trojan downloaders meaning this an extremely common method of infection. (MSFT-MMPC, 2012) By blocking outbound connections to C2

servers and thus preventing information from leaving the network, we can drastically reduce the impact of a malware infection in the network.

In addition to blocking outbound connection attempts from infected machines, we can also help prevent additional infections by identifying and responding to malware threats received through email. It is often easier to exploit a human being with social engineering than to find a vulnerability and craft a new exploit for a browser or application. Thus, email continues to be a common venue for attackers to infect computers with malware. Cisco's Senderbase reports the average daily number of emails that contained malware was 36,200,000 during the month of October, 2014. Another 235 billion spam messages are sent each day, and many of those contain links to malicious sites as well (Spam Overview, 2014). Symantec's report on Cryptolocker (a very successful piece of Ransomware) also suggests that users are typically infected through email (Symantec Security Response, 2013).

Being able to respond quickly to an email threat might prevent additional infections, but how quickly must the security team react to be effective? The author's own research, taken from three separate phishes against 400 users, indicates that the average time it takes a user to open an email and either click a link or open an attachment, after it has been received, is a whopping 2,271 minutes (over 37 hours). In the sample, the median was 54 minutes. This indicates that although many users do open mail quickly, there is still a significant opportunity for an InfoSec team to respond to an email-borne threat, and perhaps mitigate that threat before the user even sees the message.

Blocking outbound connections to C2 servers and responding to malicious emails in a reasonable amount of time can drastically reduce the impact a malware infection can have on an organization. This is great news for organizations that cannot employ the use of sophisticated and expensive tools to prevent, detect, analyze and respond to malware threats for one reason or another. What follows, is a guide to implementing these capabilities at very little cost to the organization.

The first step is to remove the malicious emails from users' inboxes. This is done with a PowerShell script that searches through Exchange tracking logs and then quarantines messages to a central mailbox for review. Next, we will begin looking into

the behavior of the malware using a Windows virtual machine. Specifically, we will inspect network traffic to identify the destination IP addresses and DNS names with whom the malware is trying to communicate. Once we know where the malware is going, we will create DNS zones with an internal IP address so we can redirect the malware to our own server and intercept outbound connections from infected machines. This will enable us to track down any other machines that may have also become infected and prevent additional infections if a second email makes it into the environment with the same links. Finally, we will discuss some other uses of this environment and next steps for more thorough analysis and response. The idea for this paper came from original thought and experience defending small business networks. The individual concepts of analyzing malware in a sandbox and spoofing DNS responses for malware domains are certainly not unique. This paper combines these ideas with a website to intercept client connections and the ability to search for and quarantine email messages to create a complete solution.

2. Searching Mailboxes and Quarantining Malware

2.1. Get the OK first!

Analyzing a potential malware sample with Wireshark and static ARP entries, as discussed below, will provide information that can be used to prevent further infection within an organization, but analysis takes time. While analysis is being performed, other machines could be getting infected. Since many initial infections come through either links or attachments in email, a method to prevent users from opening the malicious email would be ideal. Many organizations with low or no budget rely on simply sending an email that asks people not to open the malicious email once it has been identified. This is done, ironically, by either sending an email to the entire organization, or by manually reviewing SMTP tracking logs to find out who received the messages in order to send a targeted email to only those that received it. There are a few issues with this type of response. Sending an email to the entire company can generate a flood of responses that takes time away from analysis work. Alternatively, manually reviewing logs to selectively choose recipients also takes time. Additionally, sending an email will not actually "prevent" users from opening the email anyway. The proposed solution is a

PowerShell script that will search through Exchange tracking logs to find the recipients of the suspect email and then actually export the message from the user's mailbox to a quarantine mailbox. There, it can be reviewed or retrieved in case of an error.

In many organizations, IT is prohibited from searching or extracting messages from user's mailboxes without authorization from Human Resources. Doing so without permission might get someone fired and/or might have legal consequences. This process should be presented to IT leaders, HR and the legal team if the organization has one prior to implementation. In almost every case, having technical permission to do something does not imply authorization from the business to do so. It is imperative that all stakeholders understand and approve the processes to be implemented. In fact, stop here and go get permission before even beginning to play with the script discussed below.

2.2. The PowerShell Script

The script in Appendix A starts by searching tracking logs for specific text in the subject in a specified number of previous days to generate a list of potential recipients. It then searches the inbox of the identified users and exports matching messages to a mailbox named 'malware'. Finally, it deletes the messages from the users' mailboxes so they cannot mistakenly or intentionally open the message. The export uses both the subject text and either an attachment type or specific text in the body as criteria to ensure a high level of accuracy. The script may be run on any workstation with the Exchange Management Shell, however it has been shown to execute faster when run directly on an Exchange Server with the CAS/HUB role rather than from a workstation with the tools. Begin by creating a mailbox into which messages can be placed that will be removed from users' mailboxes. To avoid modifying the script, name the mailbox "malware". After creating the destination mailbox, grant full access rights to the mailboxes for the user account that will be running the script. This can be accomplished with a one-liner in PowerShell such as:

```
get-mailbox | add-mailboxpermission -AccessRights FullAccess -
    user "myuser"
```

Replace "myuser" with the user account that will be running the script.

The script uses Windows Forms to create a GUI and requires Windows Management Framework 4.0 to run. It will not run in PowerShell 2.0 or 3.0. When executed, the user must enter (at a minimum) the number of days of email that should be included in the search along with the subject. The body and attachment fields are optional. Once the information is entered, click the Search button shown in Figure 1 below. The script will run a query against message tracking logs in Exchange and will return a list of unique email addresses that received a message matching the specified criteria. It is recommended that this list be verified manually by searching the message tracking logs for the messages in question. If satisfied with the list of recipients, click the Quarantine button to move the messages out of the user's mailboxes and into the "malware" mailbox.

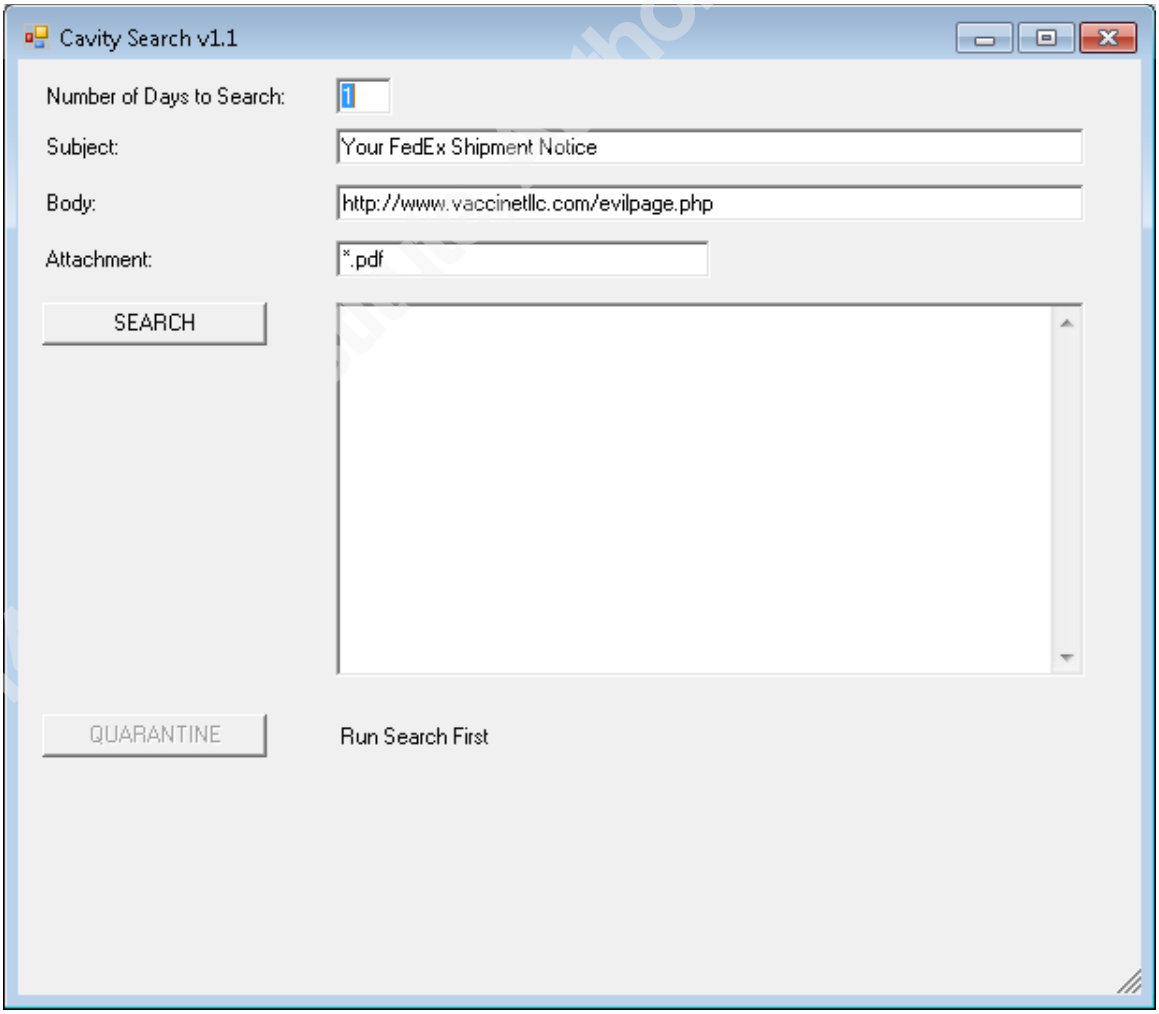

Figure 1: PowerShell Script GUI

Paul Ackerman, PAckerman@VacciNetLLC.com

Be aware, the messages will be deleted from the user's mailboxes. The script only searches the Inbox, but could be modified to search the entire mailbox. This setting was chosen for performance since the assumption is that the message was received recently and thus, is not likely to have been moved into another folder. The script creates a folder for each mailbox that should contain a matching message. However, if the message cannot be found because it was deleted or for any other reason, the destination folder in the malware mailbox will simply be empty. While the script is running, the malware mailbox will appear to contain many other messages from the users' mailboxes. Do not be alarmed. This is due to the way the export-mailbox cmdlet operates. It first copies the entire user's mailbox to the destination, then deletes any messages not matching the criteria specified in the command. When exporting a single message, this seems like the worst possible way to accomplish the task from a performance perspective, but it works. Further research into a different way of exporting a message (other than using export-mailbox), could drastically improve the performance of this script.

The search for specific text in the subject, body and attachment fields uses a -like match and supports wildcard matches. For example, to match any filename that ends in .pdf the user can enter *.pdf (as shown in Figure 1 above). If the subject does not narrow down the results sufficiently, add specific text that appears in the body as well. Once the emails have been quarantined and the immediate threat mitigated, we can begin the analysis of the malware to prevent further infection.

3. The VM Setup

3.1. VM Configuration

Several virtualization technologies could be used to create a virtual machine for analyzing malware. In this paper, we will use VMware Workstation though one could use VirtualBox, Parallels or something else. There are two major issues with using a virtual machine to analyze malware that we must discuss. The first issue is the fact that some malware detects that it is running in a VM and can alter or stop its execution making analysis more difficult. In fact, one in five samples will stop execution after detecting they are running in a VM (Candid Wueest, 2014). The second issue is that it is not impossible to break out of a VM and affect the host or other virtual machines running on

the host. The CVE-2014-0983 vulnerability can be exploited to escape Virtual Box (Florian Ledoux, 2014). Cloudburst was presented at BlackHat in 2009, which enables reliable code execution on the host from the guest in VMware ESX 4.0.0 and Workstation 6.5.X (Immunity, Inc., 2009). Even XEN Server has been shown to have vulnerabilities that allow code execution in the hypervisor context from within a guest (Rafal Wojtczuk, 2011).

This means that executing an unknown malware sample on a VM is actually putting the host and other hosts on the network at risk. While these two facts cannot and should not be ignored, there is still good reason to analyze malware within a VM. The author believes that malware authors will eventually stop altering execution based on whether it is running in a virtual machine since so many systems are becoming virtualized. Alternatively, the same techniques used here could be used on a physical PC that could be re-imaged using a whole disk imaging application like Ghost, ActiveBoot Disk or even a hardware duplicator like an ImageMasster 4000 or Kanguru KClone disk duplicator. A physical PC would be a better choice for a production environment to protect against VM escape-capable malware. In either case, it is also necessary to isolate the host itself by not connecting it to the network at all. If network access is desired in order to analyze additional downloads, for example, be sure to isolate the host on the network using a network-based firewall. This will help ensure that the malware sample does not infect the rest of the network. Properly segmenting the analysis workstation from the corporate network is beyond the scope of this paper. However, we will focus on ensuring the VM configuration is sufficient for any malware that is not designed to escape the virtual environment.

In VMware Workstation, one can isolate the network connection for a guest by creating a LAN Segment. After creating a Windows VM, edit the VM's settings and click on the network adapter. Click on the button named LAN Segments on the right side of the window, and then click Add to create a new LAN segment. Naming the segment "Infected" can help avoid using it unintentionally for another purpose in the future. After creating the LAN segment, be sure to select the radio button next to LAN segment and then choose the name that was just entered from the drop down box. One can also click Advanced here and change the MAC address of the VM to something else. While this

usually is not enough to fool malware into thinking this is a real PC, it does not hurt. One only needs to change the first half of the MAC address since that is what signifies this is a VM. Use 00:03:47 for Intel or 00:05:B5 for Broadcom or something else entirely (Vendor/Ethernet/Bluetooth MAC Address Lookup and Search, 2013). Now that the VM is sufficiently isolated on the network, we can begin configuring the guest itself.

3.2. Wireshark

Wireshark is a graphical network sniffer, formerly known as Ethereal, which runs on Windows. It will be used on the analysis VM to provide visibility into outbound connection attempts and DNS queries the malware makes. The tricky part is capturing traffic, while the VM is isolated on the network. There are a couple choices here. First, one could create additional VMs to simulate a virtual gateway and DNS server all on an isolated virtual network, but why waste the hard drive space? This becomes more costly if using real PCs for analysis instead of VMs since one would need additional hardware for the DNS server/router. In order for the analysis VM to send traffic over the network it only has to have a valid MAC address for its configured DNS server and default gateway. Instead of creating additional machines, we will simply create a static ARP entry on the analysis workstation with fake MAC addresses that will enable Wireshark to capture the malware's network traffic. Wireshark can be downloaded from www.wireshark.org. There are a couple of useful filters that will be used in Wireshark to pinpoint the interesting traffic. First, to look for DNS traffic, just type “dns” (lowercase) into the Filter. In the info column on the right, review the packets containing Standard DNS queries to see what domains the malware is attempting to reach. There will not be any responses, of course since there is not a real DNS server. To look for direct outbound connections to servers on the Internet, one can look for traffic destined for the fake MAC address of the default gateway. Fortunately, this is also a simple Wireshark filter. Enter eth.dst==aa:bb:cc:dd:ee:ff, where aa:bb:cc:dd:ee:ff is the MAC address we will choose for the gateway in the next section. Note, there are two equals signs in the filter.

3.3. Static ARP

When the malware tries to connect to a C2 server, the underlying operating system performs a DNS query for the server's name. In order to send the query over an

Ethernet network, the system must know the MAC address of the DNS server or the default gateway. To determine which, the operating system compares the IP and subnet mask of the host with the IP address configured for the DNS server to determine if the DNS server is in the same subnet (as in this case). If the DNS server is in the same subnet, the operating system will check its ARP cache to see if it has a MAC address associated with the DNS server's IP. If it does, it has all the information it needs and sends the frame out on the wire. If not, it sends an ARP broadcast instead. Since there is no real or virtual DNS server on the wire, there will never be a response to the broadcast. This is where static ARP comes in. By configuring a static ARP entry, a fake MAC address is mapped to the DNS server's IP. Now the workstation never needs to send the ARP broadcast and instead can simply fire off the query to the MAC address specified earlier and the traffic can be captured with Wireshark.

The workstation built for this demonstration was running Windows 7 Professional and was configured with a static IP address of 192.168.250.250, a DNS server of 192.168.250.53 and a gateway of 192.168.250.1. Static ARP entries were added using the following commands:

```
netsh interface ip add neighbors "Local Area Connection"
"192.168.250.53" "aa-bb-cc-11-22-33"

netsh interface ip add neighbors "Local Area Connection"
"192.168.250.1" "aa-bb-cc-dd-ee-ff"
```

Remember to take a snapshot of the analysis VM prior to copying any malware to it.

4. DNS for redirection

In the previous section, we used Wireshark to identify domain names and IP addresses to which the malware attempts to connect. Direct outbound connections to specific IP addresses can usually be blocked easily in the perimeter firewall. However, many small and medium sized firewalls lack the ability to block access to specific domain names or URLs. In many cases, an object must be created using an IP address and if the IP address for a domain changes, the rule is no longer effective. Some broadband router/firewall devices can block access by keywords but they are generally limited to only a few. A more scalable and still inexpensive method of blocking access to malware domains is to create authoritative zones for the malware domain in internal DNS

servers. This will ensure that any future emails or other sources of infection are not able to reach the domains because they will be unable to resolve the real, public IP addresses of malware domains. Some malware can actually change the DNS servers that are configured on the workstation. Thus, it is important to limit the granting of administrative privileges to users on workstations and to block outbound DNS queries to the Internet directly from hosts inside the network, thereby allowing only the DNS servers to reach the Internet on TCP/53 and UDP/53 in the perimeter firewall.

4.1. Scripting the creation of the zones

As with any task that needs to be performed quickly and with consistency, it can be helpful to create a script. The DNS script in Appendix B can be executed on a Microsoft DNS Server to quickly create a new zone. It is assumed that the organization has a Windows environment in which there are domain controllers acting as internal DNS servers that forward to non domain-joined Windows DNS servers in the DMZ that perform recursive queries. This is a well-documented best-practice design and is covered in SANS SEC505 Securing Windows as well as in the Domain Name System (DNS) Security Reference Architecture Version 1.0 from the Department of Homeland Security. (Department of Homeland Security, 2011)

To use the script, login to the DMZ DNS server that performs recursive queries and open a command prompt as an administrator. Execute the script using the following syntax:

```
InterceptDNS.bat [domainname] [hostname]
```

For example, to create a DNZ zone and host record for www.VacciNetLLC.com, enter:

```
InterceptDNS.bat VacciNetLLC.com www
```

Before using the script in production, modify the webserverIP and domainname to suit the environment. The script will clutter up the DNS server zones on the recursive servers but will help protect the organization from additional malware infection and has no real impact on administration or performance of those servers. The goal behind creating the DNS zone is two-fold. The first is to prevent the workstation from reaching the real malicious server. That can be done regardless of what information is supplied when the

zone is created. The second goal is to intercept the connection that was intended to go to the malware domain and perform some alerting. For this to happen, we must create the DNS zone such that they redirect traffic to an internal web server that can notify the team of the traffic. We call this website Interceptor.

5. Interceptor Website for interception and detection

5.1. Setting up the web server

Any web server can be used for this portion of the solution. We decided to choose IIS over Apache since it is one less application that must be downloaded and patched separately from the operating system. Setting up IIS is fairly straightforward and is described in detail here: (Microsoft, 2014). Once IIS is installed, create a new website with a dedicated IP address. If the box is not dedicated for this purpose, add an additional IP in the TCP/IP configuration and then use this additional IP for the website in IIS. Configure the authentication for the site for Windows authentication only to enable the site to capture the currently logged on user from workstations visiting the site. In order for Windows authentication to function properly, authenticated users must be granted read-only permissions on the files the site will use. The webpage we will create next will be called *interceptor.php*. The site must be configured to include this page in the default documents. In addition, the 404 error page should point to *interceptor.php* as well. This combination will ensure that requests to any page or virtual directory are redirected to the *interceptor.php* page. The site is written in PHP so it must be installed as well. Installing PHP for IIS is straightforward and is well-documented here: (PHP Manual - Installation on windows systems, 2014). Once PHP is installed, the php.ini file must be configured with a valid "from address" in order for PHP to be able to send mail. Find the section named [mail function] and configure the lines beginning with *SMTP* and *stmp_port* with the IP and port of the Exchange CAS/HUB server as shown below. Be sure to uncomment the line by removing the semicolon at the beginning of the line.

```
; For Win32 only.
; http://php.net/smtp
SMTP = 10.1.10.25
; http://php.net/smtp-port
smtp_port = 25
```

Next, configure a valid "from address" on the line beginning with *sendmail_from* as shown below:

```
; For Win32 only.
; http://php.net/sendmail-from
sendmail_from = InfoSec@lcec.pwr
```

5.2. The Interceptor site

The PHP code for the website is in Appendix C. The site uses server variables to capture information about the clients visiting the site. This information is valuable because it can be used to identify additional clients that may be infected even if they were infected by a different method than the one originally identified. When a client visits the site, the site sends an email with the IP address of the client, currently logged on user and the original URL to the InfoSec team. Other functions could be added using APIs for specific products in an environment such as initiating a virus scan or blocking all Internet access from the client's IP. The possibilities for containment and automatic remediation are nearly endless.

6. Conclusion

The combination of malware analysis, DNS spoofing, email quarantining and web traffic interception techniques provided in this discussion are designed to start the reader on a path towards a much more mature malware analysis and containment strategy. While far more advanced research certainly exists on the subjects of malware analysis and DNS spoofing, the work on the email quarantine script and the idea of using a website to identify additional infected clients are believed to be original in public literature. There will always be vulnerabilities in hardware and software and there will always be new exploits for both known and undisclosed vulnerabilities. Security vendors producing signatures for anti-virus and intrusion detection systems will always be behind the exploit and as the speed at which malware is distributed rises organizations will need to invest in a capability to detect and respond to these threats. Neglecting malware infections may result in the compromise of accounts that can be used to gain access to the network and

obtain any sort of data such as customer account information, trade secrets, employee or customer personally identifiable information, financial account numbers or anything else of value to an organization.

A single click and unchecked malware infection can lead to a full compromise of nearly any organization. Blocking Internet access will certainly help deal with an infection but the reader should not stop there. Being able to answer the questions: who did it, why and how is far more valuable. Some attempt to answer these should be made for every incident that occurs. To do this, a much more thorough analysis of the malware must be performed with more sophisticated tools. Noriben.py is one example of a free tool that can begin to provide more information about what an executable does without reverse engineering it. Noriben is a Python script freely available that can identify process, file and network activity that occurs during execution of a sample. There are many other free tools that can be used to dig deeper including Olly Debugger, Process Hacker and many others. It is left to the reader to learn more about these tools and to utilize them to increase the detection and response capabilities of their organization.

In addition, the tools and techniques discussed in this paper can be used for other purposes as well. While they were discussed specifically for analyzing malware, they could also be used to vet software downloads or to demonstrate the malicious behavior of a Trojan executable during a security awareness training, for example. The reader should consider additional uses for the environment that might be more specific to their organization. For additional information, the SANS reading room is full of articles on malware analysis, incident response and many other security-related topics.

Paul Ackerman, PAckerman@VacciNetLLC.com

7. References

Adware. (2014, 12 8). Retrieved from Kaspersky Lab: http://usa.kaspersky.com/internet-security-center/threats/adware#.VIZclhZtTY8

Candid Wueest. (2014, August 12). *Does malware still detect virutal machines?* Retrieved from Symantec Security Response: http://www.symantec.com/connect/blogs/does-malware-still-detect-virtual-machines

Department of Homeland Security. (2011, 06 21). *Domain Name System (DNS) Security Reference Architecture.* Retrieved from DHS: http://www.dhs.gov/sites/default/files/publications/dns_reference_architecture_0.pdf

Florian Ledoux. (2014, July 25). *Advanced Exploitation of VirtualBox 3D Acceleration VM Escape Vulnerability CVE-2014-0983*. Retrieved from VUPEN Vulnerability Research Team Blog: http://www.vupen.com/blog/20140725.Advanced_Exploitation_VirtualBox_VM_Escape.php

Glossary of Terms. (2014, 12 8). Retrieved from SANS.org: http://www.sans.org/security-resources/glossary-of-terms/

Immunity, Inc. (2009, June 2). *Cloudburst.* Retrieved from BlackHat: http://www.blackhat.com/presentations/bh-usa-09/KORTCHINSKY/BHUSA09-Kortchinsky-Cloudburst-PAPER.pdf

Microsoft. (2014, 12 8). *Installing IIS 7.5 on Windows 7 Professional, Enterprise, or Ultimate*. Retrieved from Microsoft: http://technet.microsoft.com/en-us/library/cc725762.aspx

Microsoft. (2014, 12 8). *What is Spyware*. Retrieved from Microsoft: http://www.microsoft.com/security/pc-security/spyware-whatis.aspx

MSFT-MMPC. (2012, January 24). *A different breed of downloader*. Retrieved from Microsoft TechNet Blogs:

Paul Ackerman, PAckerman@VacciNetLLC.com

http://blogs.technet.com/b/mmpc/archive/2012/01/24/a-different-breed-of-downloader.aspx

PHP Manual - Installation on windows systems. (2014, 12 8). Retrieved from PHP: http://php.net/manual/en/install.windows.iis7.php

Rafal Wojtczuk, J. R. (2011, April). *Following the White Rabbit: Software attacks against Intel VT-d technology.* Retrieved from invisibleThingsLab: http://www.invisiblethingslab.com/resources/2011/Software%20Attacks%20on%20Intel%20VT-d.pdf

Spam Overview. (2014, December 8). Retrieved from Senderbase: https://www.senderbase.org/static/spam/#tab=0

Symantec Security Response. (2013, December 6). *Cryptolocker Q&A: Menace of the Year*. Retrieved from Symantec Security Response: http://www.symantec.com/connect/blogs/cryptolocker-qa-menace-year

Vendor/Ethernet/Bluetooth MAC Address Lookup and Search. (2013, July 10). Retrieved from Coffer: http://www.coffer.com/mac_find/?string=broadcom

A. Appendix A - PowerShell Mailbox Script

```
#####################################################################
# CavitySearch_GUI.ps1
# Paul Ackerman
# Tom Hornby
# V 1.0      December 2014
#####################################################################

#Generated Form Function
function GenerateForm {
#####################################################################
# Base Windows Form Code Generated By: SAPIEN Technologies PrimalForms
(Community Edition) v1.0.8.0
# Generated On: 7/3/2011 11:35 AM
# Generated By: sean.kearney
#####################################################################

#region Import the Assemblies
[reflection.assembly]::loadwithpartialname("System.Windows.Forms") | Out-Null
[reflection.assembly]::loadwithpartialname("System.Drawing") | Out-Null
#endregion

#region Generated Form Objects
$MainForm = New-Object System.Windows.Forms.Form
$SearchButton = New-Object System.Windows.Forms.Button
$QuarantineButton = New-Object System.Windows.Forms.Button
$QuarantineLabel = New-Object System.Windows.Forms.Label
$DaysLabel = New-Object System.Windows.Forms.Label
$SubjectLabel = New-Object System.Windows.Forms.Label
$BodyLabel = New-Object System.Windows.Forms.Label
$AttachmentLabel = New-Object System.Windows.Forms.Label
$global:Days = New-Object System.Windows.Forms.TextBox
$global:Subject = New-Object System.Windows.Forms.TextBox
$Body = New-Object System.Windows.Forms.TextBox
$Attachment = New-Object System.Windows.Forms.TextBox
$Searchtxt = New-Object System.Windows.Forms.TextBox
$Results = New-Object System.Windows.Forms.TextBox
$InitialFormWindowState = New-Object System.Windows.Forms.FormWindowState
#endregion Generated Form Objects

#----------------------------------------------
#Generated Event Script Blocks
#----------------------------------------------
#Provide Custom Code for events specified in PrimalForms.
$handler_SearchButton_Click=
{
$Searchtxt.text = ""
$numDays = $Days.text
$numDays = -$numDays
$archive = "archivemgr_journal@lcec.pwr"

$start = (get-date).adddays($numDays).toshortdatestring()
$stop = (get-date).toshortdatestring()
$logfile = ".\mb2check.txt"
$mySubject = $Subject.text
$servers = get-clientaccessserver
foreach ($server in $servers) {
write-host $server
```

Paul Ackerman, PAckerman@VacciNetLLC.com

```
$log = get-messagetrackinglog -Server "$server" -EventID "DELIVER" -
MessageSubject $mySubject -Start "$start 00:00:00 AM" -End "$stop 11:59:59 PM"
-resultsize Unlimited
#write-host $log | where-object $log.Recipients -ne $archive | select
Recipients
       ForEach ($recipient in $log.Recipients) {
                            if ($recipient -ne "$archive") {
                                   $results += ($recipient += "`r`n").tolower()
                                   }
                     }
              }
$uniqueresults = $results | sort.exe | get-unique
$uniqueresults | out-file $logfile
foreach ($item in $uniqueresults) {
       if ($item -like "*@lcec.net") {
              $Searchtxt.AppendText($item + "`r`n")
              $count++
       }
}
if ($count -gt 0) {
       $QuarantineButton.Enabled = $True
       $QuarantineLabel.Text = "Ready"
}
else {
       $Searchtxt.AppendText("No Results")
       }
}

$handler_QuarantineButton_Click=
{
$numDays = $Days.text
$numDays = -$numDays
$start = (get-date).adddays($numDays).toshortdatestring()
$stop = (get-date).toshortdatestring()
$logfile = ".\mb2check.txt"
$mbbackup = "malware"
$mySubject = $Subject.text
$mailboxes = get-content "$logfile"
foreach ($mailbox in $mailboxes) {
       if ($mailbox.length -gt 2) {
              write-host $mailbox.Trim()
              $QuarantineLabel.Text = "Exporting mailbox $mailbox. Depending on
the size of the mailboxes, this could take several minutes."
              ########################################NEW
CODE###############################
              if (($Body.text.Length -lt 1) -and ($Attachment.text.Length -lt
1))    {
                     write-host "Command 1"

                     export-mailbox -identity $mailbox.Trim() -subjectkeywords
$mySubject -includefolders "\Inbox" -StartDate "$start 00:00:00 AM" -EndDate
"$stop 11:59:59 PM" -TargetMailbox $mbbackup -TargetFolder "Malware" -
confirm:$false -deletecontent
              }
              elseif ($Body.text.Length -lt 1){
                     write-host "Command 2"
                     export-mailbox -identity $mailbox.Trim() -subjectkeywords
$mySubject -includefolders "\Inbox" -AttachmentFilenames $Attachment.text -
StartDate "$start 00:00:00 AM" -EndDate "$stop 11:59:59 PM" -TargetMailbox
$mbbackup -TargetFolder "Malware" -confirm:$false -deletecontent
              }
              elseif ($attachment.text.Length -lt 1){
                     write-host "Command 3"
```

Paul Ackerman, PAckerman@VacciNetLLC.com

```
                    export-mailbox -identity $mailbox.Trim() -subjectkeywords
$mySubject -includefolders "\Inbox" -StartDate "$start 00:00:00 AM" -EndDate
"$stop 11:59:59 PM" -TargetMailbox $mbbackup -TargetFolder "Malware" -
contentkeywords $body.Text -confirm:$false -deletecontent
             }
             else{
                    write-host "Command 4"
                    export-mailbox -identity $mailbox.Trim() -subjectkeywords
$mySubject -includefolders "\Inbox" -AttachmentFilenames $attachment.text -
StartDate "$start" -EndDate "$stop" -TargetMailbox $mbbackup -TargetFolder
"Malware" -contentkeywords $body.Text -confirm:$false -deletecontent
             }
       }
       $QuarantineLabel.Text = "Complete"
       $QuarantineButton.Enabled = $False
}
}
$OnLoadForm_StateCorrection=
{#Correct the initial state of the form to prevent the .Net maximized form
issue
$MainForm.WindowState = $InitialFormWindowState
}

#----------------------------------------------
#region Generated Form Code
$MainForm.Text = "Cavity Search v1.1"
$MainForm.Name = "MainForm"
$MainForm.DataBindings.DefaultDataSourceUpdateMode = 0
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 600
$System_Drawing_Size.Height = 500
$MainForm.ClientSize = $System_Drawing_Size

###########################################################################
#                              Days Label                                 #
###########################################################################
$DaysLabel.Name = "DaysLabel"
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 150
$System_Drawing_Size.Height = 23
$DaysLabel.Size = $System_Drawing_Size
$DaysLabel.Text = "Number of Days to Search:"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 13
$System_Drawing_Point.Y = 13
$DaysLabel.Location = $System_Drawing_Point
$DaysLabel.DataBindings.DefaultDataSourceUpdateMode = 0
$MainForm.Controls.Add($DaysLabel)

###########################################################################
#                              Days TextBox                               #
###########################################################################
$Days.Name = "Days"
$Days.TabIndex = 0
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 30
$System_Drawing_Size.Height = 23
$Days.Size = $System_Drawing_Size
$Days.Text = "1"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 170
$System_Drawing_Point.Y = 10
$Days.Location = $System_Drawing_Point
```

Paul Ackerman, PAckerman@VacciNetLLC.com

```
$Days.DataBindings.DefaultDataSourceUpdateMode = 0
$Days.add_Click($handler_Days_Click)
$MainForm.Controls.Add($Days)

#############################################################################
#                                Subject Label                              #
#############################################################################
$SubjectLabel.Name = "SubjectLabel"
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 150
$System_Drawing_Size.Height = 23
$SubjectLabel.Size = $System_Drawing_Size
$SubjectLabel.Text = "Subject:"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 13
$System_Drawing_Point.Y = 40
$SubjectLabel.Location = $System_Drawing_Point
$SubjectLabel.DataBindings.DefaultDataSourceUpdateMode = 0
$MainForm.Controls.Add($SubjectLabel)

#############################################################################
#                                Subject TextBox                            #
#############################################################################
$Subject.Name = "Subject"
$Subject.TabIndex = 1
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 400
$System_Drawing_Size.Height = 23
$Subject.Size = $System_Drawing_Size
$Subject.Text = "Your FedEx Shipment Notice"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 170
$System_Drawing_Point.Y = 37
$Subject.Location = $System_Drawing_Point
$Subject.DataBindings.DefaultDataSourceUpdateMode = 0
$Subject.add_Click($handler_Subject_Click)
$MainForm.Controls.Add($Subject)

#############################################################################
#                                Body Label                                 #
#############################################################################
$BodyLabel.Name = "BodyLabel"
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 150
$System_Drawing_Size.Height = 23
$BodyLabel.Size = $System_Drawing_Size
$BodyLabel.Text = "Body:"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 13
$System_Drawing_Point.Y = 70
$BodyLabel.Location = $System_Drawing_Point
$BodyLabel.DataBindings.DefaultDataSourceUpdateMode = 0
$MainForm.Controls.Add($BodyLabel)

#############################################################################
#                                Body TextBox                               #
#############################################################################
$Body.Name = "Body"
$Body.TabIndex = 2
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 400
$System_Drawing_Size.Height = 23
$Body.Size = $System_Drawing_Size
```

```
$Body.Text = "http://www.vaccinetllc.com/evilpage.php"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 170
$System_Drawing_Point.Y = 67
$Body.Location = $System_Drawing_Point
$Body.DataBindings.DefaultDataSourceUpdateMode = 0
$Body.add_Click($handler_Body_Click)
$MainForm.Controls.Add($Body)

#########################################################################
#                              Attachment Label                         #
#########################################################################
$AttachmentLabel.Name = "AttachmentLabel"
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 150
$System_Drawing_Size.Height = 23
$AttachmentLabel.Size = $System_Drawing_Size
$AttachmentLabel.Text = "Attachment:"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 13
$System_Drawing_Point.Y = 100
$AttachmentLabel.Location = $System_Drawing_Point
$AttachmentLabel.DataBindings.DefaultDataSourceUpdateMode = 0
$MainForm.Controls.Add($AttachmentLabel)

#########################################################################
#                              Attachment TextBox                       #
#########################################################################
$Attachment.Name = "Attachment"
$Attachment.TabIndex = 3
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 200
$System_Drawing_Size.Height = 23
$Attachment.Size = $System_Drawing_Size
$Attachment.Text = "*.pdf"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 170
$System_Drawing_Point.Y = 97
$Attachment.Location = $System_Drawing_Point
$Attachment.DataBindings.DefaultDataSourceUpdateMode = 0
$Attachment.add_Click($handler_Attachment_Click)
$MainForm.Controls.Add($Attachment)


#########################################################################
#                              Search Button                            #
#########################################################################
$SearchButton.TabIndex = 4
$SearchButton.Name = "SearchButton"
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 120
$System_Drawing_Size.Height = 23
$SearchButton.Size = $System_Drawing_Size
$SearchButton.UseVisualStyleBackColor = $True
$SearchButton.Text = "SEARCH"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 13
$System_Drawing_Point.Y = 130
$SearchButton.Location = $System_Drawing_Point
$SearchButton.DataBindings.DefaultDataSourceUpdateMode = 0
$SearchButton.add_Click($handler_SearchButton_Click)
$MainForm.Controls.Add($SearchButton)
```

Paul Ackerman, PAckerman@VacciNetLLC.com

```
#########################################################################
#                              Search TextBox                           #
#########################################################################
$Searchtxt.Name = "Searchtxt"
$Searchtxt.TabIndex = 3
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 400
$System_Drawing_Size.Height = 200
$Searchtxt.MultiLine = $True
$Searchtxt.ScrollBars = "Vertical"
$Searchtxt.Size = $System_Drawing_Size
$Searchtxt.Text = ""
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 170
$System_Drawing_Point.Y = 130
$Searchtxt.Location = $System_Drawing_Point
$Searchtxt.DataBindings.DefaultDataSourceUpdateMode = 0
$MainForm.Controls.Add($Searchtxt)

#########################################################################
#                               Quarantine Button                       #
#########################################################################
$QuarantineButton.TabIndex = 5
$QuarantineButton.Name = "QuarantineButton"
$QuarantineButton.Enabled = $False
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 120
$System_Drawing_Size.Height = 23
$QuarantineButton.Size = $System_Drawing_Size
$QuarantineButton.UseVisualStyleBackColor = $True
$QuarantineButton.Text = "QUARANTINE"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 13
$System_Drawing_Point.Y = 350
$QuarantineButton.Location = $System_Drawing_Point
$QuarantineButton.DataBindings.DefaultDataSourceUpdateMode = 0
$QuarantineButton.add_Click($handler_QuarantineButton_Click)
$MainForm.Controls.Add($QuarantineButton)

#########################################################################
#                             Quarantine Label                          #
#########################################################################
$QuarantineLabel.Name = "QuarantineLabel"
$System_Drawing_Size = New-Object System.Drawing.Size
$System_Drawing_Size.Width = 400
$System_Drawing_Size.Height = 70
$QuarantineLabel.Size = $System_Drawing_Size
$QuarantineLabel.Text = "Run Search First"
$System_Drawing_Point = New-Object System.Drawing.Point
$System_Drawing_Point.X = 170
$System_Drawing_Point.Y = 355
$QuarantineLabel.Location = $System_Drawing_Point
$QuarantineLabel.DataBindings.DefaultDataSourceUpdateMode = 0
$MainForm.Controls.Add($QuarantineLabel)

#endregion Generated Form Code

#Save the initial state of the form
$InitialFormWindowState = $MainForm.WindowState
#Init the OnLoad event to correct the initial state of the form
$MainForm.add_Load($OnLoadForm_StateCorrection)
#Show the Form
$MainForm.ShowDialog()| Out-Null
```

Paul Ackerman, PAckerman@VacciNetLLC.com

```
} #End Function

#Call the Function
GenerateForm
```

B. Appendix B - DNS Batch File

```
@echo off

if "%1"=="" (
  echo.
  echo Syntax: InterceptDNS.bat [domainname] [hostrecord]
  echo.
  echo For Example:
  echo InterceptDNS.bat VacciNetLLC.com www
  echo will create a DNS zone for VacciNetLLC.com and then will create a
  echo cname record for www pointing to interceptor.yourdomain.local
  goto end
)

set zonename=%1
set host=%2

REM Configure the next two variables with your organziation's specific
information
set webserverIP=10.1.1.100
set domainname=mydomain.local

REM Create new DNZ Zone
dnscmd 127.0.0.1 /ZoneAdd %zonename% /primary
echo Zone Created.

echo Creating the cname record...
dnscmd 127.0.0.1 /RecordAdd %zonename% %host% CName Interceptor.%domainname%
dnscmd 127.0.0.1 /RecordAdd %zonename% Zone-Created-by-InfoSec-Interceptor
CName Interceptor.%domainname%
dnscmd 127.0.0.1 /RecordAdd %zonename% @ A %webserverIP%

:end
```

C. Appendix C - PHP Website Code

```
<?php
if ($_SERVER['REQUEST_URI']!=favicon.ico)
{
  $ip = $_SERVER['REMOTE_ADDR'];
  $user = $_SERVER['LOGON_USER'];
  $site = 'http' . (isset($_SERVER['HTTPS']) ? 's' : '') . '://' .
$_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'];
  $message = "A request to " . $site . " has been intercepted from machine " .
$ip . " from user " . $user;
  echo "Your PC has attempted to visit a website that the Information Security
Team has deemed malicious. The team has been made aware of the situation and
will be in touch if necessary.";

  $to = 'InfoSec@yourdomain.local';
```

Paul Ackerman, PAckerman@VacciNetLLC.com

```
  $subject = 'Interceptor Alert';
  $email = 'InfoSec@ yourdomain.local ';
  $header  = 'MIME-Version: 1.0' . "\r\n";
  $header .= 'Content-type: text/html; charset=UTF-8' . "\r\n";
  $header .= "From: $email\r\nReply-To: $email" . "\r\n";
  mail($to, $subject, $message, $header);
}

?>
```

D. Appendix D - Website Configuration

Create a new site in IIS called “Interceptor” and configure the secondary IP address of the server in the site bindings as shown in Figure D.1.

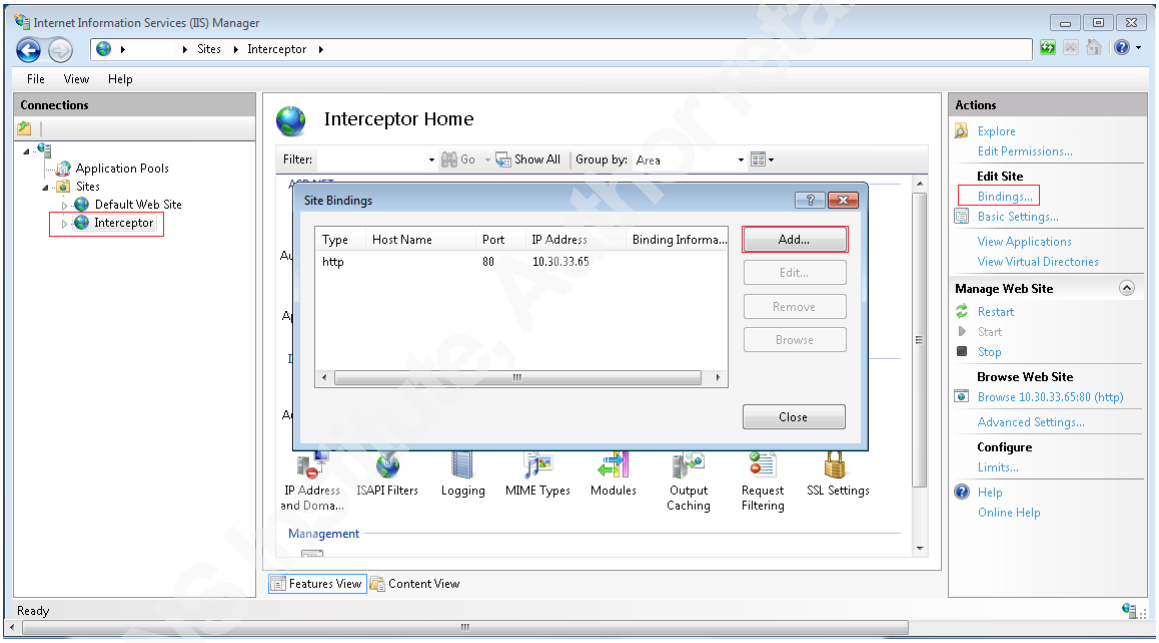

Figure D.1: Site Bindings

Next, enable Windows Authentication and disable all other methods as show in Figure D.2 in order to capture the visitor’s currently logged in username. If Windows Authentication is not available, you will need to add it using the Windows Add/Remove Features wizard.

Paul Ackerman, PAckerman@VacciNetLLC.com

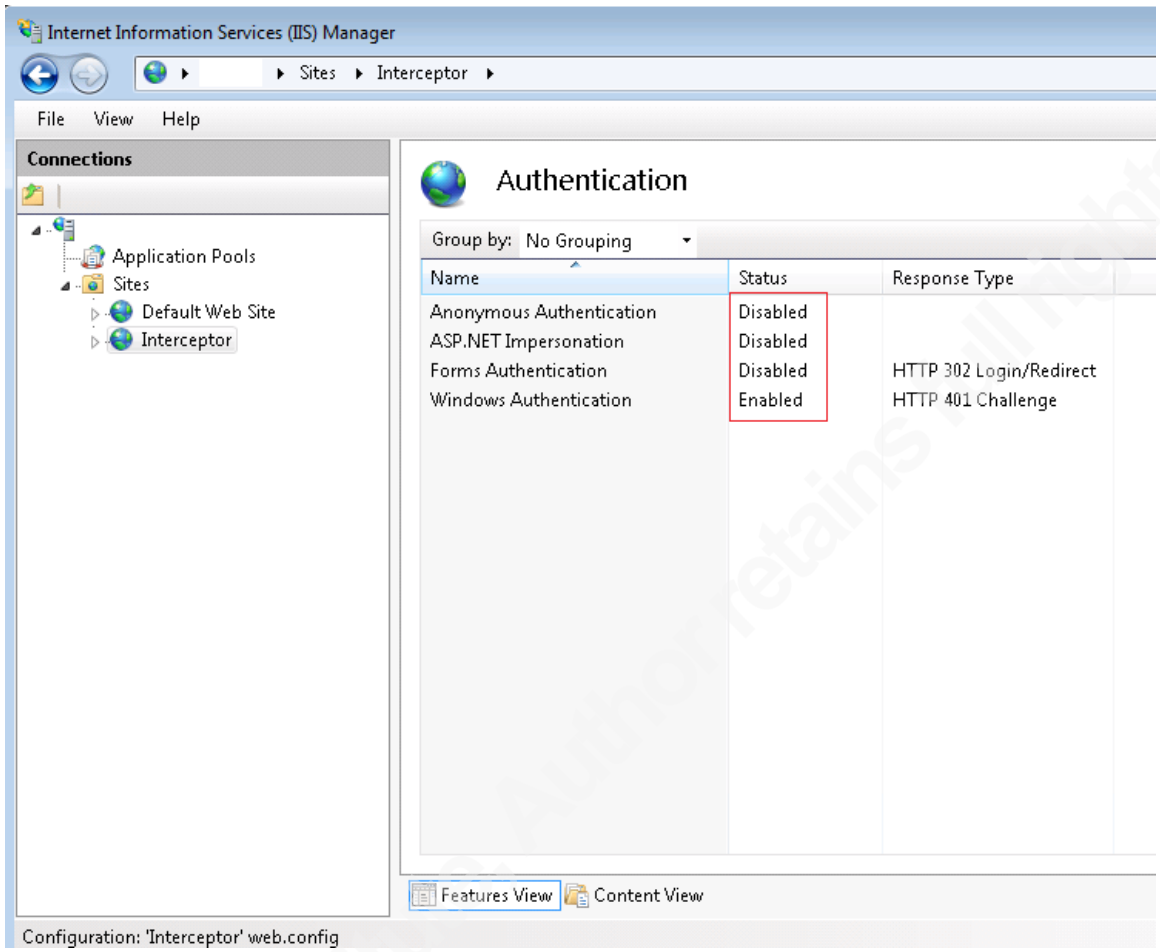

Figure D.2: Authentication

Configure the default document and the 'page not found' 404 error to redirect users to the interceptor page so they will land on the PHP file regardless of what page was requested as shown in Figures D.3 and D.4

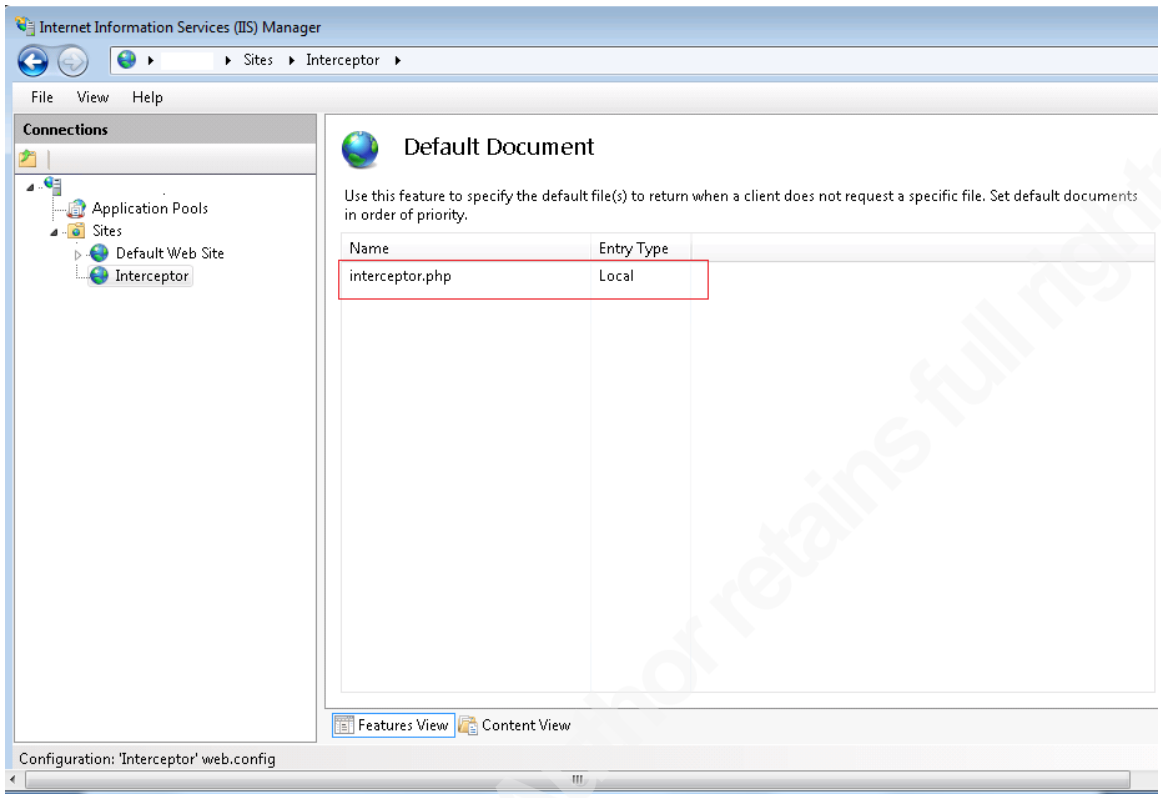

Figure D.3: Default Document

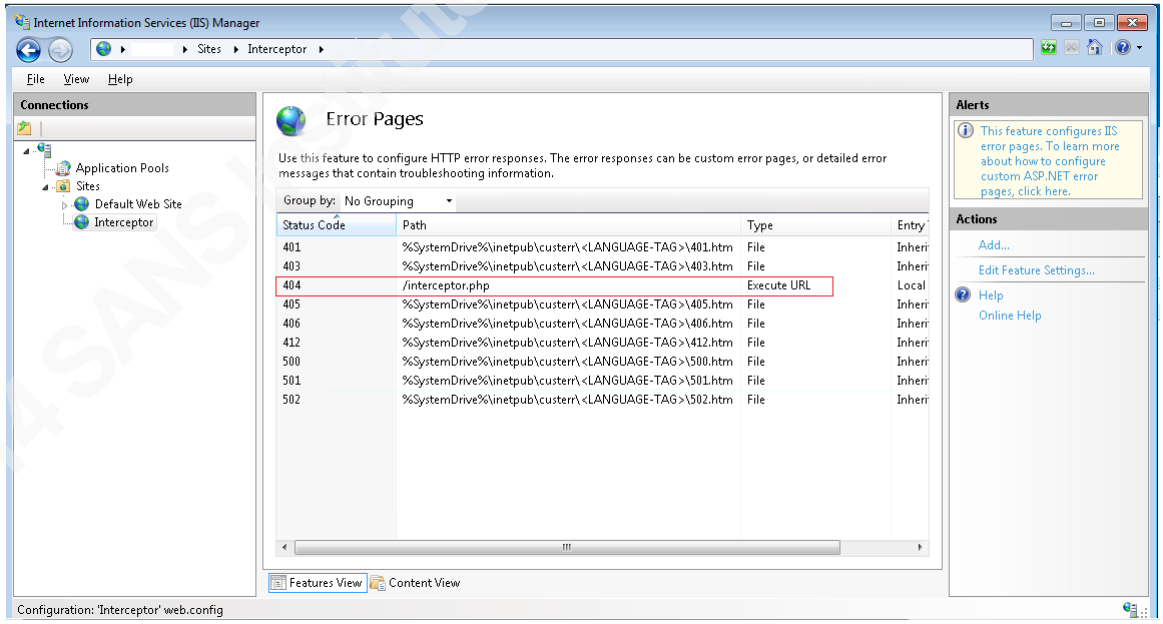

Figure D.4: Error Pages

Last Updated: May 28th, 2018

Upcoming Training

SANS Atlanta 2018	Atlanta, GA	May 29, 2018 - Jun 03, 2018	Live Event
SANS Rocky Mountain 2018	Denver, CO	Jun 04, 2018 - Jun 09, 2018	Live Event
Community SANS Bethesda SEC401 @ USO - Academy	Bethesda, MD	Jun 04, 2018 - Jun 09, 2018	Community SANS
SANS London June 2018	London, United Kingdom	Jun 04, 2018 - Jun 12, 2018	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 18, 2018 - Jun 23, 2018	Community SANS
SANS Cyber Defence Japan 2018	Tokyo, Japan	Jun 18, 2018 - Jun 30, 2018	Live Event
SANS Oslo June 2018	Oslo, Norway	Jun 18, 2018 - Jun 23, 2018	Live Event
Community SANS Madison SEC401	Madison, WI	Jun 18, 2018 - Jun 23, 2018	Community SANS
SANS Crystal City 2018	Arlington, VA	Jun 18, 2018 - Jun 23, 2018	Live Event
SANS Minneapolis 2018	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	Live Event
Minneapolis 2018 - SEC401: Security Essentials Bootcamp Style	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	vLive
Community SANS Nashville SEC401	Nashville, TN	Jun 25, 2018 - Jun 30, 2018	Community SANS
SANS Cyber Defence Canberra 2018	Canberra, Australia	Jun 25, 2018 - Jul 07, 2018	Live Event
SANS Vancouver 2018	Vancouver, BC	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANS Cyber Defence Singapore 2018	Singapore, Singapore	Jul 09, 2018 - Jul 14, 2018	Live Event
SANS Charlotte 2018	Charlotte, NC	Jul 09, 2018 - Jul 14, 2018	Live Event
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
SANSFIRE 2018 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
SANS Malaysia 2018	Kuala Lumpur, Malaysia	Jul 16, 2018 - Jul 21, 2018	Live Event
Mentor Session - SEC401	Jacksonville, FL	Jul 17, 2018 - Aug 28, 2018	Mentor
Community SANS Bethesda SEC401	Bethesda, MD	Jul 23, 2018 - Jul 28, 2018	Community SANS
SANS Riyadh July 2018	Riyadh, Saudi Arabia	Jul 28, 2018 - Aug 02, 2018	Live Event
SANS Pittsburgh 2018	Pittsburgh, PA	Jul 30, 2018 - Aug 04, 2018	Live Event
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
San Antonio 2018 - SEC401: Security Essentials Bootcamp Style	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	vLive
SANS Boston Summer 2018	Boston, MA	Aug 06, 2018 - Aug 11, 2018	Live Event
SANS August Sydney 2018	Sydney, Australia	Aug 06, 2018 - Aug 25, 2018	Live Event
SANS Hyderabad 2018	Hyderabad, India	Aug 06, 2018 - Aug 11, 2018	Live Event
Northern Virginia- Alexandria 2018 - SEC401: Security Essentials Bootcamp Style	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	vLive
SANS Northern Virginia- Alexandria 2018	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	Live Event