Hewlett Packard Enterprise

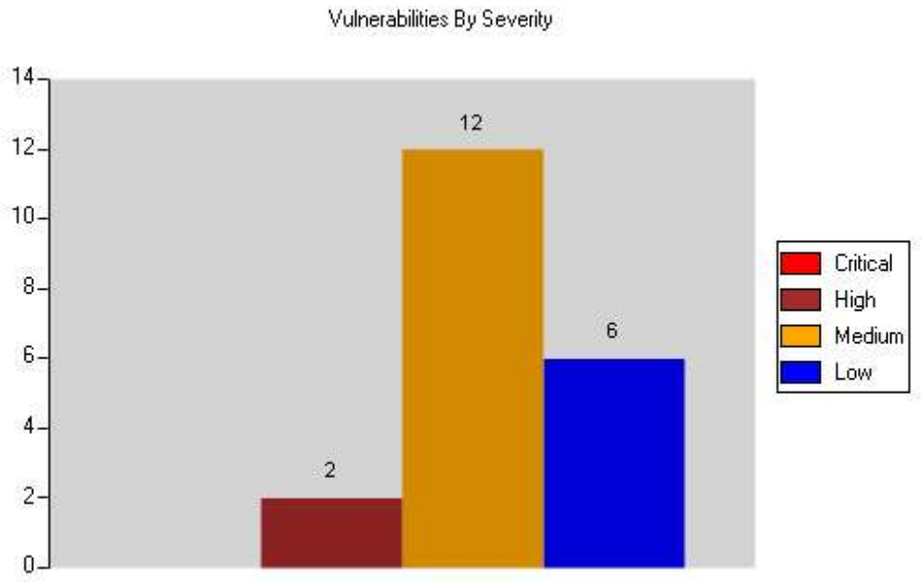
HPE Fortify WebInspect

Multiple Reports

Web Application Assessment Report

Scan Name:	Site: http://10.67.67.107:8980/cs/Logon/Logon.xhtml		
Policy:	Standard	**Crawl Sessions:**	181
Scan Date:	1/8/2019 11:07:07 AM	**Vulnerabilities:**	20
Scan Version:	17.10.283.0	**Scan Duration:**	16 minutes : 25 seconds
Scan Type:	Site	**Client:**	FF

Server: http://10.67.67.107:8980



High **Insecure Transport**

Summary:

Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. http://10.67.67.107:8980/cs/Logon/Logon.xhtml has failed this policy. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication:

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix:

For Security Operations:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For Development:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For QA:
Test the application not only from the perspective of a normal user, but also from the perspective of a malicious one.

File Names:
- http://10.67.67.107:8980/cs/Logon/Logon.xhtml

High **Often Misused: Login**

Summary:

An unencrypted login form has been discovered. Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. If the login form is being served over SSL, the page that the form is being submitted to MUST be accessed over SSL. Every link/URL present on that page (not just the form action) needs to be served over HTTPS. This will prevent Man-in-the-Middle attacks on the login form. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication:

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix:

Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Reference:

Advisory:http://www.kb.cert.org/vuls/id/466433

File Names:

- http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Medium **Poor Error Handling: Unhandled Exception**

Summary:

Unhandled exceptions are circumstances in which the application has received user input that it did not expect and doesn't know how to deal with. In many cases, an attacker can leverage the conditions that cause these errors in order to gain unauthorized access to the system. Recommendations include designing and adding consistent error-handling mechanisms that are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Implication:

Exception error messages may contain the location of the file in which the offending function is located. This may disclose the webroot's absolute path as well as give the attacker the location of application include files or configuration information. It may even disclose the portion of code that failed. In most cases, it will be the result of the web application attempting to use an invalid client-supplied argument in a SQL statement, which means that SQL injection will be possible. If so, an attacker will at least be able to read the contents of the entire database arbitrarily. Depending on the database server and the SQL statement, deleting, updating and adding records and executing arbitrary commands may also be possible. If a software bug or bug is responsible for triggering the error, the potential impact will vary, depending on the circumstances. The location of the application that caused the error can be useful in facilitating other kinds of attacks. If the file is a hidden or include file, the attacker may be able to gain more information about the mechanics of the web application, possibly even the source code. Application source code is likely to contain usernames, passwords, database connection strings and aids the attacker greatly in discovering new vulnerabilities.

Fix:

For Security Operations:

Unknown application testing seeks to uncover new vulnerabilities in both custom and commercial software. Because of this, there are no specific patches or descriptions of this issue. Please note that this vulnerability may be a false positive if the page it is flagged on is technical documentation.However, follow these recommendations to help ensure a secure web application:

- **Use Uniform Error Codes:** Ensure that you are not inadvertently supplying information to an attacker via the use of inconsistent or "conflicting" error messages. For instance, don't reveal unintended information by using error messages such as Access Denied, which will also let an attacker know that the file he seeks actually exists. Use consistent terminology for files and folders that do exist, do not exist, and which have read access denied.
- **Informational Error Messages:** Ensure that error messages do not reveal too much information. Complete or partial paths, variable and file names, row and column names in tables, and specific database errors should never be revealed to the end user. Remember, an attacker will gather as much information as possible, and then add pieces of seemingly innocuous information together to craft an attack.
- **Proper Error Handling:** Use generic error pages and error handling logic to inform end users of potential problems. Do not provide system information or other data that could be used by an attacker when orchestrating an attack.

For Development:

This problem arises from the improper validation of characters that are accepted by the application. Any time a parameter is passed into a dynamically-generated web page, you must assume that the data could be incorrectly formatted. The application should contain sufficient logic to handle any situation in which a parameter is not being passed or is being passed incorrectly. Keep in mind how the data is being submitted, as a result of a GET or a POST. Additionally, to develop secure and stable code, treat cookies the same as parameters. The following recommendations will help ensure that you are delivering secure web applications.

- **Stringently define the data type:** Stringently define the data type (a string, an alphanumeric character, etc.) that the application will accept. Validate input for improper characters. Adopt the philosophy of using what is good rather than what is bad. Define the allowed set of characters. For instance, if a field is to receive a number, allow that field to accept

only numbers. Define the maximum and minimum data lengths that the application will accept.

- **Verify parameter is being passed:** If a parameter that is expected to be passed to a dynamic Web page is omitted, the application should provide an acceptable error message to the user. Also, never use a parameter until you have verified that it has been passed into the application.
- **Verify correct format:** Never assume that a parameter is of a valid format. This is especially true if the parameter is being passed to a SQL database. Any string that is passed directly to a database without first being checked for proper format can be a major security risk. Also, just because a parameter is normally provided by a combo box or hidden field, do not assume the format is correct. A hacker will first try to alter these parameters while attempting to break into your site.
- **Verify file names being passed in via a parameter:** If a parameter is being used to determine which file to process, never use the file name before it is verified as valid. Specifically, test for the existence of characters that indicate directory traversal, such as .../, c:\, and /.
- **Do not store critical data in hidden parameters:** Many programmers make the mistake of storing critical data in a hidden parameter or cookie. They assume that since the user doesn't see it, it's a good place to store data such as price, order number, etc. Both hidden parameters and cookies can be manipulated and returned to the server, so never assume the client returned what you sent via a hidden parameter or cookie.

For QA:

From a testing perspective, ensure that the error handling scheme is consistent and does not reveal private information about your web application. A seemingly innocuous piece of information can provide an attacker the means to discover additional information that can be used to conduct an attack. Make the following observations:

- Do you receive the same type of error for existing and non-existing files?
- Does the error include phrases (such as "Permission Denied") that could reveal the existence of a file?

Reference:

Web Application Security Whitepaper:
http://download.hpsmartupdate.com/asclabs/security_at_the_next_level.pdf

Processing Unhandled Exceptions:
http://www.asp.net/(S(sf10gzjodvrpce55el2p5cnk))/learn/hosting/tutorial-12-cs.aspx

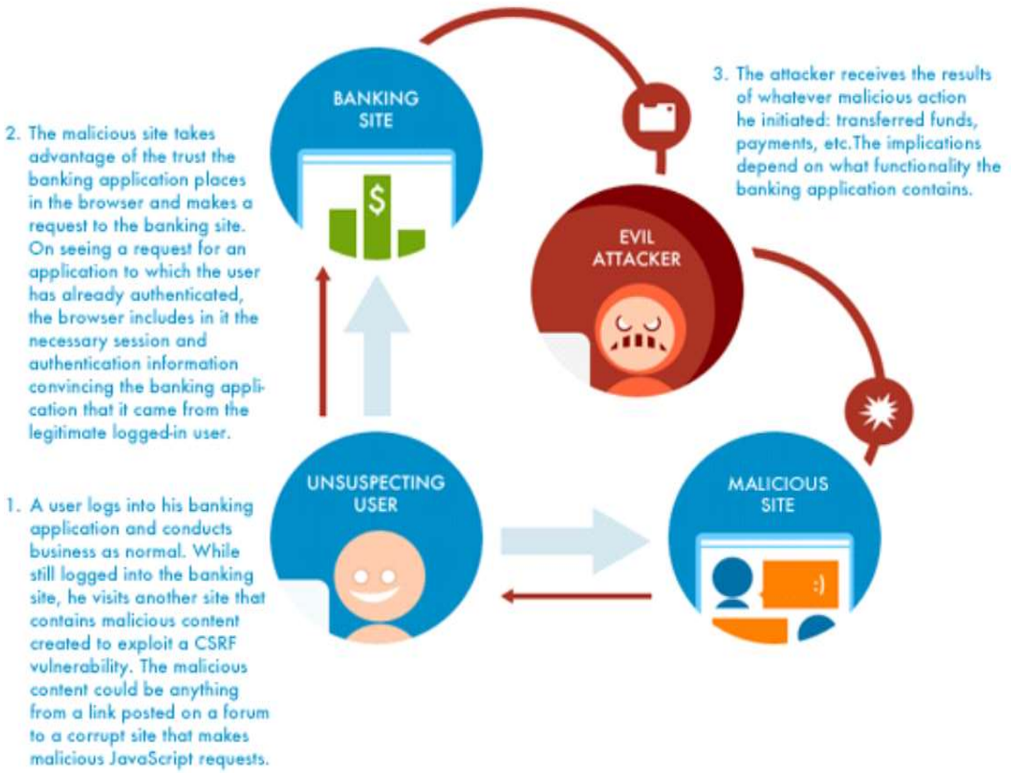
Managing Unhandled Exceptions:
http://www.informit.com/articles/article.aspx?p=32081&seqNum=3

File Names:

- http://10.67.67.107:8980/cs/Logon/Logon.xhtml
- http://10.67.67.107:8980/cs/Logon/Logon.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Medium **Cross-Site Request Forgery**

Summary:



Cross-Site Request Forgery (XSRF or CSRF) has been detected. Because browsers can run code sent by multiple sites, an XSRF attack can occur if one site sends a request (never seen by the user) to another site on which the user has authenticated that will mistakenly be received as if the user authorized the request. If a user visits a vulnerable site, the attacker can make the user's browser send a request to a different target site that performs an action on behalf of the user. The target site only sees a normal authenticated request coming from the user and performs whatever sensitive action was requested. Whatever functionality exists on the target site can be manipulated in this fashion. Recommendations include utilizing CAPTCHA's or anti-Cross-Site Request Forgery tokens to prevent Cross-Site Request Forgery attacks.

Execution:

Criteria for identifying CSRF:

1. This check is only run against POST requests.
2. The page must be either a login page, or a page in restricted session (i.e. an authenticated session) .
* Note: In order to avoid testing every POST request made during authenticated sessions, we will only run the check against a URL one time. This means that forms with multiple parameters will only be tested one time and not multiple times like a XSS or parameter injection check.
3. The page is not a re-authentication page. This is to avoid cases where a user is asked to either change a password or provide their password when they are already in an authenticated session. A re-authentication page is not CSRF vulnerable.
4. The page does not contain CAPTCHA. A CAPTCHA page is not CSRF vulnerable
5. The page is not an error page or an invalid page from the server.

More information on Login CSRF can be found here: http://seclab.stanford.edu/websec/csrf/csrf.pdf.

Implication:

Any functionality contained within the application can be exploited if it is vulnerable to XSRF. For instance, a banking application could be made to transfer funds, etc.

Fix:

Resolving Cross-Site Request Forgery may require recoding every form and feature of a web application. You can use anti-Cross-Site Request Forgery tokens or CAPTCHAs to prevent Cross-Site Request Forgery attacks. Other methods are easier but not as effective.

While no method of preventing Cross-Site Request Forgery is perfect, using Cross-Site Request Forgery nonce tokens eliminates most of the risk. Although an attacker may guess a valid token, nonce tokens are effective in preventing Cross-Site Request Forgery attacks. You can verify that a user is legitimate by generating a "secret," such as a secret hash or token, after the user logs in. You should store "the secret" in a server-side session and then include it in every link and sensitive form. Each subsequent HTTP request should include this token; otherwise, the request is denied and the session invalidated. Do not make the token the same as the session ID in case a Cross-Site Scripting vulnerability exists. Initialize the token as other session variables. You can validate it with a simple conditional statement, and you can limit it to a small timeframe to enhance its effectiveness. Attackers need to include a valid token with a Cross-Site Request Forgery attack in order to match

the form submission. Because the user's token is stored in the session, any attacker needs to use the same token as the victim.

CAPTCHA can also prevent cross-site request forgery attacks. With CAPTCHA, a user needs to enter a word shown in distorted text, contained inside an image, before continuing. The assumption is that a computer cannot determine the word inside the graphic, although a human can. CAPTCHA requires that a user authorize specific actions before the web application initiates them. It is difficult to create a script that automatically enters text to continue, but research is underway on how to break CAPTCHAs. If you use CAPTCHAs, make sure they are strong against possible attacks. Building a secure CAPTCHA takes more effort. In addition tomaking sure that computers cannot read the images, you need to make sure that the CAPTCHA cannot be bypassed at the script level. Consider whether you use the same CAPTCHA multiple times, making an application vulnerable to a replay attack. Also make sure the answer to the CAPTCHA is not passed in plain text as part of a web form.

More information is available in the HP Cross-Site Request Forgery white paper.

Reference:

HP XSRF White Paper:
Cross-Site Request Forgery

OWASP Prevention Cheat Sheet:
OWASP Prevention Cheat Sheet

XSRF FAQ:
http://www.cgisecurity.com/csrf-faq.html

Research:
http://seclab.stanford.edu/websec/csrf/csrf.pdf

File Names:

- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

---

Medium **Cross-Frame Scripting**

Summary:

A Cross-Frame Scripting (XFS) vulnerability can allow an attacker to load the vulnerable application inside an HTML iframe tag on a malicious page. The attacker could use this weakness to devise a Clickjacking attack to conduct phishing, frame sniffing, social engineering or Cross-Site Request Forgery attacks.

Clickjacking
The goal of a Clickjacking attack is to deceive the victim user into interacting with UI elements of the attacker's choice on the target web site without her knowledge and in turn executing privileged functionality on the victim's behalf. To achieve this goal, the attacker must exploit the XFS vulnerability to load the attack target inside an iframe tag, hide it using Cascading Style Sheets (CSS) and overlay the phishing content on the malicious page. By placing the UI elements on the phishing page to overlap with those on the page targeted in the attack, the attacker can ensure that the victim is forced to interact with the UI elements on the target page not visible to the victim.
WebInspect has detected a response containing one or more forms that accept user input but is missing XFS protection.

An effective frame-busting technique was not observed while loading this page inside a frame.

Execution:

Create a test page containing an HTML <iframe> tag whose **src** attribute is set to http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml. Successful framing of the target page indicates the application's susceptibility to XFS.
Note that WebInspect will report only one instance of this check across each host within the scope of the scan. The other visible pages on the site may, however, be vulnerable to XFS as well and hence should be protected against it with an appropriate fix.

Implication:

A Cross-Frame Scripting weakness could allow an attacker to embed the vulnerable application inside an iframe. Exploitation of this weakness could result in:

Hijacking of user events such as keystrokes
Theft of sensitive information
Execution of privileged functionality through combination with Cross-Site Request Forgery attacks

Fix:

Browser vendors have introduced and adopted a policy-based mitigation technique using the X-Frame-Options header. Developers can use this header to instruct the browser about appropriate actions to perform if their site is included inside an iframe. Developers must set the X-Frame-Options header to one of the following permitted values:

- DENY

Deny all attempts to frame the page
- SAMEORIGIN

The page can be framed by another page only if it belongs to the same origin as the page being framed
- ALLOW-FROM origin

Developers can specify a list of trusted origins in the origin attribute. Only pages on origin are permitted to load this page inside an iframe

Developers must **also** use client-side frame busting JavaScript as a protection against XFS. This will enable users of older browsers that do not support the X-Frame-Options header to also be protected from clickjacking attacks.

Reference:

HP 2012 Cyber Security Report
The X-Frame-Options header - a failure to launch

Server Configuration:
IIS
Apache, nginx

Specification:
X-Frame-Options IETF Draft

OWASP:
Clickjacking

Frame Busting:
Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites
OWASP: Busting Frame Busting

File Names:
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

---

Medium **Privacy Violation: Inconsistent Feedback**

Summary:

When entering an invalid username or password during a login, an application may provide meaningful feedback through a response discrepancy. For the potential attacker, this discrepancy increases the chances of a successful brute force attack against the site's authentication.

Execution:

Try logging into the application twice, once with an incorrect username and another time with an incorrect password. A generic error message for both attempts indicates a secure application. On the other hand, if there is a difference in the error messages, the application may be providing information that could be used in a brute-force attack.

Implication:

Consider an application that takes an email address/username and a password. If the application provides different error messages for a non-existent email address and an incorrect password, it will enable an attacker to submit multiple email addresses and learn about the ones that are actually registered to the application. Being able to enumerate users in an application may enable an attacker to perform a more efficient brute force attack.

Fix:

Applications should not indicate specifically whether a user account or password was incorrect, rather a very generic login failure message should be used. For example, during a new user registration or forgotten password function, no meaningful message should be returned to the requestor, rather an email should be sent to the account in question with details of how to reset a forgotten password or reclaim an account.

Reference:

OWASP Guide to Authentication
https://www.owasp.org/index.php/Guide_to_Authentication

Username Enumeration Vulnerabilities
http://www.gnucitizen.org/blog/username-enumeration-vulnerabilities

File Names:
- http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Low **Web Server Misconfiguration: Server Error Message**

Summary:

A server error response was detected. The server could be experiencing errors due to a misbehaving application, a misconfiguration, or a malicious value sent during the auditing process. While error responses in and of themselves are not dangerous, per se, the error responses give attackers insight into how the application handles error conditions. Errors that can be remotely triggered by an attacker can also potentially lead to a denial of service attack or other more severe vulnerability. Recommendations include designing and adding consistent error handling mechanisms which are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Implication:

The server has issued a 500 error response. While the body content of the error page may not expose any information about the technical error, the fact that an error occurred is confirmed by the 500 status code. Knowing whether certain inputs trigger a server error can aid or inform an attacker of potential vulnerabilities.

Fix:

For Security Operations:

Server error messages, such as "File Protected Against Access", often reveal more information than intended. For instance, an attacker who receives this message can be relatively certain that file exists, which might give him the information he needs to pursue other leads, or to perform an actual exploit. The following recommendations will help to ensure that a potential attacker is not deriving valuable information from any server error message that is presented.

- Uniform Error Codes: Ensure that you are not inadvertently supplying information to an attacker via the use of inconsistent or "conflicting" error messages. For instance, don't reveal unintended information by utilizing error messages such as Access Denied, which will also let an attacker know that the file he seeks actually exists. Have consistent terminology for files and folders that do exist, do not exist, and which have read access denied.
- Informational Error Messages: Ensure that error messages do not reveal too much information. Complete or partial paths, variable and file names, row and column names in tables, and specific database errors should never be revealed to the end user. Remember, an attacker will gather as much information as possible, and then add pieces of seemingly innocuous information together to craft a method of attack.
- Proper Error Handling: Utilize generic error pages and error handling logic to inform end users of potential problems. Do not provide system information or other data that could be utilized by an attacker when orchestrating an attack.

Removing Detailed Error Messages

Find instructions for turning off detailed error messaging in IIS at this link:

http://support.microsoft.com/kb/294807

For Development:

From a development perspective, the best method of preventing problems from arising from server error messages is to adopt secure programming techniques that prevent problems that might arise from an attacker discovering too much information about the architecture and design of your web application. The following recommendations can be used as a basis for that.

- Stringently define the data type (for instance, a string, an alphanumeric character, etc) that the application will accept.
- Use what is good instead of what is bad. Validate input for improper characters.
- Do not display error messages to the end user that provide information (such as table names) that could be utilized in orchestrating an attack.
- Define the allowed set of characters. For instance, if a field is to receive a number, only let that field accept numbers.
- Define the maximum and minimum data lengths for what the application will accept.
- Specify acceptable numeric ranges for input.

For QA:

The best course of action for QA associates to take is to ensure that the error handling scheme is consistent. Do you receive a different type of error for a file that does not exist as opposed to a file that does? Are phrases like "Permission Denied" utilized which could reveal the existence of a file to an attacker? Inconsistent methods of dealing with errors gives an attacker a very powerful way of gathering information about your web application.

Reference:

Apache:
Security Tips for Server Configuration
Protecting Confidential Documents at Your Site
Securing Apache - Access Control

Microsoft:
How to set required NTFS permissions and user rights for an IIS 5.0 Web server

Default permissions and user rights for IIS 6.0
Description of Microsoft Internet Information Services (IIS) 5.0 and 6.0 status codes

File Names:

- http://10.67.67.107:8980/cs/Logon/Logon.xhtml
- http://10.67.67.107:8980/cs/errorpages/error.xhtml
- http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
- http://10.67.67.107:8980/<script>alert('TRACK');</script>
- http://10.67.67.107:8980/cs/Main/Home/HomePage.xhtml

Low

Privacy Violation: Autocomplete

Summary:

Most recent browsers have features that will save password field content entered by users and then automatically complete password entry the next time the field are encountered. This feature is enabled by default and could leak password since it is stored on the hard drive of the user. The risk of this issue is greatly increased if users are accessing the application from a shared environment. Recommendations include setting autocomplete to "off" on all your password fields.

Please Note: Recent versions of most browsers, as noted below, now ignore the autocomplete="off" attribute for password fields in html forms. Users are allowed to decide the password policy at their own discretion using the password manager. Although setting is ineffective on these versions of browsers, it would continue to protect website users of earlier versions of these and other browsers that support this attribute.

Browsers NOT Supporting autocomplete="off":

Internet Explorer version 11 or above
Firefox version 30 or above
Chrome version 34 or above
For other browsers, please refer to vendor specific documentation

Execution:

To verify if a password filed is vulnerable, first make sure to enable the autocomplete in your browser's settings, and then input the other fileds of the form to see whether the password is automatically filled. If yes, then it's vulnerable, otherwise, not. You may need to do it twice in case it is the first time you type in the credential in your browser.

Please Note: That some modern browsers no longer support this attribute as summarized above. Verification should be done using a browser that supports this attribute.

Implication:

When autocomplete is enabled, hackers can directly steal your password from local storage.

Fix:

From the web application perspective, the autocomplete can be turned at the form level or individual entry level by defining the attribute AUTOCOMPLETE="off".

Reference:

Microsoft:
Autocomplete Security

File Names:

- http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Scan Name:	Site: http://10.67.67.107:8980/cs/Logon/Logon.xhtml		
Policy:	Standard	**Crawl Sessions:**	181
Scan Date:	1/8/2019 11:07:07 AM	**Vulnerabilities:**	20
Scan Version:	17.10.283.0	**Scan Duration:**	16 minutes : 25 seconds
Scan Type:	Site	**Client:**	FF

Server: http://10.67.67.107:8980

High Issues

Insecure Transport (4722) View Description

CWE: 287

Kingdom: Security Features

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Request:

```
GET /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl.EventMacro.Startup";
SID="1F00DBC96A4356A61FC8EDE423AFDF3A"; SessionType="StartMacro";
CrawlType="None";
X-RequestManager-Memo: sid="29"; smi="0"; Category="EventMacro.Login";
MacroName="LoginMacro1";
X-Request-Memo: ID="162a9718-f413-4436-ac35-012338e894b0"; tid="81";
Pragma: no-cache
Cookie: CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Content-Length: 13990
Set-Cookie: JSESSIONID=b1rp8b5Xo9GcWY-X8+c2sRR7; Path=/cs; HttpOnly
Content-Type: text/html;charset=UTF-8
Date: Tue, 08 Jan 2019 03:04:59 GMT

...TRUNCATED...<form id="form" name="form" method="post" action="
/cs/Logon/Logon.xhtml" enctype="application/x-www-form-urlencoded">
<in...TRUNCATED...
```

Often Misused: Login (10595) View Description

CWE: 287

Kingdom: API Abuse

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Request:

```
GET /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
```

```
Connection: keep-alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl.EventMacro.Startup";
SID="1F00DBC96A4356A61FC8EDE423AFDF3A"; SessionType="StartMacro";
CrawlType="None";
X-RequestManager-Memo: sid="29"; smi="0"; Category="EventMacro.Login";
MacroName="LoginMacro1";
X-Request-Memo: ID="162a9718-f413-4436-ac35-012338e894b0"; tid="81";
Pragma: no-cache
Cookie: CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Content-Length: 13990
Set-Cookie: JSESSIONID=b1rp8b5Xo9GcWY-X8+c2sRR7; Path=/cs; HttpOnly
Content-Type: text/html;charset=UTF-8
Date: Tue, 08 Jan 2019 03:04:59 GMT

...TRUNCATED...<form id="form" name="form" method="post" action="
/cs/Logon/Logon.xhtml" enctype="application/x-www-form-urlencoded">
<in...TRUNCATED...
```

Medium Issues

Poor Error Handling: Unhandled Exception (1498) View Description

CWE: 388,497,200

Kingdom: Errors

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=userName2&javax.faces.partial.execute=userName2&javax.faces.partial.render=userName2&javax.faces.behavior.event=change&javax.faces.partial.event=change&form=form&userName2=000196&userPwd=&javax.faces.ViewState=%00

Request:

```
POST /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Referer: http://10.67.67.107:8980/cs/Logon/Logon.xhtml
Content-Length: 262
Pragma: no-cache
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="B25228ACBF09168ADB245F478887C0D6";
PSID="DC037447CCA8EF662D6A42763A57902D"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="9"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="38";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="47"; smi="0"; sc="1"; ID="69c02489-360e-448b-
```

```
996f-8c154d5986a5";
X-Request-Memo: ID="d88f60de-c4bc-4863-890b-f9c84bb4279d"; sc="1";
ThreadId="56";
Cookie:
JSESSIONID=xV76xDwLnzMm33RU1+YiLSO0;CustomCookie=WebInspect121568ZX08AA2D62D
E924F83B2788E4F20DA3E8FYB3BD
Pragma: no-cache

...TRUNCATED...m&userName2=000196&userPwd=&javax.faces.ViewState=%00
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 5991
Date: Tue, 08 Jan 2019 03:08:11 GMT
Connection: close

...TRUNCATED...noshade"><p><b>JBWEB000309: type</b> JBWEB000066: Exception
report</p><p><b>JBWEB000068: message</b> <u></u></p><p><...TRUNCATED...
```

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=logonBtn&javax.faces.partial.execute=%40all&javax.faces.partial.render=form&logonBtn=logonBtn&form=form&userName2=000196&userPwd=Csc2618%40tgl&javax.faces.ViewState=%00

Request:

```
POST /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Referer: http://10.67.67.107:8980/cs/Logon/Logon.xhtml
Content-Length: 217
Pragma: no-cache
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="DB4CB3872FE2B85FCE207A51613575CA";
PSID="5CC058295CACA504CB833DC7FEC6962E"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="8"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="43";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="47"; smi="0"; sc="1"; ID="e42ebc63-4779-499c-
bab9-836aa755d32a";
X-Request-Memo: ID="44a92492-acfc-456f-a8f4-678c4b1452a2"; sc="1";
ThreadId="56";
Cookie:
JSESSIONID=xV76xDwLnzMm33RU1+YiLSO0;CustomCookie=WebInspect121568ZX08AA2D62D
E924F83B2788E4F20DA3E8FYB3BD
Pragma: no-cache

...TRUNCATED...00196&userPwd=Csc2618%40tgl&javax.faces.ViewState=%00
```

Response:

```
HTTP/1.1 500 Internal Server Error
```

```
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 5991
Date: Tue, 08 Jan 2019 03:08:12 GMT
Connection: close

...TRUNCATED...noshade"><p><b>JBWEB000309: type</b> JBWEB000066: Exception
report</p><p><b>JBWEB000068: message</b> <u></u></p><p><...TRUNCATED...
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=policyList%3AopOne&javax.faces.partial.execute=policyList%3AopOne&javax.faces.partial.render=policyList&javax.faces.behavior.event=click&javax.faces.partial.event=click&policyList=policyList&policyList%3AcustomRadio=opOne&policyList%3AopOne=&policyList%3AopTwo=&policyList%3AopThree=&policyList%3AopFour=&policyList%3AopFive=&policyList%3AopSix_A=&policyList%3AopSix_B_input=&policyList%3AopSix_C_focus=&policyList%3AopSix_C_input=&policyList%3AopSeven_A=&policyList%3AopSeven_B_input=&policyList%3AopSeven_C_focus=&policyList%3AopSeven_C_input=&policyList%3Aj_idt119=&javax.faces.ViewState=%00

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Host: 10.67.67.107:8980
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Content-Length: 643
X-AscRawUrl: /cs/Main/CSCArea/CSCPolicyQuery.xhtml
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="4CCC01B8161A875715E6BF2340982FFC";
PSID="EF456A06F9991224451DC947CEE3B6D4"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="22"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="41";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="47"; smi="0"; sc="1"; ID="f9eb83c4-408f-4d11-
be00-9227034cf0bc";
X-Request-Memo: ID="2c07ca5f-1126-40c4-a4ea-00b49e3c8103"; sc="1";
ThreadId="56";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-
X8+c2sRR7;_ga=GA1.1.2076340262.1546917073;_gid=GA1.1.2053967838.1546917073;_
gat_gtag_UA_116871856_1=1

...TRUNCATED...put=&policyList%3Aj_idt119=&javax.faces.ViewState=%00
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 6068
Date: Tue, 08 Jan 2019 03:09:16 GMT
Connection: close
```

```
...TRUNCATED...noshade"><p><b>JBWEB000309: type</b> JBWEB000066: Exception
report</p><p><b>JBWEB000068: message</b> <u></u></p><p><...TRUNCATED...
```

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=j_idt12&primefaces.ignoreautoupdate=true&javax.faces.partial.execute=j_idt12&javax.faces.partial.render=j_idt12&j_idt12=j_idt12&j_idt12_load=true&form=form&userName2=&userPwd=&javax.faces.ViewState=%00

Request:

```
POST /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Referer: http://10.67.67.107:8980/cs/Logon/Logon.xhtml
Content-Length: 250
Pragma: no-cache
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="8AD7401BDE3EE431FB10974F14B04ACE";
PSID="140CDDD499F9869BF8C5A963C703E3B8"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="10"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="35";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="45"; smi="0"; sc="1"; ID="e6addc5e-6e19-47cd-
b8dd-468aac9c331e";
X-Request-Memo: ID="2c989445-add2-4077-856a-74d1e37c9a2a"; sc="1";
ThreadId="55";
Cookie: JSESSIONID=Fg99C89cvrVXrHBHHMtzSRz
-;CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD
Pragma: no-cache

...TRUNCATED...rm=form&userName2=&userPwd=&javax.faces.ViewState=%003
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 5991
Date: Tue, 08 Jan 2019 03:08:06 GMT
Connection: close

...TRUNCATED...noshade"><p><b>JBWEB000309: type</b> JBWEB000066: Exception
report</p><p><b>JBWEB000068: message</b> <u></u></p><p><...TRUNCATED...
```

Cross-Site Request Forgery (10963) View Description

CWE: 284

Kingdom: Encapsulation

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: policyList=policyList&policyList%3acustomRadio=opOne

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl"; SID="8F3DED8C20B369570055B607333D3E0E";
PSID="C3A875CAE1D2CE2405A00686BF1473FD"; SessionType="Crawl";
CrawlType="Form"; AttackType="None"; OriginatingEngineID="00000000-0000-0000
-0000-000000000000"; AttributeName="action"; Format="Relative";
LinkKind="HyperLink"; Locations="HtmlNode"; Source="ScriptExecution";
ThreadId="69"; ThreadType="CrawlBreadthFirstDBReader";
X-RequestManager-Memo: sid="29"; smi="0"; sc="1"; ID="d3d5615e-60fe-4a38-
8d0b-c8ec6890b8a0";
X-Request-Memo: ID="6d5200ee-24bb-433e-8014-1b478c4a27c4"; sc="1";
ThreadId="44";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-X8+c2sRR7

policyList=policyList&policyList%3acustomRadio=opOne
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Location: http://10.67.67.107:8980/cs/Logon/ForcedLogOff.xhtml
Content-Length: 0
Date: Tue, 08 Jan 2019 03:06:30 GMT
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: policyList=policyList&policyList%3acustomRadio=opOne&policyList%3aopOne=&policyList%
3aopTwo=&policyList%3aopThree=&policyList%3aopFour=&policyList%3aopFive=&policyList%
3aopSix_A=&policyList%3aopSix_B_input=&policyList%3aopSix_C_focus=&policyList%
3aopSix_C_input=&policyList%3aopSeven_A=&policyList%3aopSeven_B_input=&policyList%
3aopSeven_C_focus=&policyList%3aopSeven_C_input=&policyList%3aj_idt119=&policyList%
3aj_idt120=&policyList%3aj_idt122=&javax.faces.ViewState=-4997982622880369550%
3a2461785887222810800

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 510
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl"; SID="66660CC44C38C20D4F79E1C58F2FE84F";
PSID="C3A875CAE1D2CE2405A00686BF1473FD"; SessionType="Crawl";
CrawlType="Form"; AttackType="None"; OriginatingEngineID="00000000-0000-0000
-0000-000000000000"; AttributeName="action"; Format="Relative";
```

```
LinkKind="HyperLink"; Locations="HtmlNode"; Source="ScriptExecution";
ThreadId="69"; ThreadType="CrawlBreadthFirstDBReader";
X-RequestManager-Memo: sid="29"; smi="0"; sc="1"; ID="3c081411-2398-47dc-
8b55-e0b058ca0f13";
X-Request-Memo: ID="edf26fb3-057b-4818-950b-df9fd8c5656f"; sc="1";
ThreadId="47";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-X8+c2sRR7

policyList=policyList&policyList%3acustomRadio=opOne&policyList%
3aopOne=&policyList%3aopTwo=&policyList%3aopThree=&policyList%
3aopFour=&policyList%3aopFive=&policyList%3aopSix_A=&policyList%
3aopSix_B_input=&policyList%3aopSix_C_focus=&policyList%
3aopSix_C_input=&policyList%3aopSeven_A=&policyList%
3aopSeven_B_input=&policyList%3aopSeven_C_focus=&policyList%
3aopSeven_C_input=&policyList%3aj_idt119=&policyList%3aj_idt120=&policyList%
3aj_idt122=&javax.faces.ViewState=-4997982622880369550%3a2461785887222810800
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Location: http://10.67.67.107:8980/cs/Logon/ForcedLogOff.xhtml
Content-Length: 0
Date: Tue, 08 Jan 2019 03:06:30 GMT
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=policyList%3AopOne&javax.faces.partial.execute=policyList%3AopOne&javax.faces.partial.render=policyList&javax.faces.behavior.event=click&javax.faces.partial.event=click&policyList=policyList&policyList%3AcustomRadio=opOne&policyList%3AopOne=&policyList%3AopTwo=&policyList%3AopThree=&policyList%3AopFour=&policyList%3AopFive=&policyList%3AopSix_A=&policyList%3AopSix_B_input=&policyList%3AopSix_C_focus=&policyList%3AopSix_C_input=&policyList%3AopSeven_A=&policyList%3AopSeven_B_input=&policyList%3AopSeven_C_focus=&policyList%3AopSeven_C_input=&policyList%3Aj_idt119=&javax.faces.ViewState=-4997982622880369550%3A2461785887222810800

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Host: 10.67.67.107:8980
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Content-Length: 682
X-AscRawUrl: /cs/Main/CSCArea/CSCPolicyQuery.xhtml
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: ScriptEngine="Gecko"; Category="Crawl";
SID="EF456A06F9991224451DC947CEE3B6D4";
PSID="C3A875CAE1D2CE2405A00686BF1473FD"; SessionType="Crawl";
CrawlType="AJAXInclude"; AttackType="None"; OriginatingEngineID="00000000-
0000-0000-0000-000000000000"; ThreadId="200"; ThreadType="JScriptEvent";
X-RequestManager-Memo: sid="29"; smi="0"; sc="1"; ID="c367b354-4cc2-4e2f-
982a-6a1f2aa620e2";
X-Request-Memo: ID="3b798305-6389-4af4-8b73-cdf03f58d000"; sc="1";
ThreadId="200";
Cookie:
```

```
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-X8+c2sRR7

javax.faces.partial.ajax=true&javax.faces.source=policyList%
3AopOne&javax.faces.partial.execute=policyList%
3AopOne&javax.faces.partial.render=policyList&javax.faces.behavior.event=cli
ck&javax.faces.partial.event=click&policyList=policyList&policyList%
3AcustomRadio=opOne&policyList%3AopOne=&policyList%3AopTwo=&policyList%
3AopThree=&policyList%3AopFour=&policyList%3AopFive=&policyList%
3AopSix_A=&policyList%3AopSix_B_input=&policyList%
3AopSix_C_focus=&policyList%3AopSix_C_input=&policyList%
3AopSeven_A=&policyList%3AopSeven_B_input=&policyList%
3AopSeven_C_focus=&policyList%3AopSeven_C_input=&policyList%
3Aj_idt119=&javax.faces.ViewState=-4997982622880369550%3A2461785887222810800
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Location: http://10.67.67.107:8980/cs/Logon/ForcedLogOff.xhtml
Content-Length: 0
Date: Tue, 08 Jan 2019 03:06:00 GMT
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: j_idt16=j_idt16

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl"; SID="2926C17950569A51401484AB271A651F";
PSID="C3A875CAE1D2CE2405A00686BF1473FD"; SessionType="Crawl";
CrawlType="Form"; AttackType="None"; OriginatingEngineID="00000000-0000-0000
-0000-000000000000"; AttributeName="action"; Format="Relative";
LinkKind="HyperLink"; Locations="HtmlNode"; Source="ScriptExecution";
ThreadId="69"; ThreadType="CrawlBreadthFirstDBReader";
X-RequestManager-Memo: sid="29"; smi="0"; sc="1"; ID="1d9f9f9e-f8f8-482e-
a742-9c61ad03f4ee";
X-Request-Memo: ID="935b1a00-d999-47d4-ae8a-e92810404cec"; sc="1";
ThreadId="48";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-X8+c2sRR7

j_idt16=j_idt16
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Location: http://10.67.67.107:8980/cs/Logon/ForcedLogOff.xhtml
Content-Length: 0
Date: Tue, 08 Jan 2019 03:06:30 GMT
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: j_idt47=j_idt47&javax.faces.ViewState=-4997982622880369550%3a2461785887222810800

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 80
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl"; SID="A45A6C36F6A29451171D57C0BA0E60ED";
PSID="C3A875CAE1D2CE2405A00686BF1473FD"; SessionType="Crawl";
CrawlType="Form"; AttackType="None"; OriginatingEngineID="00000000-0000-0000
-0000-000000000000"; AttributeName="action"; Format="Relative";
LinkKind="HyperLink"; Locations="HtmlNode"; Source="ScriptExecution";
ThreadId="69"; ThreadType="CrawlBreadthFirstDBReader";
X-RequestManager-Memo: sid="29"; smi="0"; sc="1"; ID="4b169a5a-7853-4df6-
8ede-fd364d69af30";
X-Request-Memo: ID="90c34aa4-86f1-4e50-b37d-7b3f73baba89"; sc="1";
ThreadId="47";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-X8+c2sRR7

j_idt47=j_idt47&javax.faces.ViewState=-4997982622880369550%
3a2461785887222810800
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Location: http://10.67.67.107:8980/cs/Logon/ForcedLogOff.xhtml
Content-Length: 0
Date: Tue, 08 Jan 2019 03:06:30 GMT
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: j_idt16=j_idt16&javax.faces.ViewState=-4997982622880369550%3a2461785887222810800

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 80
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl"; SID="334BE4D900AFDAF1B4960CA7B18A997A";
PSID="C3A875CAE1D2CE2405A00686BF1473FD"; SessionType="Crawl";
CrawlType="Form"; AttackType="None"; OriginatingEngineID="00000000-0000-0000
-0000-000000000000"; AttributeName="action"; Format="Relative";
```

```
LinkKind="HyperLink"; Locations="HtmlNode"; Source="ScriptExecution";
ThreadId="69"; ThreadType="CrawlBreadthFirstDBReader";
X-RequestManager-Memo: sid="29"; smi="0"; sc="1"; ID="1c6db0fb-900d-46b8-
a8e3-d3ed1acf264d";
X-Request-Memo: ID="bce62318-9a79-4f8d-b0a6-eaf3100fada8"; sc="1";
ThreadId="48";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-X8+c2sRR7

j_idt16=j_idt16&javax.faces.ViewState=-4997982622880369550%
3a2461785887222810800
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Location: http://10.67.67.107:8980/cs/Logon/ForcedLogOff.xhtml
Content-Length: 0
Date: Tue, 08 Jan 2019 03:06:30 GMT
```

Cross-Frame Scripting (11294) View Description

CWE: 352

Kingdom: Security Features

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

Request:

```
GET /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.67.67.107:8980/cs/Logon/LogOffUser.xhtml
Pragma: no-cache
Cookie: JSESSIONID=b1rp8b5Xo9GcWY-X8+c2sRR7;
_ga=GA1.1.1600098038.1546916849; _gid=GA1.1.288407134.1546916849;
_gat_gtag_UA_116871856_1=1;CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2
788E4F20DA3E8FYB3BD
Connection: keep-alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl.EventMacro.Startup";
SID="C456EA9621CE420E4D3E3852B286D65C"; SessionType="StartMacro";
CrawlType="None";
X-RequestManager-Memo: sid="29"; smi="0"; Category="EventMacro.Login";
MacroName="LoginMacro1";
X-Request-Memo: ID="dfa49315-b0d7-4b89-8dc8-9b1b734494f3"; tid="81";
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Content-Type: text/html;charset=UTF-8
Date: Tue, 08 Jan 2019 03:05:09 GMT
Content-Length: 43967

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"><head id="j_idt2">
            <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8" />
            <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, user-scalable=0" />

            <meta name="apple-mobile-web-app-capable" content="yes" /><link
type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/theme.css.xhtml?ln=primefaces-barcelona-
blue" /><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/fa/font-awesome.css.xhtml?
ln=primefaces&amp;amp;v=6.1" /><script type="text/javascript"
src="/cs/javax.faces.resource/jquery/jquery.js.xhtml?
ln=primefaces&amp;amp;v=6.1"></script><script type="text/javascript"
src="/cs/javax.faces.resource/core.js.xhtml?
ln=primefaces&amp;amp;v=6.1"></script><script type="text/javascript"
src="/cs/javax.faces.resource/idlemonitor/idlemonitor.js.xhtml?
ln=primefaces&amp;amp;v=6.1"></script><script type="text/javascript"
src="/cs/javax.faces.resource/jquery/jquery-plugins.js.xhtml?
ln=primefaces&amp;amp;v=6.1"></script><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/components.css.xhtml?
ln=primefaces&amp;amp;v=6.1" /><script type="text/javascript"
src="/cs/javax.faces.resource/components.js.xhtml?
ln=primefaces&amp;amp;v=6.1"></script><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/scrollpanel/scrollpanel.css.xhtml?
ln=primefaces&amp;amp;v=6.1" /><script type="text/javascript"
src="/cs/javax.faces.resource/scrollpanel/scrollpanel.js.xhtml?
ln=primefaces&amp;amp;v=6.1"></script><script type="text/javascript"
src="/cs/javax.faces.resource/primefaces-extensions.js.xhtml?ln=primefaces-
extensions&amp;amp;v=6.1"></script><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/blockui/blockui.css.xhtml?ln=primefaces-
extensions&amp;amp;v=6.1" /><script type="text/javascript"
src="/cs/javax.faces.resource/blockui/blockui.js.xhtml?ln=primefaces-
extensions&amp;amp;v=6.1"></script><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/css/nanoscroller.css.xhtml?ln=barcelona-
layout" /><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/css/animate.css.xhtml?ln=barcelona-
layout" /><link type="text/css" rel="stylesheet"
href="/cs/javax.faces.resource/css/ripple.css.xhtml?ln=barcelona-
layout" /><link type="text/css"
rel="stylesheet" href="/cs/javax.faces.resource/css/layout-blue.css.xhtml?
ln=barcelona-layout" /><script type="text/javascript">if(window.PrimeFaces)
{PrimeFaces.settings.locale='zh_TW';}</script>

    <title> - (HIDE)</title>

      <link rel="shortcut icon" type="image/x-icon"
href="/cs/javax.faces.resource/barcelona-
layout/images/favicon.ico.xhtml" /><script type="text/javascript"
src="/cs/javax.faces.resource/js/nanoscroller.js.xhtml?ln=barcelona-
layout"></script><script type="text/javascript"
src="/cs/javax.faces.resource/js/layout.js.xhtml?ln=barcelona-
layout"></script><script type="text/javascript"
src="/cs/javax.faces.resource/js/ripple.js.xhtml?ln=barcelona-
layout"></script><script type="text/javascript"
src="/cs/javax.faces.resource/js/localzhTW.js.xhtml?ln=barcelona-
layout"></script><script type="text/javascript"
src="/cs/javax.faces.resource/js/jquery-ui.js.xhtml?ln=barcelona-
layout"></script><script type="text/javascript"
src="/cs/javax.faces.resource/js/JQueryDatePickerTW.js.xhtml?ln=barcelona-
layout"></script><script type="text/javascript"
src="/cs/javax.faces.resource/js/sas.js.xhtml?ln=barcelona-layout"></script>


      <script src="https://www.googletagmanager.com/gtag/js?id=UA-116871856-
1"></script>
      <script>
            window.dataLayer = window.dataLayer || [];
            function gtag(){dataLayer.push(arguments);}
```

```
            gtag('js', new Date());

            gtag('config', 'UA-116871856-1');
      </script>

      <style id="antiClickjack">body{display:none !important;}</style>
      <style>
            .scrollup {
                  width: 64px;
                  height: 63px;
                  position: f

...TRUNCATED...
```

Privacy Violation: Inconsistent Feedback (11418) View Description

CWE: 209

Kingdom: Security Features

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=userName2&javax.faces.partial.execute=userName2&javax.faces.partial.render=userName2&javax.faces.behavior.event=change&javax.faces.partial.event=change&form=form&userName2=spiuser&userPwd=&javax.faces.ViewState=9219717795314723088%3A4900792364424845795

Request:

```
POST /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Referer: http://10.67.67.107:8980/cs/Logon/Logon.xhtml
Content-Length: 301
Pragma: no-cache
Cookie: JSESSIONID=SV6EbOyxITk-6cq3vGqREKfq;
_ga=GA1.1.1415451547.1546917039; _gid=GA1.1.132746382.1546917039;
_gat_gtag_UA_116871856_1=1;CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2
788E4F20DA3E8FYB3BD
Connection: keep-alive
Pragma: no-cache
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.EventMacro.Workflow";
SID="BBD32F46EF487570DD13648F4BFA2FED"; SessionType="NamedMacro";
CrawlType="None"; OriginatingEngineID="41d8521e-1c73-444b-80fa-
3518246dca05"; TriggerSID="DC037447CCA8EF662D6A42763A57902D";
X-RequestManager-Memo: sid="95"; smi="0"; Category="EventMacro.Named";
MacroName="none";
X-Request-Memo: ID="c377e4f8-9815-494e-b75e-6cc502ad88b1"; tid="255";

...TRUNCATED...ax.faces.partial.event=change&form=form&userName2=spiuser
&userPwd=&javax.faces.ViewState=921971779531472308...TRUNCATED...
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
Content-Type: text/xml;charset=UTF-8
Content-Length: 711
```

```
Date: Tue, 08 Jan 2019 03:08:14 GMT
```

Low Issues

Web Server Misconfiguration: Server Error Message (10932) View Description

CWE: 388,497,200

Kingdom: Environment

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=j_idt12&primefaces.ignoreautoupdate=true&javax.faces.partial.execute=j_idt12&javax.faces.partial.render=j_idt12&j_idt12=j_idt12&j_idt12_load=true&form=form&userName2=&userPwd=&javax.faces.ViewState=%00

Request:

```
POST /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Referer: http://10.67.67.107:8980/cs/Logon/Logon.xhtml
Content-Length: 250
Pragma: no-cache
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="8AD7401BDE3EE431FB10974F14B04ACE";
PSID="140CDDD499F9869BF8C5A963C703E3B8"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="10"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="35";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="45"; smi="0"; sc="1"; ID="e6addc5e-6e19-47cd-
b8dd-468aac9c331e";
X-Request-Memo: ID="2c989445-add2-4077-856a-74d1e37c9a2a"; sc="1";
ThreadId="55";
Cookie: JSESSIONID=Fg99C89cvrVXrHBHHMtzSRz
-;CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD
Pragma: no-cache

...TRUNCATED...rm=form&userName2=&userPwd=&javax.faces.ViewState=%003
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1...TRUNCATED...
```

Page: http://10.67.67.107:8980/cs/errorpages/error.xhtml

PostData: j_idt55=j_idt55&javax.faces.ViewState=%00

Request:

```
POST /cs/errorpages/error.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/errorpages/error.xhtml
```

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="E6773B5F1167D9FE5DFAD1974FDCA096";
PSID="4E3A73FF423B2555801A5895402CCC14"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="1"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="35";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="39"; smi="0"; sc="1"; ID="f5387e89-1767-4d1f-
8749-388f9bb11bff";
X-Request-Memo: ID="ff6298ce-d506-4889-a661-22f96c246480"; sc="1";
ThreadId="52";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=Fi0CLLsU+zM69NXz0lSEsyHS;_ga=GA1.1.650376965.1546917073;_gid=GA1.1.17718
8110.1546917073;_gat_gtag_UA_116871856_1=1

j_idt55=j_idt55&javax.faces.ViewState=^^
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1...TRUNCATED...
```

Page: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml

PostData: javax.faces.partial.ajax=true&javax.faces.source=policyList%3AopOne&javax.faces.partial.execute=policyList%3AopOne&javax.faces.partial.render=policyList&javax.faces.behavior.event=click&javax.faces.partial.event=click&policyList=policyList&policyList%3AcustomRadio=opOne&policyList%3AopOne=&policyList%3AopTwo=&policyList%3AopThree=&policyList%3AopFour=&policyList%3AopFive=&policyList%3AopSix_A=&policyList%3AopSix_B_input=&policyList%3AopSix_C_focus=&policyList%3AopSix_C_input=&policyList%3AopSeven_A=&policyList%3AopSeven_B_input=&policyList%3AopSeven_C_focus=&policyList%3AopSeven_C_input=&policyList%3Aj_idt119=&javax.faces.ViewState=%00

Request:

```
POST /cs/Main/CSCArea/CSCPolicyQuery.xhtml HTTP/1.1
Referer: http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml
Host: 10.67.67.107:8980
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Faces-Request: partial/ajax
X-Requested-With: XMLHttpRequest
Content-Length: 643
X-AscRawUrl: /cs/Main/CSCArea/CSCPolicyQuery.xhtml
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="4CCC01B8161A875715E6BF2340982FFC";
PSID="EF456A06F9991224451DC947CEE3B6D4"; SessionType="AuditAttack";
CrawlType="None"; AttackType="PostParamManipulation";
```

```
OriginatingEngineID="18264a0f-d83e-4ef5-a73d-1f06044c9fde";
AttackSequence="0"; AttackParamDesc="javax.faces.ViewState";
AttackParamIndex="22"; AttackParamSubIndex="0"; CheckId="2134";
Engine="Post+Injection"; SmartMode="NonServerSpecificOnly"; AttackString="%
2500"; AttackStringProps="Attack"; ThreadId="41";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="47"; smi="0"; sc="1"; ID="f9eb83c4-408f-4d11-
be00-9227034cf0bc";
X-Request-Memo: ID="2c07ca5f-1126-40c4-a4ea-00b49e3c8103"; sc="1";
ThreadId="56";
Cookie:
CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD;JSESSIO
NID=b1rp8b5Xo9GcWY-
X8+c2sRR7;_ga=GA1.1.2076340262.1546917073;_gid=GA1.1.2053967838.1546917073;_
gat_gtag_UA_116871856_1=1

...TRUNCATED...put=&policyList%3Aj_idt119=&javax.faces.ViewState=%00
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1...TRUNCATED...
```

Page: http://10.67.67.107:8980/<script>alert('TRACK');</script>

Request:

```
TRACK /<script>alert('TRACK');</script> HTTP/1.1
Referer: http://10.67.67.107:8980/cs/javax.faces.resource/theme.css.xhtml?
ln=primefaces-barcelona-blue
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Host: 10.67.67.107:8980
Connection: Keep-Alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Audit.Attack";
SID="C953C90EFAD693FB10A0BC993D588622";
PSID="C2EAEE49EA62C39A0F30E756D27C6B3C"; SessionType="AuditAttack";
CrawlType="None"; AttackType="Search"; OriginatingEngineID="65cee7d3-561f-
40dc-b5eb-c0b8c2383fcb"; AttackSequence="0"; AttackParamDesc="";
AttackParamIndex="0"; AttackParamSubIndex="0"; CheckId="5152";
Engine="Request+Modify"; SmartMode="NonServerSpecificOnly"; ThreadId="43";
ThreadType="AuditorStateRequestorPool";
X-RequestManager-Memo: sid="37"; smi="0"; sc="1"; ID="6b5e00ed-cf31-4655-
91a5-afe6cd7df9e8";
X-Request-Memo: ID="59527f70-e30f-4949-b130-3d038d693dc8"; sc="1";
ThreadId="51";
Cookie: CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD
```

Response:

```
HTTP/1.1 501 Not Implemented
Server: Apache-Coyote/1.1
Conte...TRUNCATED...
```

Page: http://10.67.67.107:8980/cs/Main/Home/HomePage.xhtml

Request:

```
GET /cs/Main/Home/HomePage.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
```

```
Firefox/30.0%0d%0aSPIHeader:%20SPIValue
Accept: text/html,application/xhtml+xml,applicat...TRUNCATED...
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1...TRUNCATED...
```

Privacy Violation: Autocomplete (11276) View Description

CWE: 200

Kingdom: Security Features

Page: http://10.67.67.107:8980/cs/Logon/Logon.xhtml

Request:

```
GET /cs/Logon/Logon.xhtml HTTP/1.1
Host: 10.67.67.107:8980
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101
Firefox/30.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
X-WIPP: AscVersion=17.10.283.0
X-Scan-Memo: Category="Crawl.EventMacro.Startup";
SID="1F00DBC96A4356A61FC8EDE423AFDF3A"; SessionType="StartMacro";
CrawlType="None";
X-RequestManager-Memo: sid="29"; smi="0"; Category="EventMacro.Login";
MacroName="LoginMacro1";
X-Request-Memo: ID="162a9718-f413-4436-ac35-012338e894b0"; tid="81";
Pragma: no-cache
Cookie: CustomCookie=WebInspect121568ZX08AA2D62DE924F83B2788E4F20DA3E8FYB3BD
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Frame-Options: SAMEORIGIN
Content-Length: 13990
Set-Cookie: JSESSIONID=b1rp8b5Xo9GcWY-X8+c2sRR7; Path=/cs; HttpOnly
Content-Type: text/html;charset=UTF-8
Date: Tue, 08 Jan 2019 03:04:59 GMT

...TRUNCATED... "></div>
                <div class="input-block"><input id="userPwd"
name="userPwd" type="password" class="ui-inputfield ui-password ui-widget ui
-state-default ui-corner-all" tabindex="2" /><script id="userPwd_s"
type="text/javascript">$(fu...TRUNCATED...
```

Appendix (Check Descriptions)

Insecure Transport (4722)

Summary

Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. http://10.67.67.107:8980/cs/Logon/Logon.xhtml has failed this policy. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to

prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix

For Security Operations:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For Development:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For QA:
Test the application not only from the perspective of a normal user, but also from the perspective of a malicious one.

Reference

Classifications

CWE-287: Improper Authentication
http://cwe.mitre.org/data/definitions/287.html

Kingdom: Security Features
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Often Misused: Login (10595)

Summary

An unencrypted login form has been discovered. Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. If the login form is being served over SSL, the page that the form is being submitted to MUST be accessed over SSL. Every link/URL present on that page (not just the form action) needs to be served over HTTPS. This will prevent Man-in-the-Middle attacks on the login form. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix

Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Reference

Advisory:http://www.kb.cert.org/vuls/id/466433

Classifications

CWE-287: Improper Authentication
http://cwe.mitre.org/data/definitions/287.html

Kingdom: API Abuse
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Poor Error Handling: Unhandled Exception (1498)

Summary

Unhandled exceptions are circumstances in which the application has received user input that it did not expect and doesn't know how to deal with. In many cases, an attacker can leverage the conditions that cause these errors in order to gain unauthorized access to the system. Recommendations include designing and adding consistent error-handling mechanisms that are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Implication

Exception error messages may contain the location of the file in which the offending function is located. This may disclose the webroot's absolute path as well as give the attacker the location of application include files or configuration information. It may even disclose the portion of code that failed. In most cases, it will be the result of the web application attempting to use an invalid client-supplied argument in a SQL statement, which means that SQL injection will be possible. If so, an attacker will at least be able to read the contents of the entire database arbitrarily. Depending on the database server and the SQL statement, deleting, updating and adding records and executing arbitrary commands may also be possible. If a software bug or bug is responsible for triggering the error, the potential impact will vary, depending on the circumstances. The location of the application that caused the error can be useful in facilitating other kinds of attacks. If the file is a hidden or include file, the attacker may be able to gain more information about the mechanics of the web application, possibly even the source code. Application source code is likely to contain usernames, passwords, database connection strings and aids the attacker greatly in discovering new vulnerabilities.

Fix

For Security Operations:

Unknown application testing seeks to uncover new vulnerabilities in both custom and commercial software. Because of this, there are no specific patches or descriptions of this issue. Please note that this vulnerability may be a false positive if the page it is flagged on is technical documentation.However, follow these recommendations to help ensure a secure web application:

- **Use Uniform Error Codes:** Ensure that you are not inadvertently supplying information to an attacker via the use of inconsistent or "conflicting" error messages. For instance, don't reveal unintended information by using error messages such as Access Denied, which will also let an attacker know that the file he seeks actually exists. Use consistent terminology for files and folders that do exist, do not exist, and which have read access denied.
- **Informational Error Messages:** Ensure that error messages do not reveal too much information. Complete or partial paths, variable and file names, row and column names in tables, and specific database errors should never be revealed to the end user. Remember, an attacker will gather as much information as possible, and then add pieces of seemingly innocuous information together to craft an attack.
- **Proper Error Handling:** Use generic error pages and error handling logic to inform end users of potential problems. Do not provide system information or other data that could be used by an attacker when orchestrating an attack.

For Development:

This problem arises from the improper validation of characters that are accepted by the application. Any time a parameter is passed into a dynamically-generated web page, you must assume that the data could be incorrectly formatted. The application should contain sufficient logic to handle any situation in which a parameter is not being passed or is being passed incorrectly. Keep in mind how the data is being submitted, as a result of a GET or a POST. Additionally, to develop secure and stable code, treat cookies the same as parameters. The following recommendations will help ensure that you are delivering secure web applications.

- **Stringently define the data type:** Stringently define the data type (a string, an alphanumeric character, etc.) that the application will accept. Validate input for improper characters. Adopt the philosophy of using what is good rather than what is bad. Define the allowed set of characters. For instance, if a field is to receive a number, allow that field to accept only numbers. Define the maximum and minimum data lengths that the application will accept.
- **Verify parameter is being passed:** If a parameter that is expected to be passed to a dynamic Web page is omitted, the application should provide an acceptable error message to the user. Also, never use a parameter until you have verified that it has been passed into the application.
- **Verify correct format:** Never assume that a parameter is of a valid format. This is especially true if the parameter is being passed to a SQL database. Any string that is passed directly to a database without first being checked for proper format can be a major security risk. Also, just because a parameter is normally provided by a combo box or hidden field, do not assume the format is correct. A hacker will first try to alter these parameters while attempting to break into your site.
- **Verify file names being passed in via a parameter:** If a parameter is being used to determine which file to process, never use the file name before it is verified as valid. Specifically, test for the existence of characters that indicate directory traversal, such as .../, c:\, and /.
- **Do not store critical data in hidden parameters:** Many programmers make the mistake of storing critical data in a hidden parameter or cookie. They assume that since the user doesn't see it, it's a good place to store data such as price, order number, etc. Both hidden parameters and cookies can be manipulated and returned to the server, so never assume the client returned what you sent via a hidden parameter or cookie.

For QA:

From a testing perspective, ensure that the error handling scheme is consistent and does not reveal private information about your web application. A seemingly innocuous piece of information can provide an attacker the means to discover additional information that can be used to conduct an attack. Make the following observations:

- Do you receive the same type of error for existing and non-existing files?
- Does the error include phrases (such as "Permission Denied") that could reveal the existence of a file?

Reference

Web Application Security Whitepaper:
http://download.hpsmartupdate.com/asclabs/security_at_the_next_level.pdf

Processing Unhandled Exceptions:
http://www.asp.net/(S(sf10gzjodvrpce55el2p5cnk))/learn/hosting/tutorial-12-cs.aspx

Managing Unhandled Exceptions:
http://www.informit.com/articles/article.aspx?p=32081&seqNum=3

Classifications

CWE-388: Error Handling
http://cwe.mitre.org/data/definitions/388.html

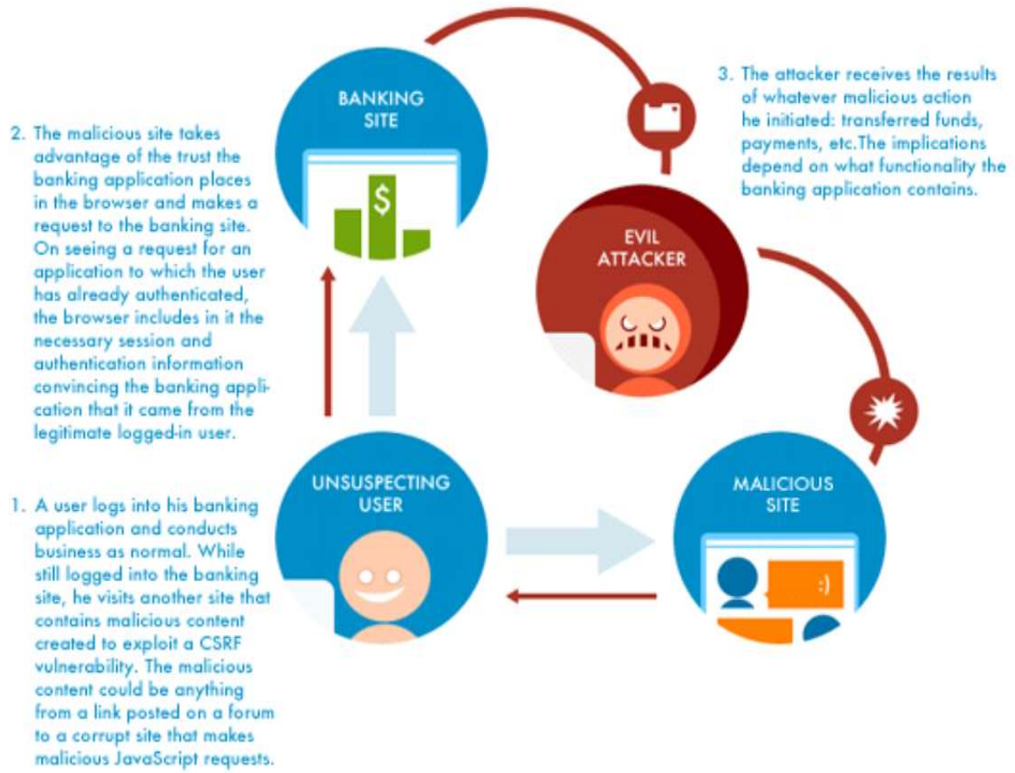
CWE-497: Exposure of System Data to an Unauthorized Control Sphere
http://cwe.mitre.org/data/definitions/497.html

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Errors
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Cross-Site Request Forgery (10963)

Summary



Cross-Site Request Forgery (XSRF or CSRF) has been detected. Because browsers can run code sent by multiple sites, an XSRF attack can occur if one site sends a request (never seen by the user) to another site on which the user has authenticated that will mistakenly be received as if the user authorized the request. If a user visits a vulnerable site, the attacker can make the user's browser send a request to a different target site that performs an action on behalf of the user. The target site only sees a normal authenticated request coming from the user and performs whatever sensitive action was requested. Whatever functionality exists on the target site can be manipulated in this fashion. Recommendations include utilizing CAPTCHA's or anti-Cross-Site Request Forgery tokens to prevent Cross-Site Request Forgery attacks.

Execution

Criteria for identifying CSRF:

1. This check is only run against POST requests.
2. The page must be either a login page, or a page in restricted session (i.e. an authenticated session) .
* Note: In order to avoid testing every POST request made during authenticated sessions, we will only run the check against a URL one time. This means that forms with multiple parameters will only be tested one time and not multiple times like a XSS or parameter injection check.
3. The page is not a re-authentication page. This is to avoid cases where a user is asked to either change a password or provide their password when they are already in an authenticated session. A re-authentication page is not CSRF vulnerable.
4. The page does not contain CAPTCHA. A CAPTCHA page is not CSRF vulnerable
5. The page is not an error page or an invalid page from the server.

More information on Login CSRF can be found here: http://seclab.stanford.edu/websec/csrf/csrf.pdf.

Implication

Any functionality contained within the application can be exploited if it is vulnerable to XSRF. For instance, a banking application could be made to transfer funds, etc.

Fix

Resolving Cross-Site Request Forgery may require recoding every form and feature of a web application. You can use anti-Cross-Site Request Forgery tokens or CAPTCHAs to prevent Cross-Site Request Forgery attacks. Other methods are easier but not as effective.

While no method of preventing Cross-Site Request Forgery is perfect, using Cross-Site Request Forgery nonce tokens

eliminates most of the risk. Although an attacker may guess a valid token, nonce tokens are effective in preventing Cross-Site Request Forgery attacks. You can verify that a user is legitimate by generating a "secret," such as a secret hash or token, after the user logs in. You should store “the secret” in a server-side session and then include it in every link and sensitive form. Each subsequent HTTP request should include this token; otherwise, the request is denied and the session invalidated. Do not make the token the same as the session ID in case a Cross-Site Scripting vulnerability exists. Initialize the token as other session variables. You can validate it with a simple conditional statement, and you can limit it to a small timeframe to enhance its effectiveness. Attackers need to include a valid token with a Cross-Site Request Forgery attack in order to match the form submission. Because the user’s token is stored in the session, any attacker needs to use the same token as the victim.

CAPTCHA can also prevent cross-site request forgery attacks. With CAPTCHA, a user needs to enter a word shown in distorted text, contained inside an image, before continuing. The assumption is that a computer cannot determine the word inside the graphic, although a human can. CAPTCHA requires that a user authorize specific actions before the web application initiates them. It is difficult to create a script that automatically enters text to continue, but research is underway on how to break CAPTCHAs. If you use CAPTCHAs, make sure they are strong against possible attacks. Building a secure CAPTCHA takes more effort. In addition tomaking sure that computers cannot read the images, you need to make sure that the CAPTCHA cannot be bypassed at the script level. Consider whether you use the same CAPTCHA multiple times, making an application vulnerable to a replay attack. Also make sure the answer to the CAPTCHA is not passed in plain text as part of a web form.

More information is available in the HP Cross-Site Request Forgery white paper.

Reference

HP XSRF White Paper:
Cross-Site Request Forgery

OWASP Prevention Cheat Sheet:
OWASP Prevention Cheat Sheet

XSRF FAQ:
http://www.cgisecurity.com/csrf-faq.html

Research:
http://seclab.stanford.edu/websec/csrf/csrf.pdf

Classifications

CWE-284: Access Control (Authorization) Issues
http://cwe.mitre.org/data/definitions/284.html

Kingdom: Encapsulation
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Cross-Frame Scripting (11294)

Summary

A Cross-Frame Scripting (XFS) vulnerability can allow an attacker to load the vulnerable application inside an HTML iframe tag on a malicious page. The attacker could use this weakness to devise a Clickjacking attack to conduct phishing, frame sniffing, social engineering or Cross-Site Request Forgery attacks.

Clickjacking
The goal of a Clickjacking attack is to deceive the victim user into interacting with UI elements of the attacker’s choice on the target web site without her knowledge and in turn executing privileged functionality on the victim’s behalf. To achieve this goal, the attacker must exploit the XFS vulnerability to load the attack target inside an iframe tag, hide it using Cascading Style Sheets (CSS) and overlay the phishing content on the malicious page. By placing the UI elements on the phishing page to overlap with those on the page targeted in the attack, the attacker can ensure that the victim is forced to interact with the UI elements on the target page not visible to the victim.
WebInspect has detected a response containing one or more forms that accept user input but is missing XFS protection.

An effective frame-busting technique was not observed while loading this page inside a frame.

Execution

Create a test page containing an HTML <iframe> tag whose **src** attribute is set to http://10.67.67.107:8980/cs/Main/CSCArea/CSCPolicyQuery.xhtml. Successful framing of the target page indicates the application’s susceptibility to XFS.

Note that WebInspect will report only one instance of this check across each host within the scope of the scan. The other visible pages on the site may, however, be vulnerable to XFS as well and hence should be protected against it with an appropriate fix.

Implication

A Cross-Frame Scripting weakness could allow an attacker to embed the vulnerable application inside an iframe. Exploitation of this weakness could result in:

Hijacking of user events such as keystrokes
Theft of sensitive information
Execution of privileged functionality through combination with Cross-Site Request Forgery attacks

Fix

Browser vendors have introduced and adopted a policy-based mitigation technique using the X-Frame-Options header. Developers can use this header to instruct the browser about appropriate actions to perform if their site is included inside an iframe. Developers must set the X-Frame-Options header to one of the following permitted values:

- DENY

Deny all attempts to frame the page

- SAMEORIGIN

The page can be framed by another page only if it belongs to the same origin as the page being framed

- ALLOW-FROM origin

Developers can specify a list of trusted origins in the origin attribute. Only pages on origin are permitted to load this page inside an iframe

Developers must **also** use client-side frame busting JavaScript as a protection against XFS. This will enable users of older browsers that do not support the X-Frame-Options header to also be protected from clickjacking attacks.

Reference

HP 2012 Cyber Security Report
The X-Frame-Options header - a failure to launch

Server Configuration:
IIS
Apache, nginx

Specification:
X-Frame-Options IETF Draft

OWASP:
Clickjacking

Frame Busting:
Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites
OWASP: Busting Frame Busting

Classifications

CWE-352: Cross-Site Request Forgery (CSRF)
http://cwe.mitre.org/data/definitions/352.html

Kingdom: Security Features
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Privacy Violation: Inconsistent Feedback (11418)

Summary

When entering an invalid username or password during a login, an application may provide meaningful feedback through a response discrepancy. For the potential attacker, this discrepancy increases the chances of a successful brute force attack against the site's authentication.

Execution

Try logging into the application twice, once with an incorrect username and another time with an incorrect password. A generic error message for both attempts indicates a secure application. On the other hand, if there is a difference in the error messages, the application may be providing information that could be used in a brute-force attack.

Implication

Consider an application that takes an email address/username and a password. If the application provides different error messages for a non-existent email address and an incorrect password, it will enable an attacker to submit multiple email addresses and learn about the ones that are actually registered to the application. Being able to enumerate users in an application may enable an attacker to perform a more efficient brute force attack.

Fix

Applications should not indicate specifically whether a user account or password was incorrect, rather a very generic login failure message should be used. For example, during a new user registration or forgotten password function, no meaningful message should be returned to the requestor, rather an email should be sent to the account in question with details of how to reset a forgotten password or reclaim an account.

Reference

OWASP Guide to Authentication
https://www.owasp.org/index.php/Guide_to_Authentication

Username Enumeration Vulnerabilities
http://www.gnucitizen.org/blog/username-enumeration-vulnerabilities

Classifications

CWE-209: Information Exposure Through an Error Message
http://cwe.mitre.org/data/definitions/209.html

Kingdom: Security Features
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Web Server Misconfiguration: Server Error Message (10932)

Summary

A server error response was detected. The server could be experiencing errors due to a misbehaving application, a misconfiguration, or a malicious value sent during the auditing process. While error responses in and of themselves are not dangerous, per se, the error responses give attackers insight into how the application handles error conditions. Errors that can be remotely triggered by an attacker can also potentially lead to a denial of service attack or other more severe vulnerability. Recommendations include designing and adding consistent error handling mechanisms which are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Implication

The server has issued a 500 error response. While the body content of the error page may not expose any information about the technical error, the fact that an error occurred is confirmed by the 500 status code. Knowing whether certain inputs trigger a server error can aid or inform an attacker of potential vulnerabilities.

Fix

For Security Operations:

Server error messages, such as "File Protected Against Access", often reveal more information than intended. For instance, an attacker who receives this message can be relatively certain that file exists, which might give him the information he needs to pursue other leads, or to perform an actual exploit. The following recommendations will help to ensure that a potential

attacker is not deriving valuable information from any server error message that is presented.

- Uniform Error Codes: Ensure that you are not inadvertently supplying information to an attacker via the use of inconsistent or "conflicting" error messages. For instance, don't reveal unintended information by utilizing error messages such as Access Denied, which will also let an attacker know that the file he seeks actually exists. Have consistent terminology for files and folders that do exist, do not exist, and which have read access denied.
- Informational Error Messages: Ensure that error messages do not reveal too much information. Complete or partial paths, variable and file names, row and column names in tables, and specific database errors should never be revealed to the end user. Remember, an attacker will gather as much information as possible, and then add pieces of seemingly innocuous information together to craft a method of attack.
- Proper Error Handling: Utilize generic error pages and error handling logic to inform end users of potential problems. Do not provide system information or other data that could be utilized by an attacker when orchestrating an attack.

Removing Detailed Error Messages

Find instructions for turning off detailed error messaging in IIS at this link:

http://support.microsoft.com/kb/294807

For Development:

From a development perspective, the best method of preventing problems from arising from server error messages is to adopt secure programming techniques that prevent problems that might arise from an attacker discovering too much information about the architecture and design of your web application. The following recommendations can be used as a basis for that.

- Stringently define the data type (for instance, a string, an alphanumeric character, etc) that the application will accept.
- Use what is good instead of what is bad. Validate input for improper characters.
- Do not display error messages to the end user that provide information (such as table names) that could be utilized in orchestrating an attack.
- Define the allowed set of characters. For instance, if a field is to receive a number, only let that field accept numbers.
- Define the maximum and minimum data lengths for what the application will accept.
- Specify acceptable numeric ranges for input.

For QA:

The best course of action for QA associates to take is to ensure that the error handling scheme is consistent. Do you receive a different type of error for a file that does not exist as opposed to a file that does? Are phrases like "Permission Denied" utilized which could reveal the existence of a file to an attacker? Inconsistent methods of dealing with errors gives an attacker a very powerful way of gathering information about your web application.

Reference

Apache:
Security Tips for Server Configuration
Protecting Confidential Documents at Your Site
Securing Apache - Access Control

Microsoft:
How to set required NTFS permissions and user rights for an IIS 5.0 Web server
Default permissions and user rights for IIS 6.0
Description of Microsoft Internet Information Services (IIS) 5.0 and 6.0 status codes

Classifications

CWE-388: Error Handling
http://cwe.mitre.org/data/definitions/388.html

CWE-497: Exposure of System Data to an Unauthorized Control Sphere

http://cwe.mitre.org/data/definitions/497.html

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html

Privacy Violation: Autocomplete (11276)

Summary

Most recent browsers have features that will save password field content entered by users and then automatically complete password entry the next time the field are encountered. This feature is enabled by default and could leak password since it is stored on the hard drive of the user. The risk of this issue is greatly increased if users are accessing the application from a shared environment. Recommendations include setting autocomplete to "off" on all your password fields.

Please Note: Recent versions of most browsers, as noted below, now ignore the autocomplete="off" attribute for password fields in html forms. Users are allowed to decide the password policy at their own discretion using the password manager. Although setting is ineffective on these versions of browsers, it would continue to protect website users of earlier versions of these and other browsers that support this attribute.

Browsers NOT Supporting autocomplete="off":

Internet Explorer version 11 or above
Firefox version 30 or above
Chrome version 34 or above
For other browsers, please refer to vendor specific documentation

Execution

To verify if a password filed is vulnerable, first make sure to enable the autocomplete in your browser's settings, and then input the other fileds of the form to see whether the password is automatically filled. If yes, then it's vulnerable, otherwise, not. You may need to do it twice in case it is the first time you type in the credential in your browser.

Please Note: That some modern browsers no longer support this attribute as summarized above. Verification should be done using a browser that supports this attribute.

Implication

When autocomplete is enabled, hackers can directly steal your password from local storage.

Fix

From the web application perspective, the autocomplete can be turned at the form level or individual entry level by defining the attribute AUTOCOMPLETE="off".

Reference

Microsoft:
Autocomplete Security

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Security Features
http://www.hpenterprisesecurity.com/vulncat/en/vulncat/intro.html