SYN Flood Mitigation

Abstract
One of the most common denial-of-service attacks on the Internet today is the TCP SYN flood. In this project part, we will look at different options of mitigating this attack type. Specifically, you will implement and compare mitigation strategies working at the transport and the application layers, and evaluate the success and performance if the mitigation measure has context knowledge about the application layer protocol.
After you can successfully defend against the SYN flood, we will turn our attention to legacy devices, in other words hosts and systems that can neither be upgraded to adhere to current protection standards nor be replaced by an alternative. Within the context of a SYN flood, we will practice how an IPS can man-in-the-middle connection attempts to filter out requests that otherwise could harm the vulnerable device.
Note: For the last class project, you have the choice whether to do this project or project 7.

Whenever there is a finite resource which can be consumed by a remote party with ease, there is the potential for a *resource depletion attack*. During the discussion of the link, network and transport layer, we have seen several types of resources which could be maliciously exploited, such as the association table in a wireless access point or the finite number of transmission control blocks in the transmission control protocol by means of source-spoofed SYN packets.

These so-called *SYN floods* are the most common distributed denial-of-service attack in today's networks. They are easy to execute, require minimal resources of the adversary and yet are very effective against an unprotected system. As we have seen within the context of TCP SYN flood mitigation in chapter 6, increasing the number of resources to mitigate a resource depletion attack is rarely successful. If we were to increase the amount of concurrent connections the operating system could have open, reduce the amount of bytes consumed by each client, or release a half-open connection earlier, the system will be able to withstand more SYN flood attack packets, but it will not change the fundamental outcome. As the victim is still expending its resources at a rate faster than it takes the attacker to generate packets, the victim will still run out of resources, certainly if the adversary is increasing the attack rate.

In order to successfully defend against resource depletion attacks, the defender has to fundamentally change the mechanics of the game. As long as there is a resource that is consumed, the vulnerability cannot be mitigated. TCP SYN cookies are such an alternative solution, as they release the server from the need to store the state of the connection and instead offload this information onto the network packets – and thus in turn on the client itself. Once the handshake actually completes, in other words it turns out that the source address is authentic, the server can verify the connection and finally store the state based on the information in the incoming packet. As the server will expend no resources until the completion of the TCP handshake, the potential depletion of the connection pool due to spoofed SYN packets is thus averted.

In this assignment, you will investigate several aspects of SYN flood attacks and mitigation. You will begin with a measurement of your operating system's TCB size and release cycle to estimate the asymmetry a server would experience in such an attack. While many operating systems now include mitigation options such as SYN cookies, they might not be available on every type of device in your organization that needs to provide a service to the outside. If the endpoint cannot be upgraded, the presence of a resource depletion attack can be detected in the network and mitigated by an intrusion prevention system. You will design and implement two strategies to mitigate a SYN flood, first by tracking the connection state and second by implementing SYN cookies in the signal path, and compare the advantages and disadvantages.

The Asymmetry of SYN Floods

Before you will design a control to defend against any type of attack, it is first essential to assess and quantify the behavior of your setup without any mitigation applied, to determine the benefit your countermeasure will provide. As the opening question in this term project part, you will start a web server on one of your test PCs and from a test application on a second test PC inject SYN packets from random source IP addresses towards this server. As the SYN+ACK packets will be returned to these spoofed addresses, you will divert this return traffic back to your measurement application within the firewall, so that your probes do not affect any other party and you can directly measure within your application the effect of the attack.

Figure 1 shows the setup conceptually. In the past parts of the IPS term project, you have already developed the com-

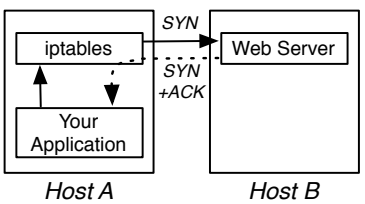

Figure 1

ponents necessary
to realize this setup,
namely the ability
to inject packets with randomly spoofed source IP addresses, an implementation of TCP headers, and the appropriate firewall rules to divert the traffic back to your application.

Setup a web server on the host[1], and implement a test application that injects SYN packets and keeps track if and when the SYNs are being acknowledged.

> **Task 1** *Design a measurement strategy to investigate the size of the transmission control block, as well as determine after which time a half-open connection is being released by your host. Implement your test plan and report on the results.* ∎

DDoS Mitigation

The mitigation of such a distributed denial-of-service attack can be pursued at various points in the network. As discussed before, modern operating systems may include some functionality such as SYN cookies, however there exist a variety of scenarios where a control implemented within the operating system is not suitable. First, recall from the discussion of secure network designs the need for load balancing to scale up services. If the load balancer would store the location to which server a flow was forwarded (in contrast to randomly selecting a destination or using information from the packet such as the source IP/port to deterministically select a target), the resource depletion attack might only be moved to another device in the network. Second, there might be cases where such functionality is not provided by the vendor, or for other reasons cannot be activated for certain hosts. The logical conclusion is thus to then move the DDoS mitigation into the network.

There exist dedicated DDoS hardware appliances, so called "scrubbers", that can be placed in the network path to filter out malicious attack traffic and only pass on "clean", legitimate packets to the destination hosts. These have filtering capacities in the order of tens of millions of packets per second, in this part of the assignment we will look at two fundamental strategies these scrubbers use to perform SYN flood mitigation and implement a proof-of-concept as a module for your IPS prototype.

Clearly, if the IPS needs to actively inspect and drop traffic, an off-path placement through a port monitor will no longer work. Instead, the IPS now needs to sit at the trust border in between two domains, and forward a packet if none of the modules have marked it as malicious. If you do not have access to a PC with two network interfaces or do not want to implement your IPS prototype on a separate hardware, you can emulate this operating mode with some caveats by means of firewall rules on a single host or with a set of virtual machines.

A a preparation step for the remainder of the assignment, implement the necessary changes into your IPS prototype so that a packet is presented for inspection to each module. Modules can return their verdict to the main program on whether the packet should be permitted or dropped from their perspective (if a module does not process this type of packet you can choose whether it returns a "don't care" or whether you update the subscription mechanisms so that uninterested modules do not get a copy of this packet delivered in the first place). Packets can be forwarded if all modules consider them as benign and dropped if one voices an objection. Make sure in your implementation that the processing is asynchronous, so that your IPS continues to deal with packets while the modules are taking their decision, but also ensure that a high peak in incoming packets does not hog the CPU.

Statistical Protection

DDoS mitigation solutions build on a number of different strategies to filter out malicious packets. Some analyze the packets statistically based on header field content, or apply rate-limiting and filtering per source, other solutions also include strategies that are aware of the application layer protocol and interact with the client. In the remainder of this assignment we will first look at protection approaches based on header information and the connection state, then extend the view to interact with the client at the application layer, and finally also design a solution that can act as a drop-in replacement to bring host-based strategies such as SYN cookies to legacy devices that otherwise would not support it.

One elementary strategy to mitigate SYN floods is by means of rate-limiting incoming requests per source address. To accomplish this, your DDoS mitigation module will need to keep track of the state of incoming connection requests. If a source sends SYNs at a high rate but never responds to a SYN+ACK, it is safe to assume that the request was malicious. As soon as your decision threshold is crossed, you may heavily rate-limit incoming requests[2] and by injecting RSTs on behalf of the attacking client to help the targeted server clean-up still unanswered SYN+ACKs as shown in figure 2.

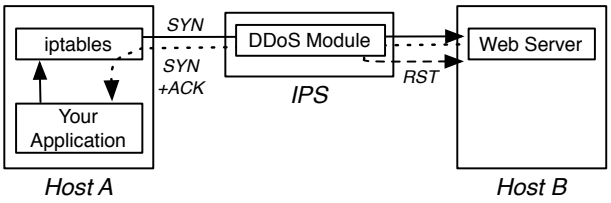

Figure 2

In order not to break any legitimate connections or create a new DoS vector an adversary may use to target specific clients, your module must keep track of connections for which the three-way handshake has been completed, and packets that are related to these connections must be allowed through.

[1] If you are working on Linux, verify that SYN cookies are not already switched by default in your system by checking the value in /proc/sys/net/ipv4/tcp_syncookies.

[2] Note that also here a simple blocking of the source IP address is actually not desirable as it could be abused by a malicious actor in an attack on the availability of a client to prevent it from accessing a service.

Task 2 *Design, describe and justify a strategy that you can use to successfully identify and relate packets belonging to established connections with the information available at layer 3 and 4. Implement the connection tracking and rate-limiting functionality described above in your IPS module. Test the resilience of your web server with the addition of the DDoS module. Is it sufficient to thwart off the attack - if not, comment where your solution falls short.* ∎

Behavioral Protection

An alternative approach to filter out malicious connection attempts from legitimate traffic is to analyze client requests and interact with them at the content level. This on the one hand is more complex as it requires some understanding of the application-layer protocol encapsulated in the packet, but on the other hand can open new possibilities for detecting more advanced distributed denial-of-service attempts. Being application-layer aware, these strategies are thus frequently referred to as layer 7 DDoS mitigation.

The fundamental reasoning behind layer 7 filtering is that it further slims down the attack surface available to an adversary. Suppose that an actor is attempting to take down a web page in a denial-of-service attack. As the IPS or a DDoS mitigation appliance is filtering out the SYN flood from spoofed IP addresses, the attacker could rent the services of a botnet to request a web page from the server over and over again. As the source addresses of the bots are not spoofed and the TCP handshake will thus complete, the basic filtering strategy of keeping track of half-open connections will no longer work, and with hundreds of thousands of clients continuously trying to connect the available resources of the service are soon exhausted again.

The botnet software (or in case of for example IoT botnets the limited capability of hijacked set-top boxes, routers etc.) will however unlikely behave like a "real" web browser, and this difference can be used by the defender to again sort out malicious from legitimate requests. For example, as a web page has temporarily moved, a web server can redirect a client to the new location by returning a 302 response code. If the web browser would send a request to get the index.html page on the website of company.com,

```
GET /index.html HTTP/1.1
Host: www.company.com
```

the server response

```
HTTP/1.1 302 Found
Location: http://www.company.com/?special-tag
```

will trigger the client to request the alternative URL. As a malicious source will likely not have implemented the full HTTP stack, it will rather request the same origin over and over again. The legitimate client – or a client that has implemented the HTTP protocol – will understand and follow the instructions of the server as shown in figure 3, then be redirected again from the customized URL to the original web page, and being recognized as a legitimate client from now on let through to the actual server.

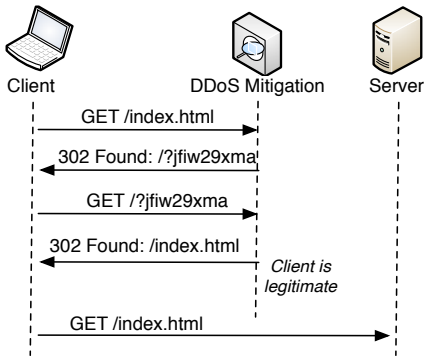

Figure 3

In addition to redirecting the client web browser, the network can also verify whether it is communicating with a "full" web browser by sending by a custom header in the HTTP response that the client needs to interpret, by sending an HTTP cookie to the client, or by including some javascript code that is executed by a regular web browser but not likely to be (correctly) interpreted by a "dumb" client that can only negotiate a TCP connection and send a HTTP request. The client check can become arbitrarily complex in terms of the client's interpretation of the HTTP protocol, and involve the human user for example by solving a captcha.

Task 3 *Evaluate this approach. Against which threats will it be effective, against which ones not? Does this solution generate any false positives?* ∎

As the DDoS appliance is a dedicated device, customized and specialized for this task, it is able to handle a lot of concurrent requests and is more resilient in an attack than a server. Although we are focusing in this example on a web server as the hypertext transfer protocol is human-readable and easy to understand, similar layer 7 mitigation concepts also apply to other application protocols.

Task 4 *Modify your DDoS protection module that it can be switched from layer 3/4 to layer 7 - HTTP protection mode. Now the IPS first interacts with the client. Once it has successfully completed a TCP handshake with the IPS, your module present the client with two 302 redirects as shown above. If the client correctly follows these, the client IP is whitelisted and data is passed through. Test your module with your SYN flood test application, a basic command line HTTP client such as curl or wget and a web browser.* ∎

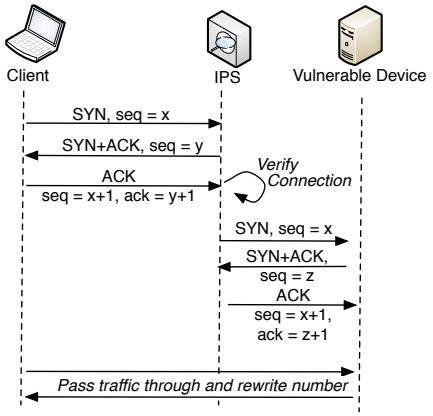

Figure 4

Protecting Legacy Devices

An important task for intrusion detection and prevention systems is also to shield clients and appliances that cannot be patched or upgraded from known attacks, and more fundamentally following the defense-in-depth paradigm even not to expose clients to obviously invalid traffic in the first place. For both of these reasons, IPSes would still often include a rule that filters packets for evidence of the "ping of death", even though this vulnerability is long solved in modern operating systems.

To visualize the first concept in a practical although hypothetical scenario and allow us to implement the concept of SYN cookies in an easy manner without writing a kernel module, suppose that there is a host on the network that needs to be configurable via a web interface, but will crash due to some software bug when it accumulates too many half-open connections. Since we want to shield the device and in this case neither rate-limiting (the adversary could send one connection attempt for each IP in your organization which could not be effectively be rate-limited) nor application-protocol mitigation is an option, we could first check within the IPS whether the connection will complete and then pass the data along. As we do not want to expose the IPS itself to a resource depletion attack, the handshake the IPS performs is protected by means of a SYN cookie. As soon as the TCP handshake is validated from the cookie information, the IPS negotiates a connection with the actual device and will pass through all data between client and server, carefully rewriting the sequence number by an offset as shown in figure 4.

> **Task 5** *List and explain which pieces of information must be minimally included within a SYN cookie to prevent a SYN cookie from being circumvented and at the same time not drop legitimate connection attempts. Implement the solution shown in figure 4.* ■