Harnessing GP²Us
Building Better Browser Based Botnets

Marc Blanchou

iSECpartners
part of nccgroup

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

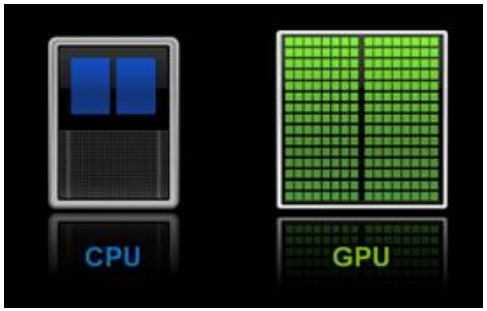
Who am I?

- 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

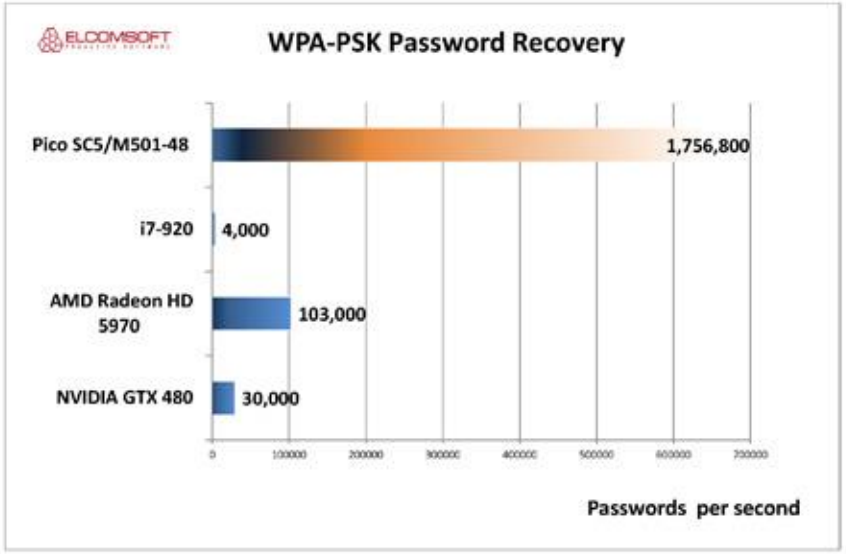
INTRODUCTION

Basics

- Cracking
  - General-purpose computing
  - Needs parallel computations
    - GPU vs CPU
- FPGA?



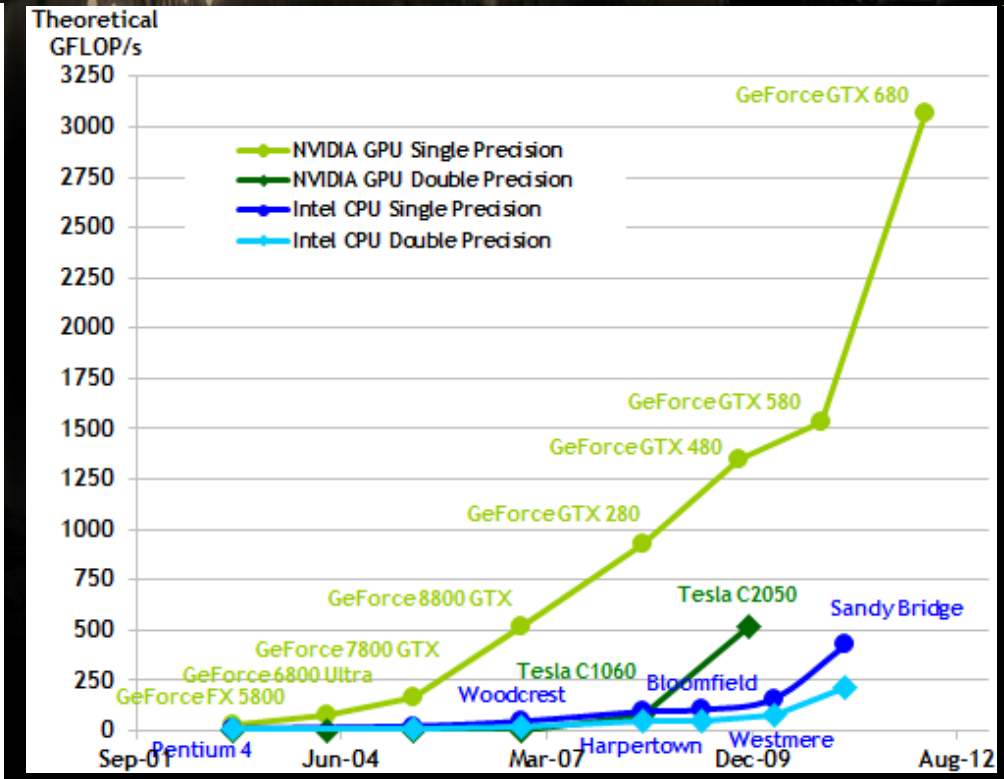
Nvidia.com



GPU Farm?

Evolution GPU-CPU

- GPU Parallelism is almost doubling every year
- Way faster evolution than CPU



NVidia CUDA/OpenCL C programming guide

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?

Botnet?

- 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

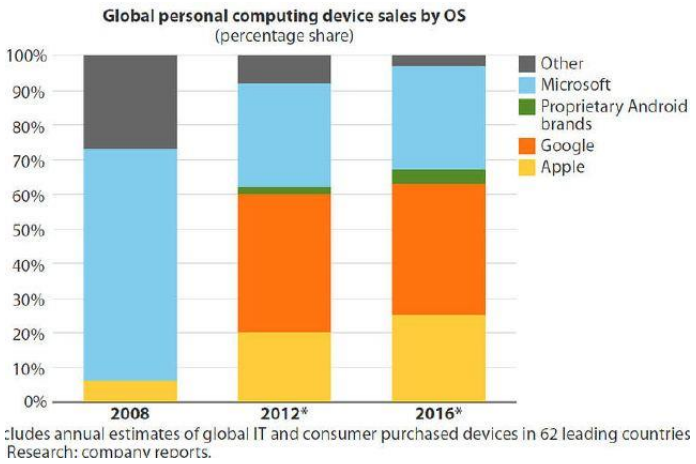
- New on-chip graphics on recent CPUs
- Intel Ivy Bridge (2011) and Intel HD 4000
  - Great support for recent techno
  - Relatively decent computing power
  - Low power consumption / heat – (discreet!)
- Intel Haswell (2013) and GT3/GT2
  - "Haswell is a graphics monster" *'Semi Accurate'*
  - ~2.5x 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)

ADOBE READER	$5,000–$30,000
MAC OSX	$20,000–$50,000
ANDROID	$30,000–$60,000
FLASH OR JAVA BROWSER PLUG-INS	$40,000–$100,000
MICROSOFT WORD	$50,000–$100,000
WINDOWS	$60,000–$120,000
FIREFOX OR SAFARI	$60,000–$150,000
CHROME OR INTERNET EXPLORER	$80,000–$200,000
IOS	$100,000–$250,000

0-days estimations per platform by "the Grugq" - Forbes

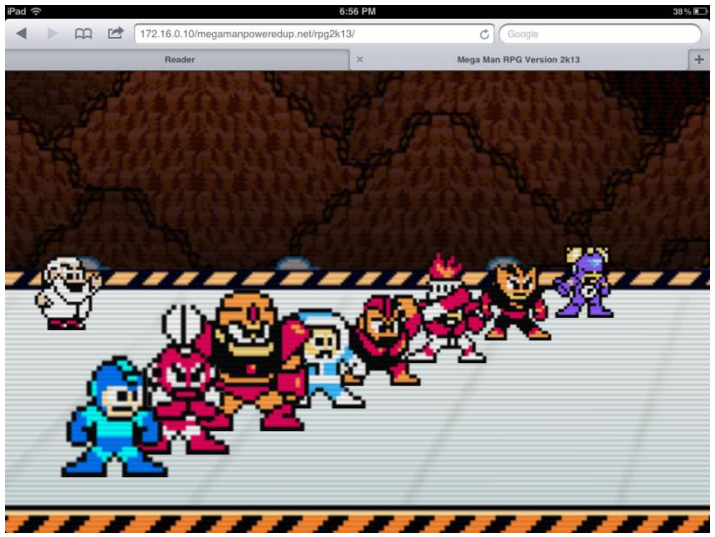


Browser Based Botnet

- 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

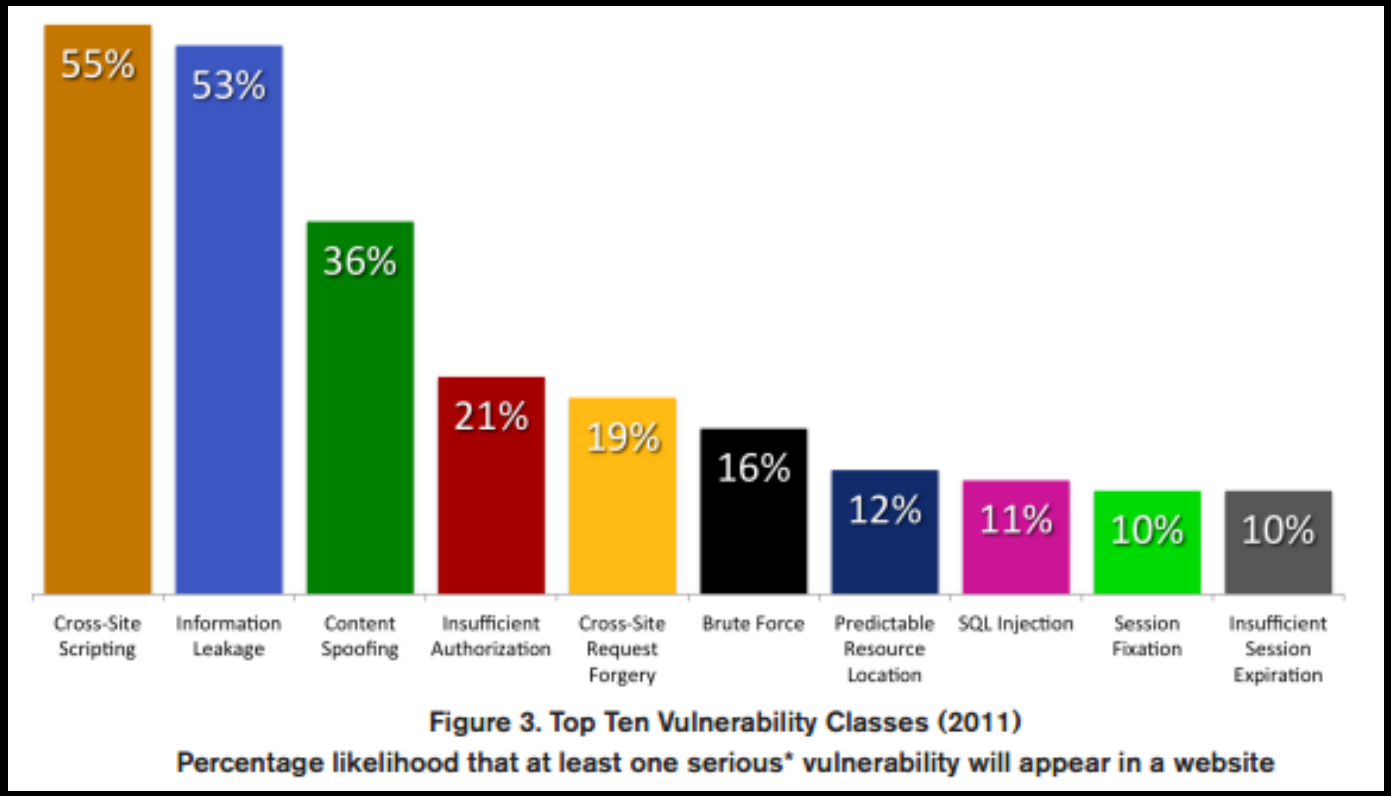


Figure 3. Top Ten Vulnerability Classes (2011)

Percentage likelihood that at least one serious* vulnerability will appear in a website

From WhiteHat Security Website Statistics Report (https://www.whitehatsec.com/assets/WPstats_summer12_12th.pdf)

Permanent code execution?

- 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

Example?

- Files stored as objects literals

```
localStorage.setItem(key, value)
localStorage.setItem('myFiles', JSON.stringify(files));
```

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

```
<script type="text/javascript">
  $( '#form' ).garlic( {
      getPath: function ( $elem ) {
          return $elem.attr( 'id' );
      }
  } );
</script>
```

Example with Garlic

```
enc = GibberishAES.enc("This sentence is super secret", "ultra-strong-password");
alert(enc);
GibberishAES.dec(enc, "ultra-strong-password");

// Now change size to 128 bits
GibberishAES.size(128);
enc = GibberishAES.enc("This sentence is not so secret", "1234");
GibberishAES.dec(enc, "1234");
```

GibberishAES - client-side crypto used by jQuery.handleStorage

Which platforms and how much space do we get?

Browser	Platform	Session Storage	Local Storage
Chrome	All	5MB	5MB
Firefox	All	unlimited	5MB*
Safari	OSX	Unlimited	5MB
	iPhone	5MB	5MB
Internet Explorer 9	Win7	4.75MB	4.75MB*
Internet Explorer 10	Win8	4.75MB	4.75MB*
Android Browser	All	unlimited	5MB

Doug DePerry – HTML5 modern web browser perspective

Spreading to other subdomains

- 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

Test description	MSIE6	MSIE7	MSIE8	FF2	FF3	Safari	Opera	Chrome	Android
Ordering of duplicate cookies with different scope	random	random	some dropped	some dropped	most specific first	random	most specific first	most specific first	by age

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?

Spreading further

- 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?

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

Ads

- 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

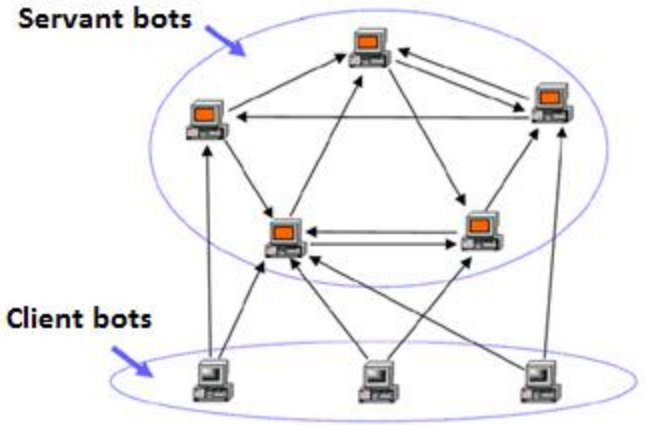
COMMUNICATION

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.)

C&C

- 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

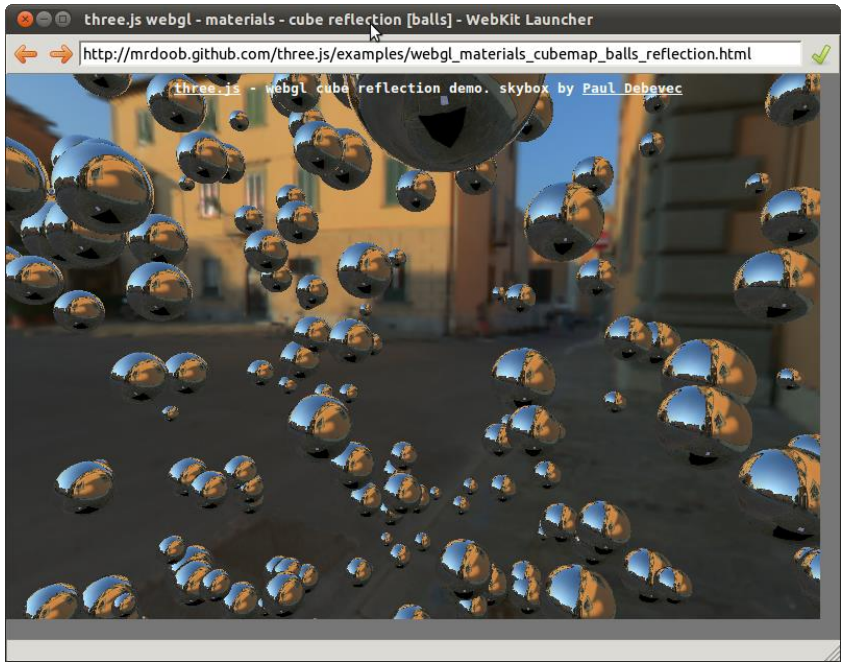


Example (static.usenix.org)

USING THE GPU IN A BROWSER

GPU in the browser

- OpenGL ES 2.0 is used by:
  - WebGL
    - Embedded into JS
    - HTML5 Canvas tag
  - Flash
    - Since flash player 11
  - NaCL

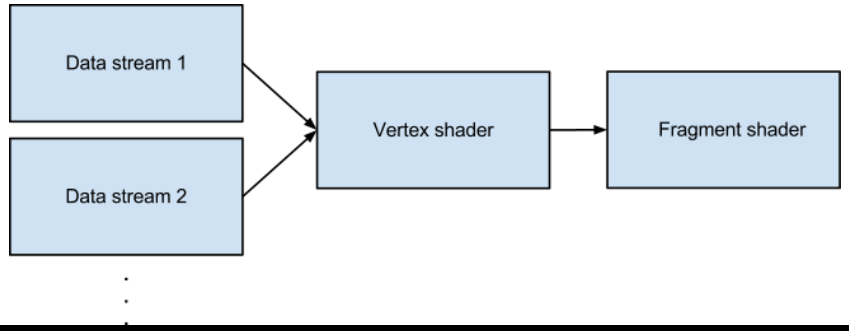


Open GL ES

- Based on OpenGL
- Use OpenGL Shading Language (GLSL)
- Can use DirectX 9 in Windows with ANGLE
  - Used by Chrome and Firefox

What about General Purpose Computing?

- How?
  - Using fragment shaders as a Hash function
    - Write to gl_FragColor
  - Store computations in a frame buffer object
    - Read with readPixels()
  - But..



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?

```
float _XOR(float a, float b)
{
    float result = 0.0;
    float n = 32768.0;
    // 0x8000 to operate on only 16-bits

    while (n > 0)
    {
        if (a >= n || b >= n)
        {
            if (a < n || b < n)
                result += n;
            if (a >= n)
                a -= n;
            if (b >= n)
                b -= n;
            n /= 2.0;
        }
    }
    return result;
}

vec2 XOR(const vec2 a, const vec2 b)
{
    return vec2(_XOR(a.x, b.x), _XOR(a.y, b.y));
}
```

Fragment shader code for

..XOR

- 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

OpenGL ES 3.0

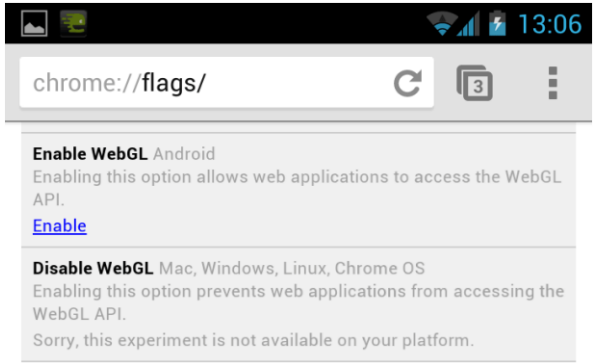
- Official release of the standard in August 2012
  - Already officially supported in Intel Ivy Bridge
- New version of GLSL ES
  - Supports 32-bit integers
  - No limitations on bitwise operations
  - More portable

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

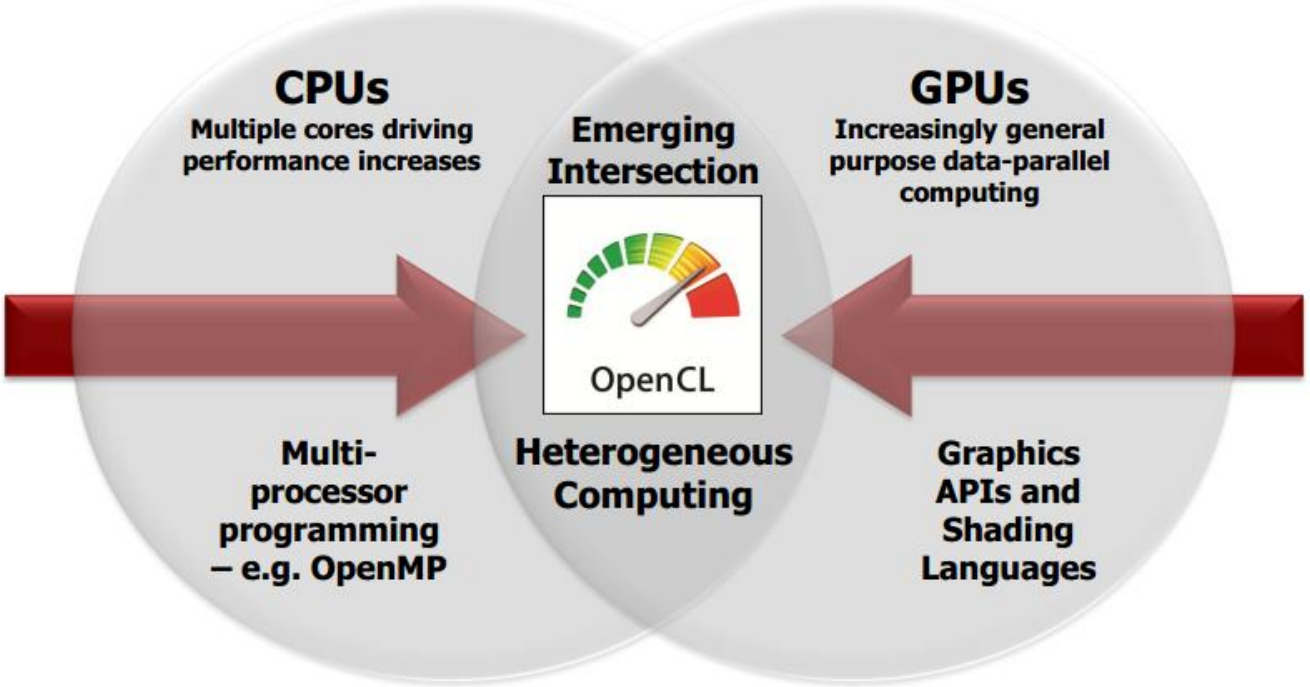
WebGL - Cross platform?

- 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



OpenCL

- Created by the same company that created WebGL (Khronos)



www.khronos.org

WebCL

- 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?

So what?

- 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?

ALL VIDEO CARDS	AUG	SEP	OCT	NOV	DEC	
Intel HD Graphics 3000	3.30%	3.55%	3.54%	3.38%	3.70%	+0.32%
NVIDIA GeForce GTX 560 Ti	3.24%	3.28%	3.21%	3.38%	2.99%	-0.39%
NVIDIA GeForce GTX 550 Ti	2.34%	2.49%	2.55%	2.57%	2.49%	-0.08%
Intel HD Graphics	2.35%	2.28%	2.25%	2.03%	2.18%	+0.15%
NVIDIA GeForce GTX 460	2.28%	2.24%	2.18%	2.40%	2.04%	-0.36%
ATI Radeon HD 5770	2.13%	2.10%	1.97%	2.17%	1.90%	-0.27%
Mobile Intel 4 Series Express	2.10%	1.78%	1.69%	1.57%	1.81%	+0.24%
NVIDIA GeForce 9800	2.02%	1.98%	1.88%	1.90%	1.76%	-0.14%
NVIDIA GeForce 9600	1.89%	1.88%	1.82%	1.77%	1.71%	-0.06%
Intel HD Graphics 2000	1.28%	1.36%	1.41%	1.38%	1.70%	+0.32%
NVIDIA GeForce GTX 560	1.79%	1.78%	1.80%	1.79%	1.69%	-0.10%
NVIDIA GeForce GTS 450	1.65%	1.66%	1.73%	1.69%	1.59%	-0.10%

http://store.steampowered.com/hwsurvey/videocard/

GPUs in the future?

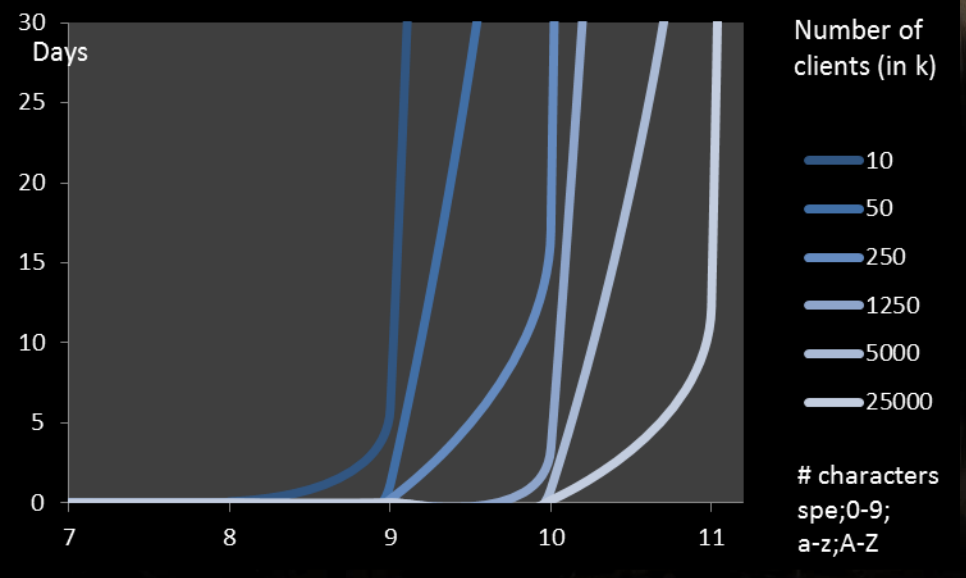
- Let's try to estimate for statistic purposes
  - Standard but decent GPU today may get 20-50MH/s for WebCL and MD5 computations
  - Average GPU in the future?
    - Including CPUs with 'on-chip' graphics
    - WebCL integrated in the browser will be faster too
  - Will only talk about pure bruteforce
    - Password lists could obviously work better, depending on what is being cracked

Number of devices per person?

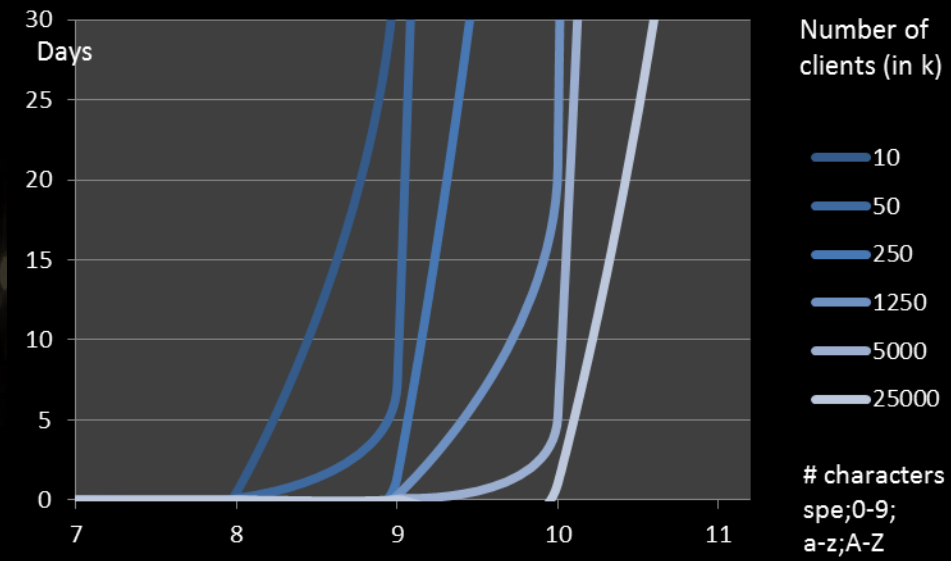
- Let's take a large estimate with 100k to 10M clients potentially compromised
  - Number of devices per person constantly increases
  - .majorSite.com with thousands or millions of users
    - Each user has X computer/devices

So..

Computing Hashes?

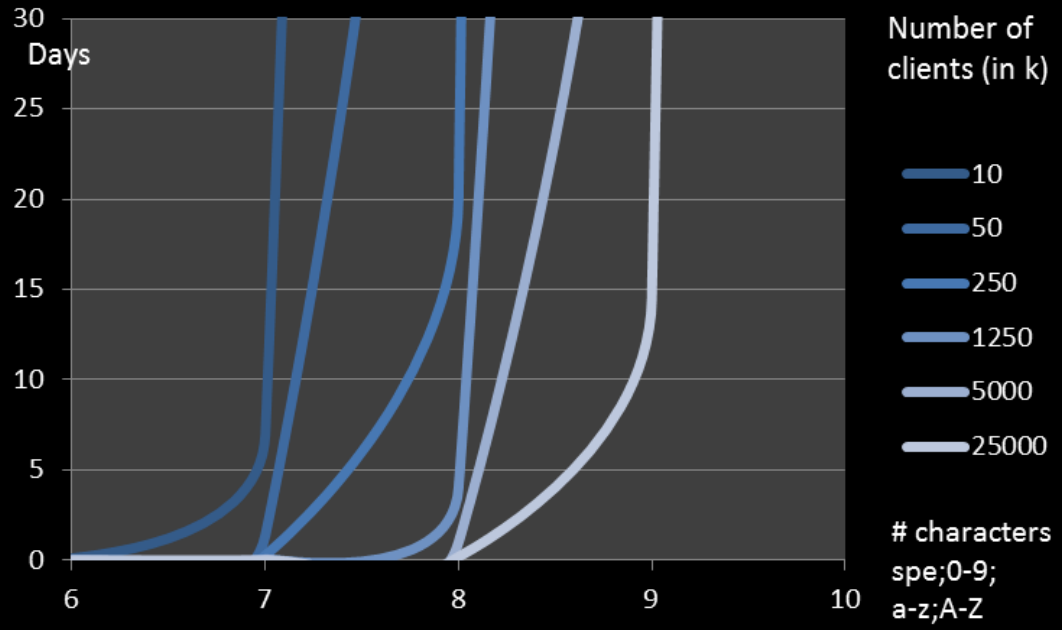


MD5



SHA-256

Cracking Keys?
PBKDF2 SHA-256 and 1000 rounds



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?

Hashes?

- 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)

Keys?

- 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

Examples

- 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

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

QUESTIONS?

Marc Blanchou
marc@isecpartners.com