Good Bot, Bad Bot:
Characterizing Automated Browsing Activity
APNIC 52
Nick Nikiforakis
Who am I?

• Associate Professor at Stony Brook
• Areas of research
  • Online tracking
  • DNS Security
  • Web application fingerprinting
  • Mobile Browser Security
  • Attack surface reduction
  • Honeypots and deception
  • Anti-bot technologies
Web bots

- Web bots are programs that interact with websites in automated ways
  - Benign bots
    - Page indexing, link previews, malware detection
  - Malicious bots
    - Scraping, brute-forcing credentials, stealing backup/configuration files, exploiting vulnerabilities

Source: Imperva Bot Report, 2021
Detecting benign web bots

- Benign bots announce themselves

- Google
  - **IP address**: 66.249.66.1
  - **User Agent**: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)

- Bing
  - **IP address**: 40.77.167.41
  - **User Agent**: Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)
Detecting malicious web bots

• This is more challenging

• Malicious bot strategy #1
  • Pretend to be a known benign bot (Googlebot/Bingbot/etc.)
  • Scrape/attack with administrators fearing the blocking of a known benign crawler
    • No one wants to block Googlebot

• Defenses
  • Reverse-DNS the IP address claiming to be a bot

<table>
<thead>
<tr>
<th>User Agent</th>
<th>IP address</th>
<th>Reverse DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla/5.0 (compatible; Googlebot/2.1…)</td>
<td>66.249.66.1</td>
<td>crawl-66-249-66-1.googlebot.com</td>
</tr>
<tr>
<td>Mozilla/5.0 (compatible; Googlebot/2.1…)</td>
<td>67.245.115.115</td>
<td>cpe-67-245-115-115.nyc.res.rr.com</td>
</tr>
</tbody>
</table>
Detecting malicious web bots

- Malicious bot strategy #2
  - Pretend to be a regular user

- Steps that malicious bots can take
  - Spoof User Agents
  - Simulate user actions
  - Low-and-slow
  - Use proxy servers

- Defenses (open ended)
  - Anomaly detection
    - Timing of requests
    - Types of requests
  - IP address blocklists
  - CAPTCHAs when suspicious
  - ????
Robotic yet circular dependencies

Datasets of malicious bots

Systems for detecting malicious bots

Prior Academic Solutions: Manual filtering of web-server logs
Research questions

• Can we curate a bot-only dataset in a way that doesn't depend on our manual-analysis prowess?
  • Benign vs. malicious bots
  • Activities of malicious bots
  • Claimed vs. actual identity of malicious bots
  • Trends of bot-activity over time
Network of honeysites

• Aristaeus
  • A system that provides flexible remote deployment and management of honeysites
• Honeysites:
  • Fully-functional web applications, augmented with state-of-the-art fingerprinting techniques
• A centralized log server pulls logs from each honeysite on a daily basis
  • Injected in a distributed database (Elastic Search)
Overview of Aristaeus

1. Deploy honeysites
2. Log aggregation
3. Bot traffic analysis
Overview of Aristaeus

1. Deploy honeysites
2. Log aggregation
3. Bot traffic analysis
What's the best bait?

- Deployed web applications
  - WordPress, Joomla, Drupal, PHPMyAdmin, and Webmin
    - Tens of years of development
    - Hundreds of vulnerabilities
    - Millions of installations
- Content Management Systems and System Administration tools
  - Promise of data and Remote Code Execution
Client fingerprinting

- Javascript API support
  - Basic support test
    - `document.write()`, `var img` ...
  - Ajax support

- Browser fingerprinting
  - What information can we gather from common JS APIs?

- Support for security policies
  - CSP, X-Frame-Options, Mixed Content (HTTP/HTTPS), etc.
One slide primer on TLS handshakes

• In TLS ClientHello, Clients inform Servers of their TLS capabilities
  • TLS versions
  • Ciphersuites
Everyone's different

- Different TLS Clients implement things slightly differently
  - Chrome/Chromium support GREASE, a mechanism for catching interoperability issues between clients and servers
  - Firefox and Safari do not support GREASE
  - Command-line tools built using Python, curl, Perl, will have different TLS libraries than both Chrome and Firefox

```go
import "net/http"

resp, err := http.Get("https://example.com/")
```

"tlsfp": {
  "ciphersuite": "0xC02F 0xC030 0xC02B 0xC02C 0xC0A8 0xC0A9 0xC013 0xC009 0xC014 0xC00A 0x009C 0x009D 0x002F 0x0035 0xC012 0x000A",
  "tls_version": "0x0303",
  "sig_alg": "0x0401 0x0403 0x0501 0x0503 0x0601 0x0603 0x0201 0x0203",
  "src_port": 22260,
  "record_tls_version": "0x0301",
  "timestamp": "2020-04-25 03:55:59",
  "server_name": "www.historytenantfile.com",
  "ipv4_src": "167.71.193.105",
  "e_curves": "0x001D 0x0017 0x0018 0x0019",
  "extensions": "0x0000 0x0005 0x000A 0x000B 0x000D 0xFF01 0x0012",
  "ciphersuite_length": "0x0020"
}
Overview of Aristaeus

1. Deploy honeysites

2. Log aggregation

3. Bot traffic analysis
Deployment of Aristaeus

- Register 100 domains
  - One condition: Domains should have never been registered before
  - Avoid residual-trust traffic from old sites and buggy systems
  - No public advertisement of these domains

- Spawn one honeysite for each domain
  - 100 VMs in AWS
    - North America, Europe, and Asia
  - Let's Encrypt automatically used to get valid TLS certificates

- 7-month long experiment recording everything and anything
By the numbers

7 Months

26.4 Millions Requests

206 GB Recorded Traffic
Daily traffic

- We keep observing new sources, for the entire 7 months
- Average of 1,235 requests per day
Site discovery

• Since we never advertised our domains, how do bots find us?

• Inspect the Host header of client-side HTTP headers:
  • 44% of bots visit through the IP address
  • 30% present no Host header
  • 26% explicitly ask for our domains
    • Certificate transparency
    • Zone files
    • Prior crawls
### Popular endpoints

<table>
<thead>
<tr>
<th></th>
<th>Wordpress</th>
<th>Joomla</th>
<th>Drupal</th>
<th>PHPMyAdmin</th>
<th>Webmin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓=exists, X=does not exist, ☹=not accessible</td>
<td>✓=exists, X=does not exist, ☹=not accessible</td>
<td>✓=exists, X=does not exist, ☹=not accessible</td>
<td>✓=exists, X=does not exist, ☹=not accessible</td>
<td>✓=exists, X=does not exist, ☹=not accessible</td>
</tr>
<tr>
<td>xmlrpc.php</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>wp-login.php</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/wp-admin/</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/administrator/</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/robots.txt</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/user/login</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/CHANGELOG.txt</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>(POST) /index.php</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/phpmyadmin/index.php</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/session_login.cgi</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
<tr>
<td>/document_root</td>
<td>✓</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
<td>☹</td>
</tr>
</tbody>
</table>
## Popular endpoints

☑️ = exists, ✗ = does not exist, ☢️ = not accessible

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordpress</td>
<td>99.78</td>
<td>99.38</td>
<td>99.72</td>
<td>0.10</td>
<td>39.25</td>
<td>19.36</td>
<td>0.00</td>
<td>0.01</td>
<td>1.56</td>
<td>0.00</td>
<td>0.00</td>
<td>21.58</td>
</tr>
<tr>
<td>Joomla</td>
<td>0.09</td>
<td>0.53</td>
<td>0.14</td>
<td>99.47</td>
<td>37.46</td>
<td>20.87</td>
<td>0.00</td>
<td>0.01</td>
<td>48.16</td>
<td>0.00</td>
<td>0.00</td>
<td>19.36</td>
</tr>
<tr>
<td>Drupal</td>
<td>0.02</td>
<td>0.46</td>
<td>0.05</td>
<td>0.15</td>
<td>9.04</td>
<td>20.07</td>
<td>100.00</td>
<td>99.96</td>
<td>1.23</td>
<td>0.00</td>
<td>0.00</td>
<td>18.73</td>
</tr>
<tr>
<td>PHPMyAdmin</td>
<td>0.04</td>
<td>0.45</td>
<td>0.05</td>
<td>0.13</td>
<td>8.16</td>
<td>19.97</td>
<td>0.00</td>
<td>0.02</td>
<td>47.97</td>
<td>100.00</td>
<td>0.00</td>
<td>19.64</td>
</tr>
<tr>
<td>Webmin</td>
<td>0.08</td>
<td>0.23</td>
<td>0.05</td>
<td>0.15</td>
<td>6.10</td>
<td>19.73</td>
<td>0.00</td>
<td>0.01</td>
<td>1.07</td>
<td>0.00</td>
<td>100.00</td>
<td>20.69</td>
</tr>
</tbody>
</table>
## Popular endpoints

✔ = exists, ✘ = does not exist, ⚠ = not accessible

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordpress</td>
<td>99.78</td>
<td>98.33</td>
<td>99.72</td>
<td>0.10</td>
<td>39.25</td>
<td>19.36</td>
<td>0.00</td>
<td>0.01</td>
<td>1.56</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Joomla</td>
<td>0.09</td>
<td>0.53</td>
<td>0.14</td>
<td>99.47</td>
<td>37.45</td>
<td>20.87</td>
<td>0.00</td>
<td>0.01</td>
<td>48.16</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Drupal</td>
<td>0.02</td>
<td>0.46</td>
<td>0.05</td>
<td>0.15</td>
<td>9.04</td>
<td>20.07</td>
<td>100.00</td>
<td>99.96</td>
<td>1.23</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PHPMyAdmin</td>
<td>0.04</td>
<td>0.45</td>
<td>0.05</td>
<td>0.13</td>
<td>8.16</td>
<td>19.97</td>
<td>0.00</td>
<td>0.02</td>
<td>47.97</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Webmin</td>
<td>0.08</td>
<td>0.23</td>
<td>0.05</td>
<td>0.15</td>
<td>6.10</td>
<td>19.73</td>
<td>0.00</td>
<td>0.01</td>
<td>1.07</td>
<td>100.00</td>
<td>20.69</td>
</tr>
</tbody>
</table>
Popular endpoints

• Clear evidence of tailored attacks
  • Bots first identify that a site is WordPress-powered
  • Then, they start bruteforcing credentials

• Implication: If you don't run multiple types of applications, you won't see a malicious bot
JavaScript and Bot Behaviors

• Out of 1.7M sessions, only 11K (0.63%) supported JavaScript
  • No JavaScript, no JavaScript-based fingerprinting
  • Fingerprints submitted on only 0.59% of sessions

• Honoring of robots.txt
  • We did not observe any violations of robots.txt
  • Popularity of fake disallow entries?

• Shared/Distributed crawling
  • 42.8% of requests with valid cache-breakers bore different IP addresses
  • Widely observed in Google bots (19.6% of all reuse)
  • No re-used cache breakers in malicious bots
Good bot or bad bot?

- We classify the connecting bots as follows:
  - Benign
    - Verified search-engine bots
    - Bots by security researchers and companies
  - Malicious
    - Sending unsolicited POST requests towards auth endpoints
    - Send fingerprinting-related, vulnerability-related requests
  - Other
    - Remainder… we don't know much about those

<table>
<thead>
<tr>
<th>Type</th>
<th>Total SEBot Requests</th>
<th>Verified Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Googlebot</td>
<td>233,024</td>
<td>210,917 (90.5%)</td>
</tr>
<tr>
<td>Bingbot</td>
<td>77,618</td>
<td>77,574 (99.9%)</td>
</tr>
<tr>
<td>Baidubot</td>
<td>2,284</td>
<td>61 (0.026%)</td>
</tr>
<tr>
<td>Yandexbot</td>
<td>4,894</td>
<td>4,785 (97.8%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>317,820</strong></td>
<td><strong>293,337 (92.3%)</strong></td>
</tr>
</tbody>
</table>
Bad Bots Brute-forcing

- Credential brute-forcing attempts
  - 50.8% of total requests
  - 47,667 unique IP addresses
  - Trying common passwords as well as the domain itself
    - [www.example.com](http://www.example.com) as a password for admin panel of example.com
  - 99.6% of bots issued fewer than 10 attempts
    - "Spray and pray"
    - We had observed the same phenomenon on SSH honeypots, in 2017 [A]

Bad bots: Reconnaissance

- Application fingerprinting
  - Attempting to infer the version of a web application or its plugins
  - Matched requests against signatures of WhatWeb and BlindElephant
- 223K requests, 12K bot IP addresses
- Exploitation attempts
  - We focused on server-side exploits from exploit-db (593 signatures)
- 238K requests, 10K bot IP addresses
Bad bots: Reconnaissance

- Searching for backdoors
  - `shell.php`, `cmd.php`, `up.php`
  - 144K requests, 6.7K unique IP addresses
- Searching for unprotected files
  - `.old`, `.sql`, `.php~`, `.zip`, `.bak`, `.env`
  - 52K requests, 5.8K unique IP addresses

- 929 bots did all of the above
  - Minority of bots willing to keep attacking until they are either blocked or they run out of vectors
Bots and TLS fingerprinting

- Unlike JS fingerprinting, TLS fingerprinting worked really well
  - 558 unique fingerprints shared over 10M requests
    - Small number of tools and libraries
- 86.2% of bots claiming Firefox/Chrome were fake
  - Matching signatures of curl, libwww-perl, Go, and Python
- Exploitation attempts do not match real browser fingerprints
Case studies

- Failed cloaking attempts
  - Bots sending two user agents
    - "User-Agent" and "userAgent"
  - Host-header weirdness
    - HOST, hoSt

- Time to weaponize
  - 5 RCE vulnerabilities got discovered during our 7-month study
  - Aristaeus could now observe how fast attackers weaponize a new exploit

<table>
<thead>
<tr>
<th>Software/Firmware</th>
<th>CVE</th>
<th>Time to weaponize</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSSQL Reporting Servers</td>
<td>CVE-2020-0618</td>
<td>4 days</td>
</tr>
<tr>
<td>Liferay Portal</td>
<td>CVE-2020-7961</td>
<td>4 days</td>
</tr>
<tr>
<td>DrayTech modems</td>
<td>CVE-2020-8585</td>
<td>2 days</td>
</tr>
<tr>
<td>Netgear GPON router</td>
<td>EDB-48225</td>
<td>Same day</td>
</tr>
<tr>
<td>F5 Traffic Management UI</td>
<td>CVE-2020-5902</td>
<td>Same day</td>
</tr>
</tbody>
</table>
Conclusion

• As more software moves to the web, so do attackers
  • Even unpopular sites are scanned thousands of times a month by malicious bots
• Honeypots and deception technology can help us attract them and fingerprint them
  • For modest operational costs, Aristaeus outperforms popular OSINT blocklists
  • Identify trends in attacker techniques, tools, and sp exploit weaponization
• Communication details can betray a client’s identity

nick@cs.stonybrook.edu
www.securitee.org