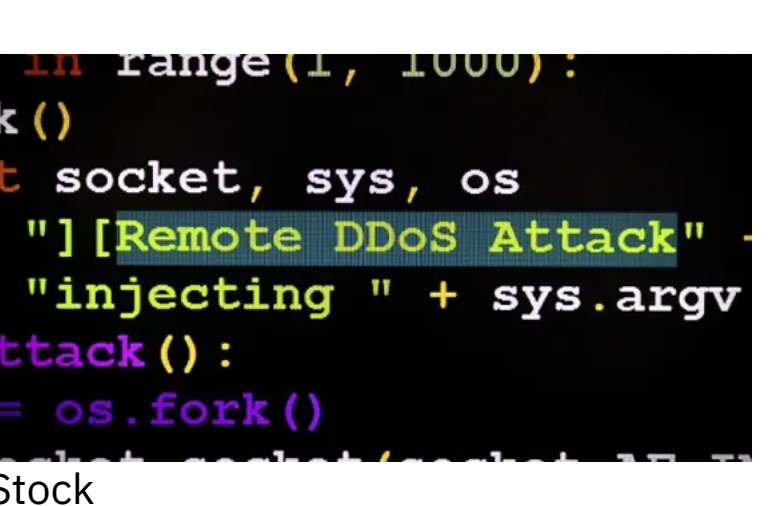
Dissecting a Hacktivist's DDoS Tool: Saphyra Revealed

September 23, 2016 | By Dave McMillen | 4 min read



iStock

Distributed denial-of-service (DDoS) attacks have been all over the news in recent months, with hacktivist groups taking major targets completely offline. According to IBM Managed Security Services data, the vast majority of DDoS attacks come in one of two flavors: SYN flood attacks, in which bad actors send multiple SYN requests to a victim's webserver in an attempt to consume enough resources to render the system unresponsive, and UDP/DNS attacks on network layers 3 (network) and 4 (transport), also known as reflection attacks.

We know, however, that attackers are constantly tweaking their techniques. With this in mind, we decided to take a look at a newer DDoS tactic. The tool in question, dubbed the Saphyra iDDoS Priv8 Tool, targets network layer 7 (application) and results in an HTTP flood DDoS attack. This tool was responsible for taking down the NASA website earlier this year, according to Yahoo Tech. Other modifications of this tool are called Sadattack, Thor and Hulk.

What Is an HTTP Flood Attack?

An HTTP flood attack is a type of layer 7 application attack that utilizes the standard, valid GET/POST requests used to fetch information, as in typical URL data retrievals, during SSL sessions. An HTTP GET/POST flood is a volumetric attack that does not use malformed packets, spoofing or reflection techniques.

How Saphyra Works

The Saphyra iDDoS tool is a Python script that can be run on virtually any device, including mobile phones. Let's take a look at this relatively simple script to understand how it operates and why it is hard to defend against.

Figure 1. Saphyra iDDoS Tool Command Line Interface

Figure 2. Saphyra iDDoS Tool script header

The script contains over 3,200 unique user agent strings and over 300 unique referrer field strings. This would allow for more than 1 million possible combinations of user agent string/referrer instances. When the tool is executed, a unique combination is sent in the form of HTTP requests to the victim's website.

Its main purpose is to generate unique requests, avoiding or bypassing caching engines and directly impacting the server load. This can result in website failure as the server becomes quickly overwhelmed by the volume of requests. The tool interface boasts an affiliation with almost 1.8 billion bots, but this could not be confirmed at the time of analysis.

The Tool's Techniques

The Saphyra tool operates using a variety of techniques, including:

- Obfuscation of source client: By using a list of known user agents, the user agent is constructed as a random value out of the known list.
- Referer spoofing: This enables the sending of incorrect referer information in an HTTP request to prevent a website from obtaining accurate data on the identity of the webpage previously visited by the user.
- Persistence: The tool uses standard HTTP commands to force the server to maintain open connections by using keep-alives with definable time window.
- No cache: By requesting the HTTP server for no cache, the sever presents a unique page for each request.

Figure 3. Example of user agent strings in script

Figure 4. Example of referer strings in the script

Below is what the traffic looks like. Note the random GET request as well as user agent/referrer fields.

```
GET /?o.=Uc.z^.P].. HTTP/1.1
Accept-Encoding: identity
Host: victim.com
Keep-Alive: 113
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; de-de) AppleWebKit/418 (KHTML, like Gecko) Shiira/1.2.2 Safari/125
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close
Referer: http://techtv.mit.edu/search?q=SZCqT.U]..y`y.U.d..n.cMU.Bjf.Ugjt\SyjT\z.d.A^qX...{...Z...d.D
Cache-Control: no-cache
```

GET /?Ch.Bg^=.r.J HTTP/1.1
Accept-Encoding: identity
Host: victim.com
Keep-Alive: 141
User-Agent: Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close
Referer: http://www.google.com/?q=~o~.Es..Te...y^F.j....wCB~Lta..`N}..FIJ.Mqom`..E.Qyn|.{GM.XhzN.Fx..C.P.G\[r..VM...zoo.`..p..m.
Cache-Control: no-cache

```
GET /?IVC=sGQj.ck HTTP/1.1
Accept-Encoding: identity
Host: victim.com
Keep-Alive: 130
User-Agent: Lynx/2.8.6rel.4 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.8g
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close
Referer: http://millercenter.org/search?q=DRn}W.}q.H.w.~.QImfZb.H.H.oBU|L.N.aBG[...X.^rmumr.w.F...ZD..Vf|akaPhSs..Kj^.I..SYtbH.dUp..m
Cache-Control: no-cache
```

The figure below shows a small snapshot of traffic while executing the tool. We were able to generate over 7,500 individual connections per second against a test server in our lab.

Figure 5. View of traffic in Wireshark depicting randomized HTTP GET requests

Mitigation Methods

HTTP flood attacks are some of the most advanced nonvulnerability threats being perpetrated against web servers today. It is difficult for network security endpoints to distinguish between legitimate and malicious web traffic. This could create a high number of false-positive alerts. Additionally, rate-based detection engines are incapable of detecting HTTP flood attacks when the traffic volume of HTTP floods is under detection thresholds.

The Saphyra tool is designed to prevent server defenses from recognizing a pattern and filtering the attack traffic. There are, however, some tactics that can be deployed to both an IPS and to the web server itself.

First, consider using a rewrite modification on NGINX. In /etc/nginx/conf.d/default.conf or similar, add the following inside the server block:

```
if ($args ~* "(.{1,})=(.{1,})" ){
rewrite ^/$ /444_rewrite?;
}
location /444_rewrite {
return 444;
}
```

The regex argument (.{1,})=(.{1,}) tells Nginx to redirect all GET requests that have any characters with = between them and redirect them to 444 (No Response).

The example GET request below shows the equal sign that is common to all requests:

GET /?uQQ[oKYF%7DH=TPTYKsO%257D%257E%255C HTTP/1.1

This tactic can be used on other web platforms as well. Consult your server documentation for instructions to create a rewrite mod.

Distinguishing Between Legitimate and Malicious Traffic

With some DDoS attacks, it's difficult to determine what traffic is legitimate and what is malicious. The best defense is a comprehensive incident response plan, including failovers and a methodology for identifying, analyzing and neutralizing the threat. As a secondary tactic, consider a managed security solution that deflects and absorbs DDoS traffic before it reaches the target.

Tags: Cybercriminals | Distributed Denial-of-Service (DDoS) Attacks | Hacking | Hacktivism

Dave McMillen
Senior Threat Researcher, IBM X-Force

Dave brings over 25 years of network security knowledge to IBM. Dave began his career in IBM over 15 years ago where he was part of a core team of six IBMers...
read more

Popular Articles

Sep 28, 2020 | 9 min read
Ransomware 2020: Attack Trends Affecting Organizations Worldwide

Oct 12, 2021 | 6 min read
What Is Extended Detection and Response (XDR)?

Jan 28, 2022 | 5 min read
What You Need to Know About Data Encryption Right Now

Nov 2, 2021 | 6 min read
Taking Threat Detection and Response to the Next Level with Open XDR

More from Threat Research