Harnessing GP²Us
Building Better Browser Based Botnets

Marc Blanchou
Introduction

• What is it about?
  – Harnessing GPUs with browser-based botnets for distributed and cheaper cracking

• Why should I care?
  – You’re doubtful that the GPU can ever be harnessed for general-purpose computation in a browser
  – You think that only “advanced attackers” can break your crypto or the crypto of the products you use
Agenda

• Introduction
• Better browser-based botnets
• Get permanent code execution in the browser
• Communication
• Leveraging the GPU from within the browser
• What for?
• Examples?
• Conclusion
Who am I?

- Senior Security Consultant at iSEC Partners
- I mainly do application security
- Past experience as a game developer
  - Worked on game engines and GPU optimizations
- Based in San Francisco
INTRODUCTION
• Cracking
  – General-purpose computing
  – Needs parallel computations
    • GPU vs CPU
• FPGA?

Nvidia.com
GPU Farm?

Bitcoinminer.com
• GPU Parallelism is almost doubling every year
• Way faster evolution than CPU
EC2 Instances?

- ‘Renting’ GPU power
- Cluster GPU Quadruple Extra Large Instance
  - 33.5 EC2 Compute Units (2 x Intel Xeon X5570, quad-core)
  - 2 x NVIDIA Tesla “Fermi” M2050 GPUs
  - 2.10$ to 2.60$ hourly

- NVIDIA Tesla limitations for cracking

- Expensive?
• Definition

• What for?

• Real practicality for general purpose computing?

• “ZeroAccess” botnet
  – “2.7 millions annually in bitcoin mining” (Sophos)
Not everyone has powerful graphic cards, though

• New on-chip graphics on recent CPUs

• Intel Ivy Bridge (2011) and Intel HD 4000
  – Great support for recent techno
  – Relatively decent computing power
  – Low power consumption / heat – (discreet!)

• Intel Haswell (2013) and GT3/GT2
  – “Haswell is a graphics monster” ‘Semi Accurate’
  – ~2.5x as fast as HD 4000x for GT3 while keeping low power use

• Intel Skylake (2015)
  – Potentially a fully flexible graphics pipeline?
Traditional Botnet?

- PC sales are diminishing
- Market got bigger
  - Have to attack more systems
- Expensive?
  - Yes for recent and patched systems
    (the ones with better GPUs, generally)

### Table: Global personal computing device sales by OS

<table>
<thead>
<tr>
<th>Device</th>
<th>2008</th>
<th>2012</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADOBE READER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC OSX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANDROID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLASH OR JAVA BROWSER PLUG-INS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICROSOFT WORD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WINDOWS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIREFOX OR SAFARI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHROME OR INTERNET EXPLORER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0-days estimations per platform by “the Grugq” - Forbes
• Difference with classic botnet
  – Price
  – Potentially multiplatform
  – More difficult to detect
  – Different use
  – Limitations
Browser Based Botnet for Cracking

- Difference with classic browser-based botnet
  - More flexible, only one task
  - With which technologies?
  - To crack what?
BETTER BROWSER BASED BOTNETS
How to achieve this?

• Get permanent code execution in the browser
  – Find a way to have code running in clients
    • Find a server-side flaw
    • Make it persistent by poisoning the client’s cache
    • Spread
      – To other subdomains
      – To different layers
    • Keep it alive

• Compute data (password hashes, keys) with GPU
• Communicate with C&C servers
GET PERMANENT CODE EXECUTION IN THE BROWSER
Several ways to code execution

• Code execution of the web: XSS
• But sites are more secure now, right?
  – XSS is still overlooked
  – Still the most prevalent web vulnerability class
  – XSS vulns are still, most of the time, fixed individually
• Is it really cheap?
  – Can oftentimes be found with simple scanning tools
  – Not every new feature is thoroughly tested
  – Example
From WhiteHat Security Website Statistics Report
(https://www.whitehatsec.com/assets/WPstats_summer12_12th.pdf)
• XSS are fixed quickly, though
• Need to craft a permanent XSS for the client
  – Through cache poisoning
  – Leverage local storage features used by applications
    • HTML5 Web Storage feature
      – Stores data with no expiration date
      – Will not be deleted when the browser closes
      – Cannot be restricted to a specific path
    • Client-side DBs
    • Unified solutions
    • Browser extensions
- Files stored as objects literals
  ```javascript
  localStorage.setItem(key, value)
  localStorage.setItem('myFiles', JSON.stringify(files));
  ```

- Stores form or profile data
  - Can use (useless) client-side encryption

GibberishAES - client-side crypto used by jQuery.handleStorage

Example with Garlic

```javascript
$( '#form' ).garlic( {
  getPath: function ( $elem ) {
    return $elem.attr( 'id' );
  }
};
</script>
```
Which platforms and how much space do we get?

<table>
<thead>
<tr>
<th>Browser</th>
<th>Platform</th>
<th>Session Storage</th>
<th>Local Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>All</td>
<td>5MB</td>
<td>5MB</td>
</tr>
<tr>
<td>Firefox</td>
<td>All</td>
<td>unlimited</td>
<td>5MB*</td>
</tr>
<tr>
<td>Safari</td>
<td>OSX</td>
<td>Unlimited</td>
<td>5MB</td>
</tr>
<tr>
<td></td>
<td>iPhone</td>
<td>5MB</td>
<td>5MB</td>
</tr>
<tr>
<td>Internet Explorer 9</td>
<td>Win7</td>
<td>4.75MB</td>
<td>4.75MB*</td>
</tr>
<tr>
<td>Internet Explorer 10</td>
<td>Win8</td>
<td>4.75MB</td>
<td>4.75MB*</td>
</tr>
<tr>
<td>Android Browser</td>
<td>All</td>
<td>unlimited</td>
<td>5MB</td>
</tr>
</tbody>
</table>

Doug DePerry – HTML5 modern web browser perspective
Why?
- Easier to find XSS on weaker subdomains
- Poison cache of other, more used, more secured subdomains

Find a XSS on the weakest/newest subdomain of .bigcorp.com

It is common to use domain-wide cookies, but if not:

Overwriting cookies of another sub-domain

<table>
<thead>
<tr>
<th>Test description</th>
<th>MSIE6</th>
<th>MSIE7</th>
<th>MSIE8</th>
<th>FF2</th>
<th>FF3</th>
<th>Safari</th>
<th>Opera</th>
<th>Chrome</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordering of duplicate cookies with different scope</td>
<td>random</td>
<td>random</td>
<td>some dropped</td>
<td>some dropped</td>
<td>most specific first</td>
<td>random</td>
<td>most specific first</td>
<td>most specific first</td>
<td>by age</td>
</tr>
</tbody>
</table>

From the Browser Security Handbook (M. Zalewski)

Trigger specific XSS on other subdomains
- Easier to find as the cookies are “trusted”
- What if they use an anti-CSRF token in cookie+body?
• Via Header Injection (HTTP Response Splitting)
  – Overview
  – Commonly used files can then be poisoned for a domain
  – Code can execute when this file is used

• Increase the scale
  – Poison proxy server’s cache?
  – Poison the most common JS files
Staying alive?

• General misconceptions about JavaScript
• What can you find out about the current user?
  – And about what is going on in the browser?
• What else can it do?
  – In tabs / popups / windows
  – ..and?
• How much can be done in iFrames?
Staying alive?

Example
Other ways toward code execution

• Other technologies
• Code execution in Java and Flash?
  – More difficult to find
  – However, from another (compromised) domain:
    • Third party flash applications most of the time are allowed code execution in the main domain (‘allowscriptaccess’ set to ‘always’)
    • No one cares about the ‘unknown’ issuer for signed Java apps
• Cache poisoning
  – Flash LSOs
• Browser plugin
• **How?**
  - Buying Ads running a script
  - PPC - CPI

• Will run on another domain, iframed

• This iframe will run on popular websites

• Works well for harnessing GPU power

• Inconvenient
  - Can be expensive
  - ~cross-platform
  - ~persistence
COMMUNICATION
Bypassing same-origin policy

• Nothing new here but HTML5 made it easier
• Traditional way to bi-directional communication
  – Script tag
    • JSONP
  – Image tag (hack-ish)
• HTML5 way
  – Ajax with CORS (Cross-Origin Resource Sharing)
    • Allows Ajax calls to read+write on a domain authorizing it
  – WebSockets
    • Read+write over a persistent TCP socket
• Other (Flash etc.)
• Options
• Classic C&C architecture
  – Centralized
  – Hybrid P2P
  – Other?
• Distribution of passwords
  – List of ranges of passwords on public dictionaries
  – Ranges of characters
  – Keep track of every single client
USING THE GPU IN A BROWSER
• OpenGL ES 2.0 is used by:
  • WebGL
    – Embedded into JS
    – HTML5 Canvas tag
  • Flash
    – Since flash player 11
  • NaCL
• Based on OpenGL

• Use OpenGL Shading Language (GLSL)

• Can use DirectX 9 in Windows with ANGLE
  – Used by Chrome and Firefox
• How?
  – Using fragment shaders as a Hash function
    • Write to gl_FragColor
  – Store computations in a frame buffer object
    • Read with readPixels()

– But..
• Current version of **GLSL ES** in browsers
  – Similar to GLSL < 1.30

  – Only 16-bit integers!
    • Using a vector with 2 floats is slow

  – No bitwise operations!
    • ‘Reserved for future use’ in the specs
Does that look fast to you?

• Results:
  – Works but very slow
  – Hack-ish

• OpenGL ES 2.0 is very limited
  – But it is going to be way better in OpenGL ES 3.0

Fragment shader code for ..XOR..
OpenGL ES 3.0

• Official release of the standard in August 2012
  – Already officially supported in Intel Ivy Bridge

• New version of GLSL ES
  – Supports 32-bit integers
  – No limitations on bitwise operations
  – More portable
OpenGL ES - Cross platform?

- Windows and MacOS
- Mobile
  - Since Android 2.0/2.2
  - iOS
    - iPad
    - iPhone since 3GS
    - iPod Touch 3rd gen)
  - Blackberry since OS 7.0
  - Nokia and Samsung phones
  - Raspberry Pi, WebOS, Archos Internet tablet
- Consoles
  - Playstation 3
  - Nintendo 3DS
- Smart TVs
WebGL - Cross platform?

- All desktop web browsers
  - Except IE – obviously (but there is a plug-in, IEWebGL)
- Mobile
  - Android
    - Hopefully soon, there is a flag in Chrome beta
  - iOS
    - Internally supported, only available to iAd developers
    - Yes, iAd, to integrate ads to iPhone apps..
    - Disabled for the browser
  - Blackberry Playbook
  - Firefox for mobile
  - Opera Mobile
  - Nokia N900
- PS3
  - Rumored
  - Supports only flash 9 for now
• Created by the same company that created WebGL (Khronos)

www.khronos.org
• Javascript binding for OpenCL
  – Made for parallel computing using the GPU

  – OpenCL is what is used by most cracking apps

  – GPU drivers support OpenCL
WebCL

- Need a browser plug-in for now
  - Plug-ins available for Chrome and Firefox
    - Made by Nokia, Motorola and Samsung
  - Is likely to be ported to browsers
    - Is currently being implemented into Firefox
      (http://hg.mozilla.org/projects/webcl/)

- Results in the order of the two digits of MH/s with a decent GPU
  - Way faster than any other browser-based tech.
  - Would be faster if not running in a plugin
Other challenges

• Cracking has to be done when GPU is idle
  – Probe with a quick computation every X seconds
  – Can be run during the night

• Code is difficult to properly obfuscate
  – Easy to debug to see what is going on

• Bottleneck in the node management (C&C)
  – Nodes dying etc.
WHAT DOES THAT MEAN?
A lot of unknown to make proper statistics

How many clients could be compromised?
  
  - Depends on the targeted site
    
    - .bigsite.com could lead to millions
    - .popular-PC-game-site.com
      
      - Thousands of powerful PCs compromised
      - Less targeted, probably easier to find flaws

For how long?
  
  - If permanent code execution in the client, potentially a pretty long time if cache is never cleared

How to determine people’s GPU for stats?
### Gaming GPUs?

<table>
<thead>
<tr>
<th>ALL VIDEO CARDS</th>
<th>AUG</th>
<th>SEP</th>
<th>OCT</th>
<th>NOV</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel HD Graphics 3000</td>
<td>3.30%</td>
<td>3.55%</td>
<td>3.54%</td>
<td>3.38%</td>
<td>3.70% +0.32%</td>
</tr>
<tr>
<td>NVIDIA GeForce GTX 580 Ti</td>
<td>3.24%</td>
<td>3.28%</td>
<td>3.21%</td>
<td>3.38%</td>
<td>2.99% -0.39%</td>
</tr>
<tr>
<td>NVIDIA GeForce GTX 550 Ti</td>
<td>2.34%</td>
<td>2.49%</td>
<td>2.55%</td>
<td>2.57%</td>
<td>2.49% -0.08%</td>
</tr>
<tr>
<td>Intel HD Graphics</td>
<td>2.35%</td>
<td>2.28%</td>
<td>2.25%</td>
<td>2.03%</td>
<td>2.18% +0.15%</td>
</tr>
<tr>
<td>NVIDIA GeForce GTX 460</td>
<td>2.28%</td>
<td>2.24%</td>
<td>2.18%</td>
<td>2.40%</td>
<td>2.04% -0.36%</td>
</tr>
<tr>
<td>ATI Radeon HD 5770</td>
<td>2.13%</td>
<td>2.10%</td>
<td>1.97%</td>
<td>2.17%</td>
<td>1.90% -0.27%</td>
</tr>
<tr>
<td>Mobile Intel 4 Series Express</td>
<td>2.10%</td>
<td>1.78%</td>
<td>1.69%</td>
<td>1.57%</td>
<td>1.81% +0.24%</td>
</tr>
<tr>
<td>NVIDIA GeForce 9800</td>
<td>2.02%</td>
<td>1.98%</td>
<td>1.88%</td>
<td>1.90%</td>
<td>1.76% -0.14%</td>
</tr>
<tr>
<td>NVIDIA GeForce 9600</td>
<td>1.89%</td>
<td>1.88%</td>
<td>1.82%</td>
<td>1.77%</td>
<td>1.71% -0.06%</td>
</tr>
<tr>
<td>Intel HD Graphics 2000</td>
<td>1.28%</td>
<td>1.36%</td>
<td>1.41%</td>
<td>1.38%</td>
<td>1.70% +0.32%</td>
</tr>
<tr>
<td>NVIDIA GeForce GTX 550</td>
<td>1.79%</td>
<td>1.76%</td>
<td>1.60%</td>
<td>1.79%</td>
<td>1.69% -0.10%</td>
</tr>
<tr>
<td>NVIDIA GeForce GTS 450</td>
<td>1.65%</td>
<td>1.66%</td>
<td>1.73%</td>
<td>1.59%</td>
<td>1.59% -0.10%</td>
</tr>
</tbody>
</table>

Source: [http://store.steampowered.com/hwsurvey/videocard/](http://store.steampowered.com/hwsurvey/videocard/)
Let’s try to estimate for statistic purposes

- Standard but decent GPU today may get 20-50MH/s for WebCL and MD5 computations
- **Average** GPU in the future?
  - Including CPUs with ‘on-chip’ graphics
  - WebCL integrated in the browser will be faster too
- Will only talk about pure brute force
  - Password lists could obviously work better, depending on what is being cracked
• Let’s take a large estimate with 100k to 10M clients potentially compromised
  – Number of devices per person constantly increases
  – .majorSite.com with thousands or millions of users
    • Each user has X computer/devices

So..
Computing Hashes?

MD5

SHA-256

Number of clients (in k)

- 10
- 50
- 250
- 1250
- 2500
- 25000

# characters: 0-9; a-z; A-Z
Cracking Keys?
PBKDF2 SHA-256 and 1000 rounds

Number of clients (in k)
- 10
- 50
- 250
- 1250
- 5000
- 25000

# characters
spe;0-9; a-z;A-Z
Examples

• Example with 100k clients and cracking of MD5
  – 1000+ GH/s
    • On a larger scale: 1M clients would get 10,000 GH/s
  – Fastest FPGAs barely reach the hundreds of GH/s
    • ‘Only’ 10k clients to reach the power of an expensive FPGA
  – Amazon EC2, ads and exploits are expensive

• Example of complex 10 characters password with MD5
  – ~1day to find the password with 4M clients
  – $40k with Amazon
  – May only take an hour in 5 years
MASSIVE COMPUTING POWER, WHAT FOR?
• MD5?
  – Yes it is still used..

• SHA-256 is supposed to be safe to use
  – Depends how it is used

• Other
  – Rounds of hashes
  – Hashcash
  – Bitcoin

• bcrypt / scrypt
  – Not “really” crackable using these methods
  – Companies should use it more
    • Should also be aware of issues it can add (DoS)
• Symmetric
  – Password Based Key Derivation function (PBKDF2)
    • FIPS requires a minimum of 1000 iterations
  – Weak keys

• Asymmetric
  – RSA
    • <= 768-bit
  – DKIM
    • <= 768-bit
  – What about 1024-bit?
EXAMPLES
Examples

• Hash functions
  – Single round of a hash function for storing passwords
    • + not using a strong and unique salt
• DKIM
  – Spoofing emails
    • Z. Harris: lots of companies with 512-768-bit keys
• NTLM (LM)
On the phone: Poor Keyboards

Yi<Dz*ba1pWn
• Symmetric keys
  – Data encrypted with keys derived from a weak password
    • This is very common for local encryption
      – Both in servers and in clients
    • Password managers
    • Secure containers
CONCLUSION
• Using browser-based botnets can be very effective and cheap for cracking – but is not possible to fully exploit today
  – May be possible sooner than you may think

• OpenGL ES 3.0 and WebCL have not been integrated YET
  – OpenGL ES 3.0 may arrive soon
  – WebCL will definitely be needed in browsers at some point
    • There are plugins and it is already being implemented in Firefox

• In addition to introducing new issues, HTML5 also increases the severity of other web security issues
  – Companies should have a well defined security process to avoid being so vulnerable to the specific issues mentioned