Global Information Assurance Certification Paper

Cross-Site Tracing – Protecting Businesses from a Simple Attack
GIAC Security Essentials (GSEC)
Practical Assignment, Version 1.4b, Option 1
June 1, 2003
Cheryl Stephens

Abstract

Businesses and corporations are beginning to use web-based applications for their core business functions. By using these applications, organizations become more vulnerable to malicious attacks from customers, partners, internal staff members or any other outside individual interested in gaining access to their data. As organizations begin to use these applications, they also need to understand and recognize the application is not the only threat to their business but also the web server and that this needs to be taken into consideration when purchasing a web-based application. One of the newest published cross-site scripting attacks (XSS), cross-site tracing (XST) bypasses any security mechanism put into place by a developer and enables an attacker to gain access to an individual's cookies and authentication credential information via a simple client-side script. In this paper, I will discuss how easy cross-site tracing could effect an organization and how an organization can protect itself from this type of attack.

Introduction

Every day organizations are inundated with offers to purchase business applications to help cut costs, to make communication between customers, suppliers and partners easier, and to provide employees access to company tools and databases. The ability to access a business twenty-four hours a day seven days a week is a popular feature of most business enterprise applications. However, in order to provide many of these services, organizations are generally required to use some sort of web-based solution. Although web-based solutions give organizations the flexibility to perform, daily business transactions from anywhere at anytime, they also pose a serious threat to the business's overall reputation and infrastructure. As organizations begin utilizing and relying on web-based applications to carry out their day to day activities, many will learn and have learned that securing the web server is as important as securing the application.

Most web-based business applications require the use of a common web browser, a way to authenticate and a method to keep a session open while making some type of transaction. Web applications generally utilize the HTTP protocol, which is a stateless connection between the client and the server. The client makes the connection, gets the data it needs and disconnects. If someone wanted to perform a transaction that needed to be opened longer, like banking,

shopping, or any other e-commerce type transaction, the application would need to be able to maintain "state". To maintain state, application developers generally use cookies and/or session ids. A cookie is a simple text file that is written to the user's computer and often contains some information about the user to make their browsing experience more enjoyable. With a cookie, the client and server are able to maintain a connection, generally called a session, between each other.

Since cookies are simple text files with information about the user, like user id, password and sometimes even credit card information, accessing this information used to be somewhat easy for a hacker at any level. However, to access this information required some ingenuity and a script created by the hacker. The hacker needed some way to get an unsuspecting user to execute a script either by clicking a link, mousing over an image, or opening an e-mail. The script would send the data to a potential hacker and they would use this data to do their thing like hijack a session or impersonate the victim. It has become harder and harder for hackers to access cookies with the adoption of security and privacy policies and average user's knowledge of browser settings changing, but attacks still happen and will continue to happen.

By design most web applications require users to have scripting enabled in their browsers and to accept the cookies in order to complete a transaction. Because of this type of requirement and the way applications are developed, hackers have the ability to exploit known bugs in applications to get cookies and session information. So how does an organization protect the privacy of employees, customers and themselves while at the same time making their content readily available to anyone and still maintain “state” using cookies? Microsoft came up with a solution and implemented it in IE 6.0 SP1, the httpOnly cookie attribute.

HTTP-only cookies, are cookies that have the httpOnly attribute inserted into the cookie. This cookie eliminates the ability of any client-side script from accessing a user’s cookies. The following is an example of the httpOnly cookie format as presented by Microsoft.

Set-Cookie: USER=123; expires=Wednesday, 09-Nov-99 23:12:40 GMT; HttpOnly (Microsoft)

The httpOnly cookie was implemented to help prevent hackers from capturing cookie data through cross-site scripting. Before the httpOnly cookie an attacker would generally write a script to call the "document.cookie" object in order to obtain the cookie data. With the httponly cookie the data returned to a potential hacker and to the web browser itself is blank. Like any other mitigation tool or technique someone found a clever way to bypass the httpOnly cookie using cross-site tracing.

Cross-site tracing bypasses the web application all together and takes advantage of the web server’s HTTP TRACE method. By using the HTTP 1.1 TRACE method, the http functions of the browser (XMLHTTP for IE) and (XMLDOM for

Mozilla) an attacker can write a basic script to access a user's cookies and a user's basic authentication information. Before the news of cross-site tracing, capturing a user's basic authentication credentials was generally done through packet sniffers. To better understand how cross-site tracing works, one must first understand both TRACE and cross-site scripting.

What is TRACE?

TRACE isn't a new concept or function of a web server that was just discovered. TRACE has been around as long as the protocol Http 1.1 has been around. HTTP TRACE is a function of the http 1.1 protocol. It is used to echo back to the client the exact response header information sent to the web-server from the client. TRACE was originally implemented as a debugging and analysis tool for developers and administrators. Any web server that supports the Http 1.1 protocol as defined by (RFC 2616, section 9.8) is vulnerable to a trace attack.

The following taken from Cert® Vulnerability Note VU#867593

> 9.8 TRACE
>
> The TRACE method is used to invoke a remote, application-layer loopback of the request message. The final recipient of the request SHOULD reflect the message received back to the client as the entity-body of a 200 (OK) response.
>
> …
>
> If the request is valid, the response SHOULD contain the entire request message in the entity-body, with a Content-Type of "message/http". Responses to this method MUST NOT be cached. (Cert® Coordination Center)

Web servers such as Apache, IIS and iPlanet do support http 1.1 and TRACE. If you are not sure check with your vendor or run the script in example-1. Since TRACE is a method like GET and POST it is by default part of the web server's configuration and usually never disabled by most administrators. Most IT professionals, who work with web servers have not seen any need to disable this feature, many probably never knew about the existence and others never knew of the undesirable impact this could have on potential applications and their users. However, by utilizing the features of current web browser http support, a client-side scripting language, such as javascript, vbscript, flash, etc and TRACE a potential hacker can read the headers of a third party request. By accessing the header information, a hacker can access all cookies; session ids and basic authentication information for a user set by a domain. TRACE bypasses the new httpOnly cookie option implemented by Microsoft to prevent cookie stealing and it bypasses SSL support, a hacker can still access secure cookie data and authentication information. The data communication between server and client is encrypted, but the basic authentication information is encoded and not encrypted thus making it easier for the password to be cracked. So, even setting a cookie

to secure, does not prevent TRACE from showing the cookie information and using basic password protection with a secure server does not ensure security.

TRACE is limited by browser security restrictions, allowing only connections to the domain hosting the script. However, since cross-site tracing or XST is a client-side threat and limited to these restrictions, a hacker can access any domain using the script from their desktops or the desktops of their victims. This immediately becomes a problem for businesses because a malicious attacker could perform a cross-site scripting attack against anyone, making everyone vulnerable. There have been several advanced scripts written to bypass the domain restriction issue. Because cross-site tracing is a client-based issue with domain based restrictions, the potential for an internal cross-site scripting attack increases, thus making internal web applications more vulnerable to attacks by disgruntled employees, customers or partners.

The following example is a simple script by Jeremiah Grossman written to show how easily one can gain access to the cookie data of a potential victim. Using this script and modifying the URL, I was able to gain access to my authentication information on our corporate extranet.

```
function sendTrace() {
        var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
        xmlHttp.open("TRACE", "https://www.foobar.com/extranet/",false);
        xmlHttp.send();
        xmlDoc=xmlHttp.responseText;
        alert(xmlDoc);
}

<input type=button OnClick="sendTrace();" value="Send Trace Request">
```

Example: 1

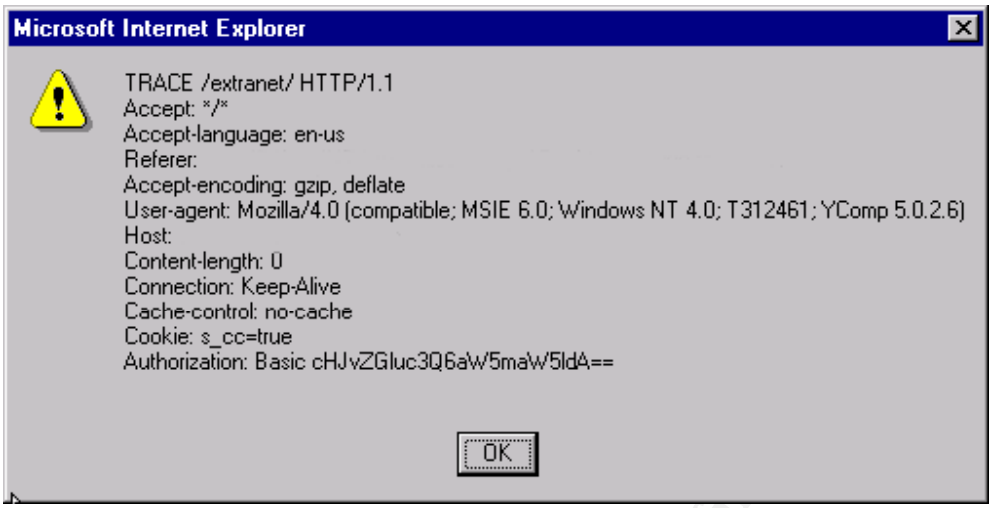

Figure 1 - Results of TRACE response from the /extranet server

By using the above script or a similar script written in another scripting language, a hacker can bypass the httpOnly cookie attribute and access both the cookie data and any basic authentication information of the user. In figure 1 notice the Authorization: Basic field, TRACE was able to capture this data using a secure connection. Because TRACE is a function of the HTTP protocol, it disregards the security of the web server echoing back the content of the headers. If a hacker can bypass the web application altogether using TRACE then cross-site tracing can be considered the next big cross-site scripting exploit and every measure should be taken to prevent this type of attack.

What is Cross-Site Scripting?

Cross-site tracing is another variation of cross-site scripting. Cross-Site Scripting or XSS is an event that occurs when a web application receives bad code or malicious data that has not been validated either through client-side validation or server side validation. The most popular ways of gathering bad data or code are through html forms on a web site, through hyperlinks sent by e-mail or on a web page by mousing over images and web forums. The attacker uses the data captured often times to create a hyperlink with a script inserted into the URL, for example sports.domain.com/login.asp?<script>badscript</script>. The attacker will generally encode the script and will send the link to a potential victim. The user clicks on the link and goes to a familiar page and at the same time sends back to the attacker the victim's personal information. A cross-site scripting attack like this enables a hacker to gather sensitive information from a potential victim. By using the information captured from a cross-site tracing attack a potential hacker could write another script and perform different types of cross-site scripting attacks to:

- hijack the victim's session
- change the victim's browser settings
- capture a victims cookie

- poison a victims cookies

Putting It All Together

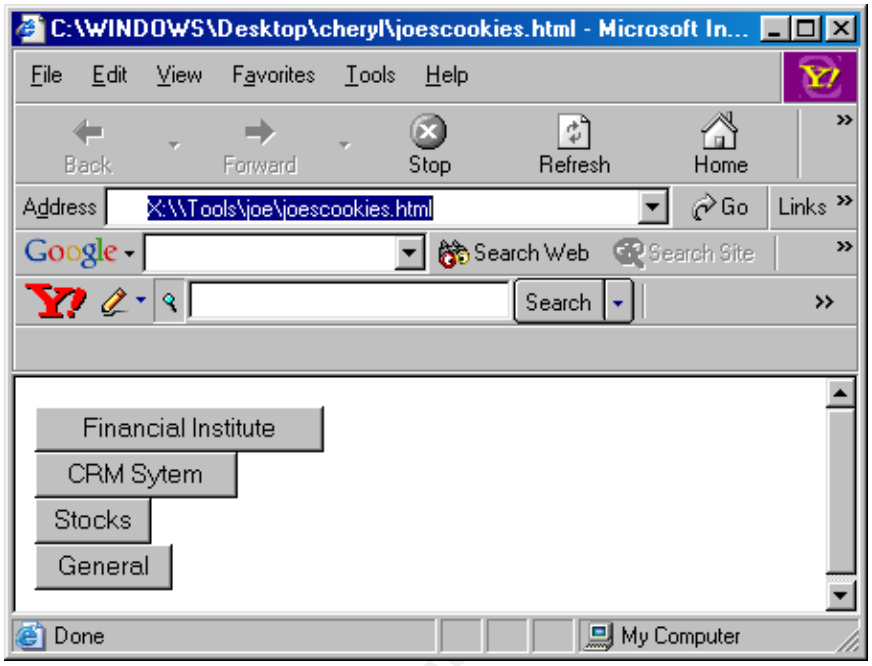

Figure 2 - Joe's web page

The following scenario illustrates how easy it is to perform a cross-site trace. Although the victim in this scenario is the CEO of a manufacturing company, it could be similar to many organizations.

Every organization has an average Joe who dabbles with computing and the Internet as a hobby. Joe is one of the IT departments support people. He has created a simple web page (Figure 2) to test this script (Figure 1) he found on the Internet. According to the information on the Internet he could access cookie information and most importantly authentication information from any site as long as he changed the URL. He made the modifications and was able to access his own information, now he wanted to access others information.

The other day the administrative assistant had been talking to Joe about her job and during the conversation casually mentioned the name of the CEO's bank. Joe listened and noted what she had told him. With this information, he edited his web page and added the URL of the CEO's bank to his new script. He'd like to not only see if he could access the CEO's financial information with his script but also log into the company's customer database with the CEO's account. He'd love to be able to read the comments from the CEO in regards to several customers and maybe even make some notes of his own or edit notes the CEO had written. Joe isn't strong in the coding department so in order to run his script, he needs a reason to gain access to the CEO's computer. Luckily, the CEO needs help with his e-mail program. Joe, goes to the CEO's office to work

on his computer, he has not only fixed the mail problem, he also notices the CEO has left several web browsers open and one of those happens to be the CEO's bank and the company's customer database. Joe decides to run his script to capture all of the CEO's cookie data and authentication information. The CEO never noticed the pen drive dangling from Joe's key ring. Joe inserts the pen drive into the USB port, captures the data and heads back to his desk. He is now able to look at this information and create an automated script to let him know when the CEO has logged into his bank or the customer service database. As soon as the CEO logs in, Joe could hijack his session and take on the CEO's identity. Today, Joe was just satisfied that he was able to retrieve the data. He had no intentions to actually log into either account. However, he did want to log into the customer database using some of his fellow co-workers usernames. Now that he knew how easy it was to access the information he set off to write a script that he could send through e-mail and have the data posted back to a file on the company web server.

How does this affect your Business?

Most businesses have a regular Joe who trouble-shoots, repairs and updates the company's personal computers. The scenario of stealing the boss's cookies may seem a little far-fetched but without appropriate security policies and knowledgeable staff, this type of attack could happen to anyone. For the CEO it could have been a loss of money in his bank account, if it had been the company it could have been a loss of credibility with customers and a major loss of revenue. Although this was a simplistic cross-site tracing attack, it could have compromised the reputation and credibility of the organization by the loss of confidentiality, the loss of integrity and the loss of availability, which are the "three basic goals of network security." (Cole, p60)

Using the data captured, or data like it Joe would be able to access a customer's account, this would be an example of a *loss of confidentiality*. With access to a customer's account, Joe could modify any record within the account and these modifications would appear as if the customer had made them, this would be a *loss of integrity*. While in the customer's account Joe could change the user's password or lock the account, this would be a *loss of availability*. In a short period, Joe would have compromised the "three basic goals of security, confidentiality, integrity and availability." (Cole, p60) Although, there are other ways to access a person's data, this is an example of how a cross-site tracing attack could impact an organization and compromise the integrity and reputation of the organization. It is important to understand how and why an organization should protect them selves from this type of simple attack. Even a regular Joe can perform this attack by copying the script and making a few minor changes.

Result of Cross-Site Tracing	
Loss of Confidentiality	• Unauthorized access • Identity theft • Impersonation
Loss of Integrity	• Data Corruption • Misinformation • Unauthorized modifications to data or files
Loss of Availability	• Denial of Service • No Access to account

(Microsoft 2002, p12)

"Pound of Prevention worth a $$$"

It is NOT OK for the security community to be aware of vulnerabilities such as XST and brush it off like it is yesterday's news. Organizations are becoming more Internet savvy and anything that will effect a company's reputation or contributes to a loss of revenue, should be considered harmful. To protect an organization and their users from a cross-site tracing attack, an organization can perform the following:

- Disable TRACE on all production web servers
- Update all employee browsers to the latest version and apply any new patches on a regular basis.
- Maintain consistent browser settings for each employee.
- Educate both staff and customers on how to use the application safely.
- Teach staff to ask questions of anyone using their computers or making changes.
- Identify a staff member who has the authority to send mass e-mails to customers, employees and partners.
- The company security policy should include the appropriate way to represent the company so that any malicious e-mail sent out could be easily identifiable.
- Keep abreast on Internet technology, web-browsers, web servers and how to secure them.
- When purchasing new applications or upgrades for your business, identify a staff member to help with identifying security requirements. Most importantly someone to help with asking the right questions.
- Learn to ask the right questions!
- Educate, Educate, Educate

How to disable TRACE on common Web Servers

All of the methods for disabling web servers have been copied. I have not tested the code or configurations. Whichever method you choose to use you should always contact the vendor first. At this point they may have already documented a solution or written a patch to address this particular issue.

Apache Web Server
Use the Apache mod_rewrite module to disable the HTTP TRACE requests.

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .* - [F]
```

(Cert® Coordination Center)

iPlanet Web Server
The iPlanet web server seems to be much harder to disable TRACE. Several people have offered solutions. I have not tested any of these solutions and cannot guarantee if they work or do not work. All of the code has been copied directly from their original sources. I copied the first bit of code from Jeremiah Grossman; however,I did edit one line of his original code, I changed the [emacs libnc-httpd40.so to [emacs libns-httpd40.so] .

```
cd ${IPLANET_ROOT}
mkdir secure_lib
cp bin/https/lib/libns-httpd40.so secure_lib
cd secure_lib
emacs  libns-httpd40.so
```

- Iplanets supported methods will look like this:HEAD^@GET^@POST^@DELETE^@^TRACE^@OPTIONS^@MOVE^@INDEX^@MKDIR^@RMDIR
- In this file there are multiple lists find all the instances of this list and edit each like so: change the method TRACE or any other method to 'GET ' add spaces to match the length of the word, this example added two spaces.
- Edit the start script for the web server to protect and prepend the secure_lib at the front of the LD_LIBRARY_PATH. For example LD_LIBRARY_PATH=${IPLANET_ROOT}/secure_lib:<The rest of the line>.
- Restart the web server.

(Grossman)

Peter Watkins posted the following code on bugtraq. Although he was unable to disable TRACE he said “I wrote this code to prevent TRACE from echoing certain headers.”

----------BEGIN CODE----------

```
#include "nsapi.h" /* NSAPI definitions */

/*
  PW-strip-trace.so
  NSAPI SAF to prevent the TRACE method from echoing Authorization
  or Cookie headers back to the HTTP client to prevent possible
  session hijacking or other XST information theft; see
  http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf

  Usage:
  At the beginning of obj.conf:
    Init fn=load-modules shlib=PW_strip_trace.so funcs=PW-strip-trace
  Inside an object in obj.conf (before other NameTrans calls, preferably
  at the top of the Default object stanza)
    NameTrans fn=PW-strip-trace
 */

NSAPI_PUBLIC int PW_strip_trace(pblock *pb, Session *sn, Request *rq)
{
    /* working variables */
    char *requestMethod = pblock_findval("method", rq->reqpb);

    /* bail out if we've got nothing to work with */
    if (!requestMethod) {
        return REQ_NOACTION;
    }

    if (strcmp(requestMethod,"TRACE") == 0 ) {
        /* remove the cookie & authorization headers so we don't echo them */
        param_free(pblock_remove("cookie", rq->headers));
        param_free(pblock_remove("authorization", rq->headers));
    }

    /* all done */
    return REQ_NOACTION;
}
```

---------- END CODE----------

Microsoft IIS Webserver

To disable TRACE for IIS you will need to use the URLScan tool, which is available through Microsoft. You can configure URLScan tool to filter the TRACE method or any other type of http requests.

Microsoft Knowledge Base articles specifically for installing and configuring the URLScan tool.

307608 Info: Using URLScan on IIS
326444 HOW TO: Configure the URLScan Tool
309394 HOW TO: Use URLScan with FrontPage 2000
318290 HOW TO: Use URLScan with FrontPage 2002
313131 HOW TO: Use URLScan with Exchange Outlook Web Access in Exchange Server 5.5
815155 HOW TO: Configure URLScan to Protect ASP.NET Web Applications

Conclusion

Although the method and the ability to perform a cross-site tracing attack has been around for a while, the knowledge to access a user's credentials using TRACE and a client-side script to bypass SSL had not been identified. There is a lot of debate on the discussion boards about whether TRACE should be disabled or if cross-site tracing is truly something to worry about when securing applications. Because organizations are beginning to rely on web-based transactions for their core business operations, it should be the responsibility of the security community to ensure that ALL organizations and businesses understand the need to not only ensure applications are secure but also verify the security of the web server. As consumers and employees of businesses that offer these services we should all have a vested interest in ensuring that the applications hosted in our environments and used on a daily basis are secure, whether the security measures to secure them seem to be trivial or not. In the end it is the reputation of the company that is at stake. To disable or not disable TRACE is simply up to the organization.

References

admin@cgisecurity.com, "The Cross Site Scripting FAQ" (July 2002) URL: http://www.cgisecurity.com/articles/xss-faq.txt

Cert/CC, Vulnerability Note VU#867593, "Multiple vendor's web servers enable HTTP TRACE method by default" (January 2003) URL: http://www.kb.cert.org/vuls/id/867593

Cole, Eric. Hackers Beware. Indianapolis: New Riders, 2002.

Endler, David, "The Evolution of Cross-Site Scripting Attacks." idefense (May 2002) URL: http://www.idefense.com/papers.html

Grossman, Jeremiah, "Cross-Site Tracing (XST): The New Techniques and Emerging Threats to Bypass Current Web Security Measures Using Trace and XSS." (January 2003) URL: http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf

Microsoft, "Mitigating Cross-site Scripting With HTTP-Only Cookies" URL: http://msdn.microsoft.com/workshop/author/dhtml/httponly_cookies.asp?frame=true

Microsoft. Security Operations for Microsoft Windows 2000 Server. Microsoft Corporation, 2002.

Spi Dynamics, "Cross-Site Scripting: Are your web applications vulnerable?", (2003) URL: http://www.idefense.com/papers.html

Watkins, Peter, peterw@usa.net, URL: http://cert.uni-stuttgart.de/archive/bugtraq/2003/01/msg00238.html, (January 2003).

Worthington, David "Web Vulnerability Puts Internet Users, Sites At Risk" ExtremeTech, (January 2003) URL: http://www.extremetech.com/print_article/0,3998,a=35995,00.asp