Screw You and the Script You Rode in On
Introductions

Charles Henderson
Director, Application Security Services

David Byrne
Managing Consultant
Agenda

• The Problem
• Current Solutions
• Our Solution
• Examples
The Problem: Automated website access...

- Search engine bots
- Vulnerability scanners
- Spam-bots (pills & porn)
- DDoS attacks
- Miscellaneous crawlers
The Problem: Automated website access...

- **Search engine bots**
- Vulnerability scanners
- Spam-bots (pills & porn)
- DDoS attacks
- Miscellaneous crawlers
Common Solutions

• Web Application Firewalls
  – Pro:
    • Do a good job of filtering out automated vulnerability scanners
  – Cons:
    • Aren’t well suited for identifying non-attacks
    • DDoS attacks will almost always be missed
Common Solutions

• Request Throttling
  – Pros
    • Effective at stopping aggressive crawlers
  – Cons
    • Likely to block aggregated traffic (proxy servers or NAT)
    • Or, aggressive crawling can be passed off as aggregated traffic using forged HTTP headers
Common Solutions

• CAPTCHA
  – Pros:
    • Good protection against simple spam-bots
    • Really hard ones can’t be solved by even advanced scripts
  – Cons:
    • Really hard ones can’t be solved by even humans
    • Easy ones can be solved by scripts
    • Everyone hates them
    • You can only use them on key components
CAPTCHA Scope Limitations

• Generally only used on key operations:
  – Account creation
  – Auction bids
  – Comment posts
CAPTCHA Solving

- OWASP AppSec DC 2012, Gursev Singh Kalra released TesserCap
- Nice automation to solve common CAPTCHA formats using Tesseract
- Accommodations to users introduce weaknesses
Digitizing Books One Word at a Time

Type the two words:

Submit

The words above come from scanned books. By typing them, you help to digitize old texts.
Uncommon Solutions

• Honeypot tags (injecting hidden content that only an automated tool would request)
  
• Pros:
  – Theoretically, very sound. Avoiding it requires extensive client-side DOM modeling to identify which components are visible. Files like robots.txt must be avoided, etc.

• Cons:
  – Must be implemented before the problem occurs
  – Many organizations are currently reluctant to implement
Uncommon Solutions

• Honeypot tags (con’t)

• Cons:
  – Only blocks complete crawlers – a price crawler won’t request hidden links
Our Motivation

- Client’s repeated problems with aggressive crawling
- First time was easy to spot
- Second time was a little harder...
- Third time was a huge pain
Our Solution:

- Voigt-Kampff
- Offline log analysis
- Entirely passive
- Designed with the goal to grow into a real-time traffic analysis engine
Voigt-Kampff

• Java-based

• High-performance
  – Designed for multi-core/CPU, high-RAM computers
  – Separate threads for file reading, parsing, analysis
  – Uses java.nio.channels.FileChannel for file reading
  – Regular expressions rarely used, only after initial simpler pattern matching
  – Uses H2 database – easy switching between in-memory and on-disk storage
  – Custom string cache engine (truncated MD5)
Voigt-Kampff

• High-performance (con’t)
  – All behavioral pattern analysis done against “long” data type
  – Javolution collections
  – Log file parsing with modified OpenCSV
Voigt-Kampff Techniques

• Confidence score-based
• Per IP-address analysis
• Attempts to categorize as:
  – Search engine
  – Scripting tool
  – Spider
  – Security scanner
  – Unknown automated
  – Link checker
  – Validator
  – Web library
  – Human
Voigt-Kampff Techniques

• Static analysis
  – Performed against every log entry
  – Typically simpler tests
  – Is started while logs still being read

• Dynamic analysis
  – Pattern creation
    • Baseline of “normal” behavior (only works if most behavior is human)
  – Pattern comparison
    • Checks for deviations from normal baselines
Voigt-Kampff Techniques

• Simplest detection with known user agent strings
  – LWP: libwww-perl/5.821
  – Curl: curl/7.9.8 (i686-pc-linux-gnu) libcurl 7.9.8 (OpenSSL 0.9.6b) (ipv6 enabled)
  – Google images: Googlebot-Image/1.0
  – Java: Java/1.6.0_26
  – Nikto: Mozilla/4.75 (Nikto/2.1.2)

• Implemented as static test
Voigt-Kampff Techniques

• Multiple categories of known user agents
  – Link checkers
  – Security scanners
  – Validators
  – Web libraries
  – Search engines
Voigt-Kampff Techniques

• Other simple tests, all implemented as static tests
  – Requests for robots.txt
  – Requests for sitemap.xml
  – Unknown / unique user agents
Voigt-Kampff Techniques

- Anomalous response code rates...

- Baseline:
  - 200 – 80%
  - 304 – 10%
  - 302 – 8%
  - 404 – 2%

- Anomaly:
  - 200 – 50%
  - 404 – 40%
  - 500 – 10%
Voigt-Kampff Techniques

- Anomalous file not founds (depends on real 404 codes)...
  - 3032 -- /scripts/tracking.js
  - 4268 -- /images/spacer.gif
  - 1 -- /admin.aspx
  - 4729 -- /css/tables.css
Voigt-Kampff Techniques

• Anomalous file not founds (depends on real 404 codes)...
  – 3032 -- /scripts/tracking.js
  – 4268 -- /images/spacer.gif
  – 1 -- /admin.aspx  <--------
  – 4729 -- /css/tables.css
Voigt-Kampff Techniques

• Application entry point (no referer header, or external referer header)

• Most applications will have a relatively small number of entry points
  – Main page
  – Key Google results
  – Login pages
  – Popular bookmarks
Voigt-Kampff Techniques

• Dependency requests: JavaScript, style sheets, images, etc.

• Automated tools may not request dependencies they don’t use (especially large files)

• Passive dependency mapping isn’t easy. Based on referer headers in proximity to original request.

• Requires ALL logs from a web site
Voigt-Kampff Techniques

- Multiple user agents per IP over small time
- Could be aggregated traffic (NAT or proxy)
- Could be automated tool trying to mask its signature
- Low confidence level
Voigt-Kampff Techniques

- Average request rate (requests per IP over a one minute period)
- Could be aggregated traffic (NAT or proxy)
- Low confidence level
Voigt-Kampff Techniques

- Request delays
- Standard deviation for delay between requests for an IP address
- If a client is very consistent in how frequently it sends requests, that is very suspicious
Voigt-Kampff Techniques

- Navigational patterns
Voigt-Kampff Techniques

- Navigational patterns
Voigt-Kampff CLI

usage: voigtkampff [options] <filename>

-\(v\)--verbose
Increase verbose output. Can be used multiple times.

-\(r\)--recursive
Use <logfile> as a directory and recursively search for all log files

-\(f\)--file <filename>
additional log file(s) to parse, can be used multiple times

-\(o\)--format <format string>
A W3C or format string defining the columns. For example, -o "%h %l %u %t "%r" %>s %b "\%{Referer}\i "\%{User-agent}\i"" or -o "date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)"
If this is omitted, voigtkampff will look for a file header, then try to guess the format.

-\(D\)--skip-dependencies
Do NOT perform dependency request analysis. This is useful if you are missing log files from a load balanced website.

-\(m\)--all-memory
Keep all databases in memory for faster performance.

-\(r\)--report <filename>
Report file name. Defaults to report.html
Voigt-Kampff CLI

./voigtkampff -v -v -m ex20120320.log

345 [main] INFO root -
Lines read: 0
Requests parsed: 0
Parsing queue: 0
Static tests: 0

5354 [main] INFO root -
Lines read: 399,960
Requests parsed: 199,722
Parsing queue: 200,495
Static tests: 0

6385 [Static testing thread 0] INFO root - Flushing string cache with 23184 records
10354 [main] INFO root -
Lines read: 562,267
Requests parsed: 382,584
Parsing queue: 167,704
Static tests: 10,799
Voigt-Kampff CLI

991071 [Log parsing thread 1] INFO root - Exiting after 28735316 jobs on Log parsing thread 1
991071 [Log parsing thread 2] INFO root - Exiting after 28735316 jobs on Log parsing thread 2
991071 [Log parsing thread 0] INFO root - Exiting after 28735316 jobs on Log parsing thread 0
Voigt-Kampff CLI

===================================
IP Address - 28.481.381.45
Total score - 100
Possible profiles -
===================================

The user-agent string matches a known scanner: Mozilla/4.75 (Nikto/2.1.2)

===================================
IP Address - 132.278.184.28
Total score - 78
Possible profiles - Unknown automated tool
===================================

The IP had an unusually high number of 404 response codes from the server. 11.31% of the IP's responses were this code, while most clients averaged 1.02%
The IP had an unusually high number of 500 response codes from the server. 5.2% of the IP's responses were this code, while most clients averaged 0.29%
Voigt-Kampff Release

- Not today 😞
- As soon as Trustwave Legal approves it on our return
Questions or Ideas?
Survey

https://www.surveymonkey.com/s/Research12_Byrne_Henderson