radware

A Game of Cat and Mouse: Dynamic IP Address and Cyber Attacks

In recent years, cyber attackers have adopted a new, more surreptitious operational mandate; one that employs a set of strategies and technologies that dramatically complicates the detection process. At the forefront of these rogue-like tactics: serving up dynamic IP addresses.

Dynamic IP addresses are an effective way to defeat IP-based defense systems: launch application-level attacks that originate from real—but dynamic—IP addresses. This paper outlines some of the most common variations of dynamic IP attacks, explores challenges in defending against them, and points to best practices for thwarting these attacks.

Overview

In general, dynamic IP attacks target Layer 7, the application layer. Using real IP addresses, they establish a three-way TCP handshake and successfully bypass cookie and JavaScript challenges. These attacks are highly disruptive and difficult, if not impossible, for IP-based defense systems to distinguish between legitimate and malicious visitors.

To overcome traditional defenses, attackers commonly use headless browser software, such as PhantomJS or a Selenium WebDriver. They also employ multiple evasion tactics. To avoid triggering size- or rate-limiting thresholds, they split the load between dozens of IP addresses and constantly add new IP addresses. Human-like “behaviors” are incorporated—starting at different landing pages and mimicking human-like timings and patterns of movement. They can be especially difficult to detect when attacks are low rate and low volume and are spread over time and across a large pool of changing IP addresses.

Types of Dynamic IP Attacks

Dynamic IP attacks come in various shapes and colors, but some of the most common scenarios include:

- **HTTP/S flooding**. This technique involves full-page reloads of dynamic content, fetching large elements and bypassing cache. Imagine 100 visitors arriving from what appear to be legitimate IP addresses and client headers. The empty browser cache issues a full-page reload that fetches about 50 HTML elements. After a minute, the process repeats with a new group of 100 IP addresses—resulting in 5,000 HTTPS requests per second.

- **Password brute-force attempts**. These often target HTTP, FTP, SQL, SSH and RDP. For example, 100 simultaneous clients, each with a unique IP, issue one request per second. After a minute, every client returns with a new IP address, generating 100 password attempts per second.[1]

- **Web scraping/data harvesting by gray marketers**. This technique can be used to attack online ticketing systems, enabling attackers to buy and sell tickets at a profit. Launching 500 clients with unique IPs, attackers monitor 500 tickets, waiting for a dramatic price drop to make a "bargain" purchase. Every client refreshes the pages every 10 seconds. After a minute, each of the 500 clients returns with a new IP—resulting in 500 bots online, each making 50 requests per second.

- **Web scraping/data harvesting by competitors**. This type of attack is similar to the one described above but is executed to collect competitive pricing and plagiarize content. In this type of dynamic IP attack, 100 clients with unique IPs issue 10 requests per minute, with each client crawling through a different category and clicking on items in random order. After three minutes, each client returns with a new IP. The result is the ability to "scrape" 1,000 items per minute.

- **Clickjacking**. This attack involves click fraud on a competitor's pay-per-click (PPC) advertisements. A common scenario: An operator remotely controls 1,000 malware-infected PCs. Every day, the malware generates 1,000 faked clicks on a competitor's PPC affiliate ads, leading to 30,000 monthly clicks. The competitor must then pay the affiliate regardless of whether or not a purchase is made. At one cent per click, the attack drums up $300 for the affiliate.

Methods of Execution

Attackers commonly use one of four methods to gain access to a large pool of IP addresses: malware botnets, lists of SOCK proxies, VPN services or cloud services.

Malware Botnets

The notorious botnet created by the Linux XOR. DDOS malware has been responsible for thousands of DDoS attacks and hundreds of thousands of SSH brute-force attempts. The vast majority of targets infected by this malware are personal home routers or modems, all of which receive dynamic IPs from the respective Internet service providers.

Another example is the recently discovered LinuxEllipses malware, which infects the Linux host. In a sophisticated technique, it installs an anonymous proxy server that carries out future attacks. This malicious behavior further increases the prevalence of dynamic IT attacks.

1 Note that brute-forcing also can be amplified, as explained in the article, Brute Force Amplification Attacks Against WordPress XMLRPC – https://blog.sucuri.net/2015/10/brute-force-amplification-attacks-against-wordpress-xmlrpc.html

Lists of SOCK Proxies

A huge number of SOCK proxies lists are floating publicly on various amateur forums (see Figure 1). New lists are submitted every day, with numerous offers from sellers of "verified and working" lists. Some sites have transformed this into a business of renting SOCK servers for a specific duration. Various attack scripts and tools can use lists of SOCK proxies to generate traffic over thousands of real clients.

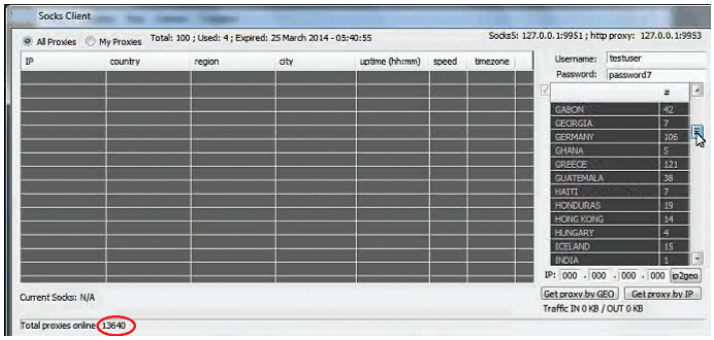

Figure 1: Example of SOCK proxy list

VPN Services

A variety of companies offer virtual private networking (VPN) services—including Hotspot Shield, TunnelBear, Private Internet Access, HideMyAss and CyberGhost, to name just a few. With hundreds of servers spread all over the world, these providers offer a pool of more than 100,000 IP addresses (see Figure 2).

In mid-2015, the free "Hola VPN" browser extension was used to carry out a DDoS attack against the popular 8chan image board. More than 50 million users around the world use Hola to mask their true locations—bypassing censorship and gaining access to geo-blocked content, such as Netflix and BBC programming.

Available Socks Proxy Packages

Name	Socks Daily	Period	Price
Demo - 5	5 proxies	2 days	$2.00
Daily - 5	5 proxies	15 days	$8.00
Daily - 5	5 proxies	30 days	$15.00
Daily - 10	10 proxies	15 days	$11.00
Daily - 10	10 proxies	30 days	$19.00

Figure 2: Example of virtual private networking (VPN) services

Cloud Services

Many cloud providers offer a free tier for developers and users who want to run small-sized servers and applications on cloud infrastructures. Such cloud providers are often the target of hackers, who are continually seeking access to more servers and services for launching malicious activity.[2]

In the quest to attract more customers, many cloud providers offer a simple and easy process for creating a new account. This ease of use has a dark side: insufficient security validations that enable hackers to abuse the cloud services and generate massive quantities of fraudulent accounts. Those fraudulent accounts can then be used to launch network attacks.

Existing cloud customers also can be the target of hackers, who welcome opportunities to obtain leaked or stolen API keys. Hackers can then use those keys to programmatically manipulate cloud services, such as Google AppEngine and Amazon Web Services (AWS). When such API keys fall into the wrong hands, they can be abused—as evidenced by a web developer who recently lost a reported $6,500 in just a few hours after his Amazon API keys were accidentally leaked on the public Internet.[3]

2 IAT 2015 featured an excellent analysis of this topic in a session titled CloudBots: Abusing Free Cloud Services to Build Botnets in the Cloud – http://www.bishopfox.com/news/2015/09/itac-2015-cloudbots-abusing-free-cloud-services-to-build-botnets-in-the-cloud/

3 https://www.humankode.com/security/how-a-bug-in-visual-studio-2015-exposed-my-source-code-on-github-and-cost-me-6500-in-a-few-hours

Simulating an Attack

One of the best ways to grasp this growing threat is by simulating a dynamic IP attack. This simulation is built around these steps:

1. Register and activate an Amazon EC2 account

Amazon offers a free-tier package for new users during their first year. During new account creation, Amazon enforces several mechanisms for preventing abuse of its services. These mechanisms include the need for a valid credit card and the ability to pass an account verification process via email or phone. Amazon ensures that all EC2 resources have well-defined quotas—including a limit of five elastic IP addresses per instance, as well as extra costs for a high number of IP remaps.

Once the account is established, the API and SSH keys can be used to configure two servers: a WordPress backend and a PhantomJS headless browser.

2. Set up a headless-browser server (Ubuntu Linux or PhantomJS) on Amazon

Even on an Amazon m4.xlarge Linux instance, building PhantomJS from source code can take more than 30 minutes and possibly several hours. Some quick online searches uncovered a very fast and elegant solution that leverages a readymade, docker-ized version of PhantomJS. Simple instructions are available on the Docker Hub.[5]

In mere minutes, it is possible to create a Linux Amazon instance from scratch, update it and install a PhantomJS docker container. The next step: customizing a sample PhantomJS script for loading a web page so it has simple userAgent spoofing (see Line 14 in Figure 3). This customization makes it appear to be a Chrome browser running on a Mac.

```
script.js
1  var system = require('system');
2
3  if (system.args.length <2) {
4      console.log('Usage:', system.args[0], 'URL');
5      phantom.exit();
6  }
7
8  url = system.args[1].trim();
9
10 console.log("Loading", url);
11
12 var webPage = require('webpage');
13 var page = webPage.create();
14 page.settings.userAgent = 'Mozilla/5.0 (Macintosh; Intel Mac OS X
       10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.
       0.2454.101 Safari/537.36';
15
16 page.onConsoleMessage = function (msg) {
17     console.log('Page title is ' + msg);
18 };
19
20 page.open(url, function (status) {
21  if (status !== 'success') {
22          console.log('Unable to load the address!');
23          phantom.exit();
24      } else {
25          page.evaluate(function () {
26              console.log(document.title);
27          });
28
29      window.setTimeout(function () {
30              phantom.exit();
31          }, 3000);
32      }
33 });
34
35 page.onLoadFinished = function(status) {
36   console.log('onLoadFinished - Status: ' + status);
37 };
```

Figure 3: Headless browser sample script

3. Write an automated script for dynamically rotating the headless-browser IP address

The Amazon EC2 API can be used to write a simple script that releases the existing IP address, allocates a new one and associates it with the running instance.

For a sample code, see Figure 4. The next step: executing the script in a loop and writing the IP change to a log file.

```
1  IID=i-df111234
2
3  NAME=tiny1
4  IID=`aws --output=text ec2 describe-instances --filters "Name=tag:Name,Values=$
       NAME" | grep INSTANCES | awk '{print $7}'`; echo $IID
5
6  # release existing address
7  OUT=`aws ec2 describe-addresses --output text`; echo $OUT; eipalloc=`echo $OUT |
       awk '{print $2}'`; echo $eipalloc
8  aws ec2 --output text release-address --allocation-id $eipalloc
9
10 # allocate a new one
11 OUT=`aws ec2 allocate-address --output text`; echo $OUT; eipalloc=`echo $OUT |
       awk '{print $1}'`; echo $eipalloc
12
13 # associate it
14 aws ec2 --output text associate-address --allow-reassociation --instance-id $IID
       --allocation-id $eipalloc
```

Figure 4: Dynamic IP allocation using AWS API

5 Find the instructions at https://hub.docker.com/r/rosenhouse/phantomjs2/

The bottom line? It takes only a few seconds to assign a new IP address to a running host. After about 15 minutes, the script has been able to generate more than 300 unique IP addresses.

With the ability to easily generate dynamic IP addresses now proven, the next step in the simulation is to conduct tests using the PhantomJS headless browser. Figure 5 shows the Web Server logs of requests coming from the PhantomJS headless browser. PhantomJS renders the JavaScript WordPress homepage—generating about 10 requests per page load. Notice how the client IP address changes between different iterations of a page load.

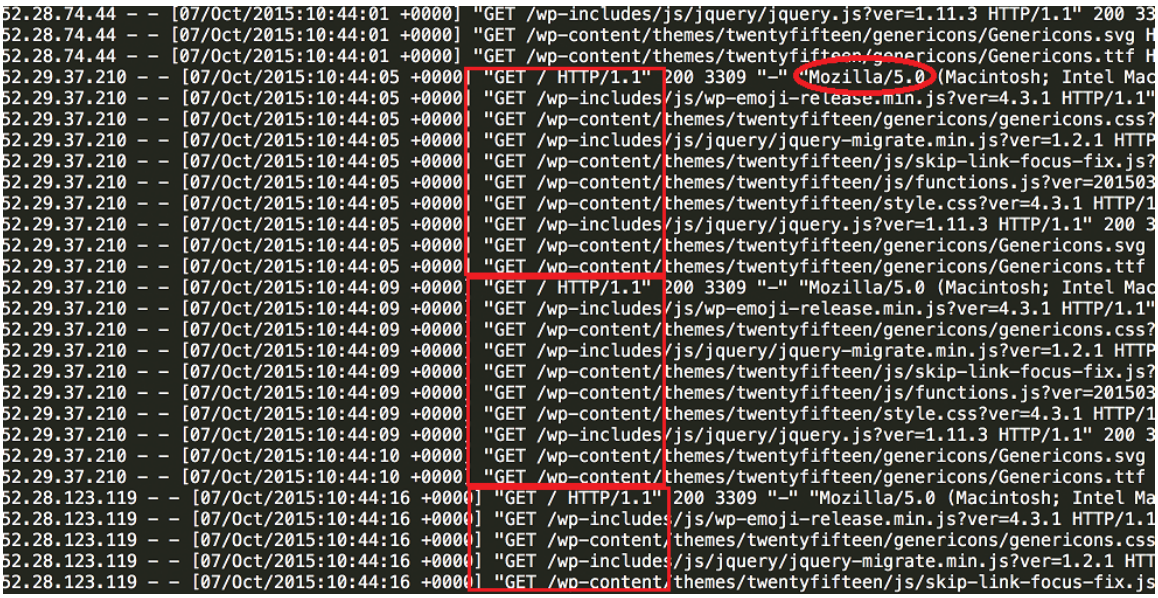

Figure 5: Web Server log file showing the headless browser requests

Comparing these logs with lines belonging to real clients, it becomes clear that they are nearly identical. That can be challenging if seeking to block headless browsers that behave like legitimate users. This test is only one example of the many that can be conducted with a powerful tool such as PhantomJS.

Defending Against Dynamic IP Attacks

It is not unusual for dynamic IP attacks to be overlooked. After all, these attacks are challenging to defend against and most defense systems are not capable of acting against attacks that so closely resemble real user patterns. Even so, Radware expects focus and attention on these attacks to grow as organizations become more aware of the risks.

If traditional cyber and application protection systems cannot thwart dynamic IP attacks, what can organizations do to protect themselves? The answer lies in advanced defense systems that leverage behavioral-based detection mechanisms. These sophisticated capabilities help in identifying malicious bots, headless browsers and other dynamic IP attacks. Ideally, behavioral-based defense should offer an advanced host fingerprinting mechanism, which goes far beyond IP-based detection to identify—and block—malicious actors in real time.

Learn More at DDoS Warriors

To know more about today's attack vector landscape, understand the business impact of cyber-attacks or learn more about emerging attack types and tools visit DDoSWarriors.com. Created by Radware's Emergency Response Team (ERT), it is the ultimate resource for everything security professionals need to know about DDoS attacks and cyber security.