

## Letters to Santa--a real world attack

### Part 4, the Attack Begins

In Part 3 you found that Alabaster built the [dev site](#) using Apache Struts. In Part 1 you found that there were two major Apache Struts vulnerabilities in 2017, CVE-2017-5638 in March and CVE-2017-9805 in September. In Hint 6 Sparkle said that Alabaster had patched CVE-2017-5638, which was used in the “Equal-Facts” attack. She does not mention CVE-2017-9805.

In Hint 7, Sparkle gives a link to a [SANS Penetration Test Blog article](#) by Chris Davis. The article explains that a good penetration tester can modify exploit code, since the code may not meet their needs. Chris uses CVE-2017-9805 as an example and gives us a link to his final product, a Python script, on GitHub. We won’t have to modify any code this time. Since Chris has done the heavy lifting, we can simply use his script in script kiddy fashion.

### Some Notes about the Exploit

<sidebar>

A “shell”, or command shell, is similar to a Linux terminal or Windows command prompt. Penetration testers distinguish between shells and terminals, however. A shell gives the ability to run commands on a host and see some results but is not as powerful as a terminal or SSH session. Control characters, commands that require user input like usernames and passwords, and other things may not work in a shell or may cause the shell to crash.

</sidebar>

The Python script Chris wrote to exploit CVE-2017-9805 is not a shell. It executes commands on the target but does nothing to show the target’s response. You are essentially blind. You send the command off to the target and have no idea if it worked, unless you provide for the response yourself.

### The Best Way

The best way to attack would be to build a mock-up of the target in a lab environment, and then test the commands we wish to use in the lab. Through reconnaissance, we know that the site uses Apache Struts. Using Nmap scans, we could determine the server operating system. We would build virtual machines to match what we know about the target and then practice the attack. This method is more work but increases the likelihood of our success and reduces the noise we make on the network (and reduces frustration, too.)

### Our Approach

This is a learning environment, so we will learn by doing but we will proceed slowly. The first thing we need to know is whether the exploit works or not. We need the simplest command we can find that will create some network traffic that we can detect. Not every web site will allow its servers to ping the outside world but maybe Alabaster has not been diligent, and we will get lucky. Also, this gives us the opportunity for a brief networking review.

<sidebar>

Recall from our discussions about networking last semester that most ISPs use [Network Address Translation \(NAT\)](#). Using NAT, the ISP, our wireless router, or whatever, gives each of our computers a separate private IP address, and sends the traffic for all computers to the Internet using a single public IP

address. It separates the traffic by adjusting port numbers and making a table to keep track of what traffic should go where. This creates a problem for us, as the NAT table is only updated when we initiate an outgoing connection. When the target tries to initiate an incoming connection, there won't be an entry in the table and the connection attempt will be dropped.

</sidebar>

There are several ways to work around the NAT problem, but the easiest is to use our Amazon AWS instance. It has a public IP address that the target can reach, assuming we configure the inbound rules for the instance properly. Instructions for creating a free AWS instance are available [here](#).

### Configure an Inbound Rule on the AWS Instance

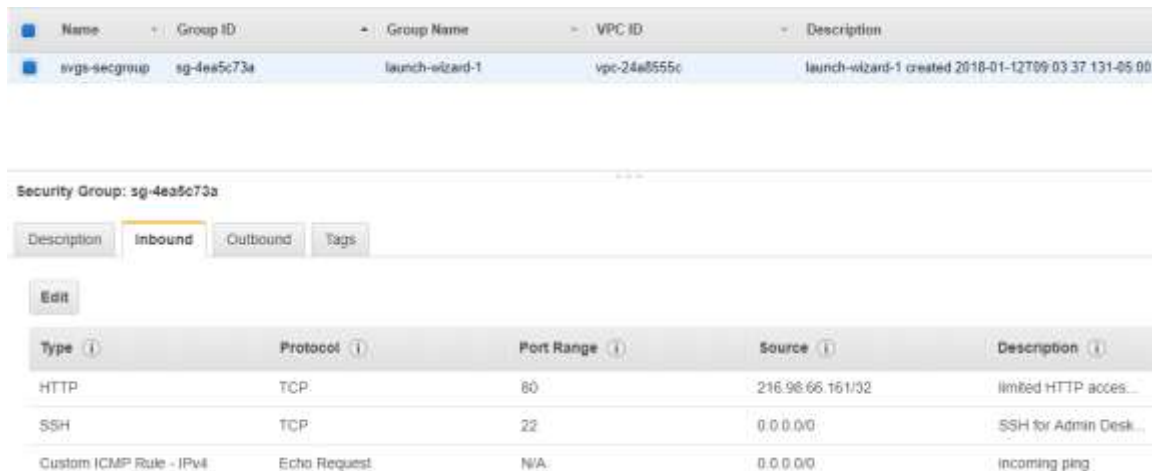
Add a rule to the Inbound tab of the Security Group to allow inbound ping traffic. Be sure to set it to ICMP IPv4 (not IPv6) and to Echo Request (not Echo Reply.)



Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	Custom 216.96.66.161/32	limited HTTP access
SSH	TCP	22	Custom 0.0.0.0	SSH for Admin Desktop
Custom ICMP	Echo Request	N/A	Anywhere 0.0.0.0::0	incoming ping

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

The result should look like this.



Name	Group ID	Group Name	VPC ID	Description
evgs-secgroup	sg-4ea5c73a	launch-wizard-1	vpc-24a8555c	launch-wizard-1 created 2018-01-12T09:03:37.131-05:00

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	216.96.66.161/32	limited HTTP acces...
SSH	TCP	22	0.0.0.0	SSH for Admin Desk...
Custom ICMP Rule - IPv4	Echo Request	N/A	0.0.0.0	incoming ping

Verify the rule by pinging your instance from your workstation.

```
C:\Users\John>ping ec2-54-159-104-104.compute-1.amazonaws.com

Pinging ec2-54-159-104-104.compute-1.amazonaws.com [54.159.104.104] with 32 bytes of data:
Reply from 54.159.104.104: bytes=32 time=45ms TTL=241
Reply from 54.159.104.104: bytes=32 time=45ms TTL=241
Reply from 54.159.104.104: bytes=32 time=45ms TTL=241

Ping statistics for 54.159.104.104:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 45ms, Maximum = 45ms, Average = 45ms
Control-C
^C
C:\Users\John>
```

## Monitoring Traffic on the AWS instance

There are many ways to monitor traffic from the target to our AWS instance. In this challenge we will use packet capture (sniffing) and later a netcat listener. When we studied networking we mostly used Wireshark to monitor traffic. The AWS instance does not have a GUI, so we will use the command line tool tcpdump instead.

Test to see if tcpdump installed on your instance by running the command, `tcpdump --help`. If you find that it is not installed, install it using `sudo yum install tcpdump` (Amazon Linux AMI and RedHat based images) or `sudo apt-get install tcpdump` (Debian or Ubuntu based images.)

We will use the syntax, `sudo tcpdump -nn -i <interface> '<filter>'`. The option `-nn` tells tcpdump not to convert IP addresses to DNS names, and not to convert port numbers to protocol names. The `-i` option tells tcpdump which network interface to listen to. Use the command, `sudo tcpdump -D`, to determine the available interfaces.

```
[ec2-user@ip-172-31-37-34 ~]$ sudo tcpdump -D
1.eth0
2.nflog (Linux netfilter log (NFLOG) interface)
3.nfqueue (Linux netfilter queue (NFQUEUE) interface)
4.any (Pseudo-device that captures on all interfaces)
5.lo
[ec2-user@ip-172-31-37-34 ~]$
```

In this case, we will use `eth0` as our interface. You could also determine the available interfaces with the commands `ipconfig -a` (deprecated or removed in some Linux distributions,) or the command `ip` (`ip link` or `ip address`.)

You are connecting to the AWS instance with SSH, so if you do not use a filter you will see all the traffic between your workstation and your instance. We need to filter that out. One way would be to get rid of port 22 (SSH) traffic.

```
sudo tcpdump -i eth0 'not port 22'
```

A better way would be to use a filter that just shows what we are looking for. The ping command uses ICMP echo requests and responses. Note: Word uses “smart quotes” so this may not work if you cut and paste the commands. You may have to change to single quotes.

```
sudo tcpdump -i eth0 'icmp'
```

Here is a ping from a workstation.

```
C:\Users\John>ping ec2-54-159-104-104.compute-1.amazonaws.com

Pinging ec2-54-159-104-104.compute-1.amazonaws.com [54.159.104.104] with 32 bytes of data:
Reply from 54.159.104.104: bytes=32 time=45ms TTL=241
Reply from 54.159.104.104: bytes=32 time=45ms TTL=241
Reply from 54.159.104.104: bytes=32 time=46ms TTL=241
Reply from 54.159.104.104: bytes=32 time=45ms TTL=241

Ping statistics for 54.159.104.104:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 45ms, Maximum = 46ms, Average = 45ms

C:\Users\John>
```

This is how it should look on the AWS instance.

```
[ec2-user@ip-172-31-37-34 ~]$ sudo tcpdump -nni eth0 'icmp'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
01:39:56.467057 IP 64.203.133.36 > 172.31.37.34: ICMP echo request, id 1, seq 11, length 40
01:39:56.467087 IP 172.31.37.34 > 64.203.133.36: ICMP echo reply, id 1, seq 11, length 40
01:39:57.470478 IP 64.203.133.36 > 172.31.37.34: ICMP echo request, id 1, seq 12, length 40
01:39:57.470511 IP 172.31.37.34 > 64.203.133.36: ICMP echo reply, id 1, seq 12, length 40
01:39:58.477055 IP 64.203.133.36 > 172.31.37.34: ICMP echo request, id 1, seq 13, length 40
01:39:58.477078 IP 172.31.37.34 > 64.203.133.36: ICMP echo reply, id 1, seq 13, length 40
01:39:59.484087 IP 64.203.133.36 > 172.31.37.34: ICMP echo request, id 1, seq 14, length 40
01:39:59.484112 IP 172.31.37.34 > 64.203.133.36: ICMP echo reply, id 1, seq 14, length 40
^C
8 packets captured
8 packets received by filter
0 packets dropped by kernel
[ec2-user@ip-172-31-37-34 ~]$
```

Test it on your instance to make sure it is working before we begin the attack.

### Begin the Attack (Finally!)

We will use the Apache Struts exploit code from the GitHub link at the end of the [blog article](#). Copy the Python code from Chris Davis' link to your workstation. If you use a Windows workstation you will have to install Python. If you use Linux or a Linux VM, python is already installed.

Note: The article talks about encoding the command with base64 and then decoding the command on the target server with a line like this.

```
echo L3Vzci9iaW4vd2hvYW1pIA== | base64 -d | tee -a /tmp/payload.tmp ; /bin/bash /tmp/payload.tmp; /bin/rm /tmp/payload.tmp
```

The exploit code in the Python script does this for you (see line 37) so your command does not have to include it.

Run the exploit code on your workstation. The URL should be the complete URL to a page on the dev site. The command should be something like this.

```
ping -c 4 ec2-54-84-224-72.compute-1.amazonaws.com
```

It is important that you include the `-c 4` option. Remember that in Linux a ping runs forever unless you tell it the number of packets to send with the count option.

Note: if you decide to run the Python script from your CentOS VM, you will have to add a module to your Python installation. To add the module, you need to install the tool Python uses to add modules (or packages) which is called [pip \(Package Index\)](#). The repository that contains pip for RedHat variants is called [Extra Packages for Enterprise Linux or EPEL](#). The following commands will add EPEL to your yum repository list, install pip, and then install the module that the exploit script needs. The module is called requests, which gives Python the capability to build HTTP requests.

```
sudo yum install epel-release
sudo yum install python-pip python-wheel
sudo pip install requests
```

If you run the Python script from your Kali VM, the requests module is already installed.

#### Questions Part 4

- 1) Did your first try at the exploit work? Include a screenshot of the command you used and a screenshot of tcpdump on your AWS instance.