Project 3

CS7543 Network Security, Dr. Papa

Ian Riley, Graduate Student

04-20-18

Description

The objective for the last and final project was to implement a basic IDS on-top of our preexisting project that could detect attacks in both fragmented and non-fragmented IP packets. There were two tasks that were required to complete this objective. First, a set of rules had to be defined that would apply to the packets of the different attacks. Sample packets were provided that could be studied. Second, an IDS had to be implemented that could read these rules and apply them to packets that were sniffed.

Rules

As per the guidelines of the project report, all rules were designed using the snort rules specification. However, rather than using the full specification, only those fields that were needed and specified in the report were used. Each rule contained an action, a protocol, a source IP address and port, a direction, a destination IP address and port, and a set of options that could be used to test fields in the header of a specific protocol, whether that be IP, TCP, UDP, ICMP, or ARP. Once the rules specification was understood, rules needed to be designed that could detect the three attacks that were provided: WinNuke, Teardrop, and Jolt 2. Each of these attacks is a type of DOS attack and each attack has a signature which can be tested for.

To develop the rules, I first researched what the attacks are. The WinNuke attack is an exploit of Windows Operating Systems that sends a string of out-of-band data to a target computer on port 139 using TCP. A sample of the attack was provided which targeted the IP address under the subnet of 192.168.1. To detect the WinNuke attack, a rule can look for TCP traffic directed from anywhere to 192.168.0/24:139. However, I was worried that this might be too general. Further suggestions advised looking for the TCP U flag to be set so the rule was modified to test for this flag. Luckily, one of the sample rules was already crafted for this very attack so I reused that rule. The rule to detect WinNuke attacks is stated below.

```
alert tcp any any -> 192.168.1.0/24 139 (msg: "DOS Winnuke attack"; flags: U+; logto:"\\output-3\\winnuke.log";)
```

The Teardrop attack is not as sad as it seems. The Teardrop attack is a type of DOS attack that seeks to exploit a vulnerability with fragmented packets. Many systems can not properly handle overlapping fragmented packets and crash when they receive them. However, we implemented an IDS that can handle overlapping fragmented packets in project 2 so this was not a problem. However, from observation of the Teardrop attack that was provided, it would appear that the packets simulating a Teardrop attack are not actually a proper Teardrop attack. The packets have the ‘Don’t Fragment’ flag set in the IP header and aren’t actually fragmented. Further reading on Teardrop attacks led me to

conclude that all Teardrop attacks would have their IP identification field set to 242. This was the exact value of the IP identification field in packets of the Teardrop attack that were provided so I designed a rule that would test for this field. The Teardrop attack was aimed at the 129.244 subnet. The rule to detect Teardrop attacks is stated below.

alert udp any any -> 129.244.0.0/16 any (msg:"DOS Teardrop attack"; fragbits:D; id:242; logto:"\\output-3\\teardrop.log";)

Lastly, the Jolt 2 attack is a type of DOS attack that uses illegally fragmented UDP or ICMP packets. Jolt 2 attacks will include only a fragment whose stated total length and fragmentation offset will be correct, but whose packet payload size would violate the maximum size of 65,535 bytes if included. In essence, this packet requires that an IDS check that the payload size and the fragmentation offset are reasonable. An IDS cannot trust that the total length as specified in the IP header is correct. Both of the samples of the Jolt 2 attack that were provided used a fragmentation offset of 8190 and had a payload size of 26, when the stated payload size was 9. A fragmentation offset of 8190 and a total length of 29 yield that the size of the fully assembled packet would be 65,529 bytes which is under the acceptable limit of 65,535 bytes. However, if the payload size is actually 26 bytes, then the total length would be 65,546 which would be invalid. Therefore, I designed a rule for the Jolt 2 attack that would look for a fragmentation offset of 8190. This left 15 bytes remaining so I then ensured that the rule would match any packet with a payload size greater than 15. The Jolt 2 attack was targeted at the 192.168.1 subnet. The rule to detect Jolt 2 attacks is stated below.

alert ip any any -> 192.168.1.0/24 any (msg:"DOS Jolt 2 attack"; dsize:>15; fragoffset:8190; logto:"\\output-3\\jolt2.log";)

IDS

Once the rules were created, it was necessary to build a parser that could parse the rules and then integrate that parser into my preexisting project so that the rules could be read and applied to the packets that were sniffed. I decided to design the rules parser in a completely object-oriented manner. The rules were read from a file, each line in the file corresponds to one rule. Each rule is split into individual parts, each part corresponding to one field of the rule. The options field of the rule is then split into different parts where each part represents one option. Each of these different fields and parts are represented with a different class and are contained within a distinct instance, all held under one Rule object. All of the Rule objects are held under one RuleModule object which is used by the IDS. Once a packet is received, it is sent to the RuleModule which then sends it to each rule to be tested. Each rule sends the packet to each of its fields which test the packet. If any test yields false, then the rule does not apply to that packet.

After the packet has been tested by each of the Rules, those that tested true are returned as a list to the Sniffer. Only those Rules which require an alert are alerted and only those rules with a specified logto option are logged to an output file. If there is no logto field, then the alert message is output if there is a msg option which is set. If there is no msg option which is set, then the alert is ignored. Writing the code for the IDS in a completely objected oriented fashion was long and arduous, but there were not many curveballs. The only case that caught me off-guard was the Jolt 2 case since it never sent enough packets for full assembly. As such, I designed the IDS so that each fragment is tested and so that all assembled datagrams are also tested.

Results

There were four sample files that were provided: one for the WinNuke attack, one for the Teardrop attack, and two for the Jolt 2 attack. I created four runtime configurations which tested each of these sample files and logged the results to a logfile. The four run configurations are as follows. The rules were provided above.

WinNuke:

-rules C:\\Users\\ianri\\Workspace\\CS7473\\resources\\rules.txt -r C:\\Users\\ianri\\Workspace\\CS7473\\resources\\winnuke.dat -o C:\\Users\\ianri\\Workspace\\CS7473\\output-3\\winnuke.txt

Teardrop:

-rules C:\\Users\\ianri\\Workspace\\CS7473\\resources\\rules.txt -r C:\\Users\\ianri\\Workspace\\CS7473\\resources\\teardrop.dat -o C:\\Users\\ianri\\Workspace\\CS7473\\output-3\\teardrop.txt

Jolt 2:

-rules C:\\Users\\ianri\\Workspace\\CS7473\\resources\\rules.txt -r C:\\Users\\ianri\\Workspace\\CS7473\\resources\\jolt2packet.dat -o C:\\Users\\ianri\\Workspace\\CS7473\\output-3\\jolt2.txt

Jolt 2 UDP:

-rules C:\\Users\\ianri\\Workspace\\CS7473\\resources\\rules.txt -r C:\\Users\\ianri\\Workspace\\CS7473\\resources\\joltUDP.dat -o C:\\Users\\ianri\\Workspace\\CS7473\\output-3\\jolt2udp.txt

The rule for the Jolt 2 attack applies to both the Jolt 2 packet file that was provided and the Jolt 2 UDP file that was provided. I used the same rule and renamed the log files to match the sample dat file that was tested.