

CMSC 491: Cyber Attacks and Intrusion Detection

Assignment 3

October 18, 2015

Due November 1, 2015 at 11:59pm through blackboard submission. In this assignment, you are expected to work individually. You may use any sources that you want, but you must cite them. The submission should be a zip file named "A3_ < your name > .zip", which contains a written assignment "A3.pdf", and all source code files `sploit1.java`, `sploit2.java`, `sploit3.java`, and `firewall.java`. Source code should be well documented for a high score. Please include authors name in the code. My office hours are M/W 3:30pm - 5:30pm in E4234. You can also email me if you have questions.

1 FIREWALL HACKING

(20 pts) Suppose that VCU has determined that YouTube has a detrimental effect on students' productivity and has banned access to YouTube from all VCU machines with the help of firewalls. Your objective is to circumvent these firewalls.

Note: This problem requires some knowledge of TCP/IP headers. A short summary is at the end of this assignment.

The first firewall is stateless. It discards all packets going to `www.youtube.com`. However, this approach is too simple to work on its own. Users could use a proxy to connect to YouTube. Therefore, the firewall also discards a packet if it contains a particular string (e.g., "youtube"). The other two firewalls are stateful. They take into account that transmitted data might get split up into multiple IP fragments or TCP messages.

You should write three exploit programs, `SploitX.java` ($X = 1..3$). Each program should create one or multiple packets and pass them to one of the four firewalls. The firewall will investigate a packet and discard or forward it. See `sploit.dummy.c` for an example.

We provide a fake proxy server. This is not a real proxy server that can be used to access a Web server or that processes and responds with real *TCP/IP* packets. Instead, you need to use routines provided by us to send fake *TCP/IP* packets to the server via the firewall and to display the server's response. See `sploit_dummy.java` for an example. Your goal is to generate fake *TCP/IP* packets that will not be dropped by the firewall and that form a TCP stream containing the forbidden content (e.g., "youtube") so that the stream will be accepted by the fake proxy server to provide connection to youtube service.

Rules:

- You should write one exploit for each firewall.
- Each of the firewalls has at least one fundamental design flaw. It may be necessary to exploit the same flaw in multiple exploits. However, in each exploit you should also exploit at least one additional flaw that is different from the flaws exploited in your other exploits. Talk to the instructor if you are not sure whether two exploits are different enough.
- A single exploit can interact with only one type of firewall.
- Exploits that are based on CPU or memory exhaustion are not allowed. Of course, real-world firewalls might fall victim to such an exploit.
- Your exploit must be accepted by our fake proxy server. You can access this server to test your exploits; there is a server class provided with this assignment. If you execute your exploit program, the `firewall_finish()` routine will forward all non-filtered packets to the server and output the server's response. The routine does not work on other machines.
- Do not mis-use the firewall routines. See `sploit_dummy.java` for an example of proper use.

We will compile and execute your exploits on a linux machine in the following way:

1. `>javac sploitX.java`
2. `>java sploitX`

In this assignment, you are expected to finish the following tasks:


1. In a file called `a3.pdf`, for each of the three exploit programs: identify the exploited vulnerability/vulnerabilities, say how your exploit program exploits it/them, and describe how it/they could be repaired [3 marks total].

2. Complete the corresponding `splloitX.java` file ($X=1..3$) that exploits one or multiple vulnerabilities in a firewall [12 marks total].
3. Implement a new firewall called **stateful_firewall3 (packet ip)** in class `firewall` that does not have the flaws in the previous firewalls. We will test the new firewall using our testing `splloit` code [5 marks total].

In the assignment folder, you can find the source code for the firewalls, `firewall.java`, a dummy exploit program, and the various class files. Do not modify any of those source files in your assignment except adding a new firewall method as requested. We will use the original class files for testing. You are not given the source code for the server. Note that the server might process or filter packets in a different way than the firewall does. Also, the server is somewhat picky and does not allow all packet sequences that would be valid in practice. Finally, if you are not familiar with *TPC*, you might want to read up on it.

Simplified TCP/IP Information:

To begin, information on the internet is transferred in IP packets, which often contain TCP messages. Both the IP and TCP protocols contain headers explaining what the packet contains. For our purposes, we assume that all IP packets use IPv4 and both TCP and IP headers are 20 bytes long. The header fields we will use are:

- Total length: Length of this packet in bytes, including the lengths of the headers.
- Identification: Unique packet id. Same for fragments of same IP packet, different for other IP packets.
- Flags: We will only use MoreFragments (MF). 1 if more fragments expected, 0 for last/only fragment. 
- Fragment Offset: Location of this IP fragment's data in original IP packet, in units of 1 bytes.
- Source Address: IP address of sender. Usually identifies a machine. In network byte order.
- Destination Address: IP address of the recipient. In network byte order.
- Source Port: Identifies sending application; typically random.
- Destination Port: Identifies receiving application, e.g., port 25 is used for email.

A TCP stream is the data that a sender host wants to send to some receiver host, such as an email. To send the data, the sender host breaks up the TCP stream into TCP messages. Each TCP message is put into an IP packet (see below), which is then transmitted.

The sender host then fills in the header fields of the IP packet containing the TCP message and sends it to a router on the Internet. A packet can be split into fragments. Each fragment is itself an IP packet and will have the same source/dest IP address and id. The last fragment

will have MF set to 0; all others will have it set to 1. The offset is calculated as the offset from the start of the original payload data to the start of the payload data in the current fragment, in units of 1 bytes. For example, if the original payload is "abcdefg" and the first fragment contained "abcd" as payload, then the second fragment contained "efg" should have offset 4. If an IP packet is broken into multiple fragments, the server will first extract all payload in each fragments. Next, the server will re-assemble the payload based on their offset. The unpredictable nature of Internet routing means that IP packets might be received out of order, and fragment offsets can be used to re-order them appropriately.