THE INDUSTRIAL AGE OF WEBSITE BOTS: HOW TO DETECT AND BLOCK AUTOMATED ATTACKS

IDO SAFRUTI
CTO

ARIEL SIROTA
VP Engineering
Agenda

• Overview: Why do we care about bots?

• Labs
  • Present concept
  • Explore ways to detect
  • Lab work: implement bot enhancement and detection
  • Discuss how can bots by pass the defence

• Let’s keep it interactive!
Provider of behavior-based web protection

Focus on identifying bots and other automated threats to websites

Today we’ll look at automated attacks generationally
  • The approach by malicious bots
  • The response to identify and block the bad behavior
THREAT LANDSCAPE

ACCOUNT TAKEOVER

FAKE USER CREATION

CARDING

MARKETING FRAUD

CONTENT THEFT (SCRAPING)

CHECKOUT ABUSE
BOT EVOLUTION

SCRIPTS

GEN 1

LATE ‘90s

SCRIPTS + STATE

GEN 2

Early 2000s

HEADLESS BROWSERS

GEN 3

2010

INFECTED USERS

GEN 4

2014

BOT GENERATION

© 2016 PerimeterX™ - Proprietary and Confidential
**GEN 1 BOTS: SCRIPTS**

**Gen 1: Very simple**
Content and price scraping

They don’t have a sense of context, they don’t keep a session about the user

**Detect/block Gen1 Bots**
Absence of cookie
**GEN 2 BOTS: SCRIPTS + STATE**

**Gen 2 bots**
Add the ability to maintain session, have a cookie

**Account takeover, brute force/carding**

**Detect/block Gen 2 Bots**
Detect absence of Javascript

```
import cookielib
import urllib2

cj = cookielib.CookieJar()
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
# use opener to open different urls
```
**GEN 3 BOTS: HEADLESS BROWSERS**

**Gen 3 bots**
Based on a real browser core, adding a scripting engine

- Behaves like a browser - run Javascript, can render images, render sound
- But no human being behind them, usually in a data center or single location

**Detect/block Gen 3 Bots**
Signature approach

```javascript
// Simple Javascript example
console.log('Loading a web page');
var page = require('webpage').create();
var url = 'http://phantomjs.org/';
page.open(url, function (status) {
    // Page is loaded!
    phantom.exit();
});
```
Gen 4 bots
All tests through browser OK, so now we need to focus on user behaviour

- Nuances in behavior to positively ID humans (exonerate)
- More perfect / linear / non-random behavior of bots (incriminate)
- Even when a human is present, it can find a malware parasite in the background with unlikely parallel behavior.

Detect/block Gen 4 Bots Behavioral-based detection
EXONERATE OR INCRIMINATE

Incriminate We assume visitor is a human, it’s up to the evidence to prove us wrong

Exonerate We want proof that visitor is a human, not just lack of proof that a bot
ACCOUNT TAKEOVER

IoT Device Attack

• Peaked at 60 attempts per second
• 8% Success rate
• 3,500 IPs (IoT devices)
• Each node making only 1 request per minute

GEN 2 BOT

© 2016 PerimeterX™ - Proprietary and Confidential
SCRAPING

• Scraping a modern, single page application using Selenium
• Every 3 hours scraping the whole site
• Hundred of nodes, primarily from IaaS providers such as Linode, Digital Ocean & Hetzner
ACCOUNT ABUSE

INFECTED USER

• Malware deployed as a malicious Chrome extension
• Executing Javascript code in the context of attacked site with existing user session
• Creates fake account using legitimate user details pulled from their Google and Facebook accounts
• 100,000s executions from 100,000s of users.
• Real IP, real browser, real user session

Hear more about this tomorrow, Nov 10 16:45
“Abusing Google and Facebook login: On the risks in trusting third-party logins”
TUTORIAL SETUP

SETTING UP A WEBSITE
Clone the tutorial repository

go to https://github.com/PerimeterX/bot-tools/

Clone or download the repository.

To clone:

git clone https://github.com/PerimeterX/bot-tools.git
Setting up Docker

• Hopefully docker is already installed

• Installed images
Test Website

- Build docker image for application
- Or run it locally with python
• Docker containers:
  • Wrap a piece of software in a complete filesystem that contains everything needed to run
  • Guarantees the software will always run the same
• Lightweight:
  • Containers running on a single machine share the same operating system kernel
  • Start instantly and use less RAM
  • Images constructed from layered filesystems, share common files
• Secure:
  • Containers isolate applications from one another
  • Isolate from the underlying infrastructure
Why Docker?

- Set up bot lab, running different bots on a single environment
- Easy to share on different environments
Useful Docker Commands

- Download an image, and all its parents, from the registry:
  ```
  docker pull <image name>
  ```

- Start and stop a container:
  ```
  docker start/stop <container name>
  ```

- To check the running containers execute:
  ```
  docker ps
  ```

- To show last 50 lines and follow the log output of the container execute:
  ```
  docker logs --tail=50 -f <container name>
  ```

- To check the running and stopped containers execute:
  ```
  docker ps -a
  ```

- To show the list of the images run:
  ```
  docker images
  ```

- To run a command inside the container namespace you run the following docker command:
  ```
  docker exec -ti <container name> <command>
  ```

- This is very useful when you want to connect to a running container and work inside of the container.
  ```
  docker exec -ti <container name> bash
  ```
PART I

FIGHTING PRIMITIVE BOTS
LAB 1: USER AGENT

• HTTP header that describes the user agent

• For example:

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36

• Many bad bots use the default - look for them!

• Check your access logs
LAB 1: tools

• Demonstrate *ModHeader*

• Go to https://github.com/PerimeterX/bot-tools/tree/master/lab1
• Don’t be too aggressive - many bots are good
  • Monitor for several days, logging all potential blocks

• It is extremely easy to change it!
LAB 2: USER IP AND IP REPUTATION

• Cloud and Hosting services
• VPNs, free proxies and anonymizing services
• Bad Reputation
• Two “types” of IP addresses
  • physical link (socket)
  • declared/forwarded IP (not reliable but commonly used)
Forwarding headers

• Common method to identify originating IP address, when proxy servers are involved

• Common forwarding headers:
  • X-Forwarded-For
  • via
  • Forwarded (https://tools.ietf.org/html/rfc7239)
  • True-client-IP
About Maxmind DB format

- [http://maxmind.github.io/MaxMind-DB/](http://maxmind.github.io/MaxMind-DB/)
  - The MaxMind DB file format is a database format that maps IPv4 and IPv6 addresses to data records using an efficient binary search tree.

- Open format to efficiently classify IPs
  - provided with various libraries to efficiently process it

- Used by Maxmind to store their IP classification database

- Adopted by us to store our own classification database
  - We are open-sourcing our tools to create mmdb files from IP lists, and tools to generate IP feeds
LAB 2 - details

• Block requests from known cloud vendors
• Use ModHeader
• Go to github - see details
LAB 3: RATE LIMITING

- Protect your infrastructure from high load
- Prevent aggressive brute-forcing and scraping
- Can be done by:
  - IP address
  - Network
  - Session
  - different unique identifiers
LAB 3 - details

- Go to https://github.com/PerimeterX/bot-tools/tree/master/lab3
- Review `wget` options
- You can also try Scrapy, for more advanced capabilities:
  - https://scrapy.org/
LAB 4: CLICK TRAPS

• Fighting crawlers with hidden links

• Create hidden links as honeypots for bad crawlers

• Use \textit{rel}="nofollow" and \textit{style}"display:none;"  
• Deny access on \texttt{robots.txt}

• Humans will not see

• Good bots will not follow!
PART II

GOOD BOTS
FOUR CRITERIA FOR GOOD BOTS

• Declare yourself

• Make it easy to validate your identity
  • Come from declared IP addresses

• Respect robots.txt

• Play nicely!
  • Don’t be too aggressive
LAB 5: DETECTING GOOGLEBOT

• **Googlebot in the user-agent.**
  • See [https://support.google.com/webmasters/answer/1061943](https://support.google.com/webmasters/answer/1061943)

• **Reverse-dns logic:**
  • Get IP of the request
  • Reverse dns on the IP => receive hostname
  • Domain name fro the host is either `google.com` or `googlebot.com`
  • Resolve the hostname, and verify that the resulting IP address is the same as the IP address we started with
  • See [https://support.google.com/webmasters/answer/80553?hl=en](https://support.google.com/webmasters/answer/80553?hl=en)

• Many other browsers followed
LAB 5: details

• go to https://github.com/PerimeterX/bot-tools/tree/master/lab5

• review Google’s instructions on identifying Googlebot
  • https://support.google.com/webmasters/answer/1061943
  • https://support.google.com/webmasters/answer/80553?hl=en

• Review code on lab 5 Readme

• Performance matters:
  • most good bots don’t change IPs often
  • you can generate list offline, and use logs to trigger new evaluation
HANDLING GOOD BOTS

• Whitelisting good-bots
  • Review your access log periodically

• Rate-limit
  • Some bots have controls through robots.txt

• Instruct search engines to not show cached content:
  • `<meta name="robots" content="noarchive">`
LAB 6: JAVASCRIPT SUPPORT

• Javascript is code running on the client... use it!
  • Serve a javascript challenge
  • Page will be served only if the challenge is solved

• No legitimate user runs without cookie support and/or javascript support
  • Specifically - on “sensitive” pages - login, add comment/review, add to cart, checkout, ...

• Remember to first whitelist the good bots!
LAB 6: details

• go to https://github.com/PerimeterX/bot-tools/tree/master/lab6

• Adding a challenge page /html/index.html
  • This challenge page calls a javascript function to set a cookie with a name fpid and once set, will reload the page

• if the cookie exists - serve requested page
• if the cookie doesn’t exist - serve the challenge page
PART III
HEADLESS BROWSERS
• Support Javascript
  • Single page application
  • Full page rendering
• Used for QA and automation testing
  • But also warmly adopted by the hacking community…
• Examples:
  • PhantomJS
  • Selenium
  • Chromium Embedded Framework (CEF)
  • https://github.com/dhamaniasad/HeadlessBrowsers
LAB 7: COLLECT BROWSER INFORMATION

• With Javascript enabled - we can collect much more information

• For instance - specific attributes of the browser

• Headless browsers will typically modify their user-agent header

• The browser however, has an internal attribute javascript can query:
  • **navigator** object, see https://developer.mozilla.org/en-US/docs/Web/API/Navigator
  • Compare `navigator.userAgent` to User-agent header

• Note that some minor changes may occur, so compare smartly
LAB 8: BROWSER FINGERPRINTING

• Different Browsers and headless browsers have different capabilities and can be fingerprint by them.

• https://clientjs.org/ - collects a lot of helpful information

• Example: detect tracks of PhantomJS
  • Look for a function: `window.callPhantom`
SUMMARY
Things to Remember

• Reviewed some basic techniques and tools
  • that still are very efficient today

• Blocking by IP typically isn’t the right thing

• Blocking isn’t always the right thing

• Don’t forget to disable ModHeaders, and enable JS and Cookies!
  • or at least don’t browse the web as a bot from a cloud service ;-)
THANK YOU!

IDO SAFRUTI
ido@perimeterx.com
@safrutl

ARIEL SIROTA
ariel@perimeterx.com