WHITE PAPER | FEBRUAR 2015

Design einer CA Single Sign-On-Architektur für mehr Sicherheit

Wie Sie mithilfe vorhandener Einstellungen die Sicherheit Ihrer Architektur erhöhen

ca technologies

Inhaltsverzeichnis

Kurzfassung

Ausgangssituation

CA Single Sign-On (CA SSO) wird in der ganzen Welt vielfach eingesetzt, um Single Sign-On für ein breites Spektrum an Anwendungen mit unterschiedlichsten Sicherheitsanforderungen auf sichere Weise bereitzustellen. CA SSO stützt sich bei der Verwaltung von Anwender-Sessions auf mehrere Methoden. Die am meisten verwendete Option ist der Einsatz von Cookies. Oftmals wird CA SSO von den Administratoren so konfiguriert, dass diese Cookies an verschiedene Webserver gesendet werden, einschließlich der Webserver, die gar keinen Zugriff auf den Session-Cookie benötigen. Wenn alle Anwendungen so konfiguriert sind, dass ein Cookie an viele verschiedene Server geschickt wird, entsteht dadurch eine Schwachstelle innerhalb der Architektur, die es einem Angreifer ermöglicht, Cookies zu stehlen und die gestohlenen Session-Cookies dann erneut wiederzugeben, um sich als ein authentifizierter Anwender auszugeben.

Chancen

CA SSO verwendet den zum Patent angemeldeten Ansatz der erweiterten Session-Absicherung (Enhanced Session Assurance) mit Device DNA™, um die Gefahr der erneuten Wiedergabe von Sessions einzudämmen. CA SSO bietet verschiedene Einstellungen, mit denen sich die Sicherheit von Sessions erhöhen lässt, z. B. durch den Einsatz von „Host Only“-Cookies. Dies sind Session-Cookies, die dazu ausgelegt sind, lediglich zurück zu dem Host übertragen zu werden, der sie erstellt hat. Bei diesem Ansatz kann weiter gefasstes und anwendungsübergreifendes SSO für die End User bereitgestellt werden, und die Agenten können über einen zentralen Cookie-Provider zwischen den Domänen kommunizieren. Der Cookie-Provider kann einer Session, die sich in einem zentralen Session-Speicher befindet, einen Verweis zur einmaligen Verwendung zur Verfügung stellen, um die Session von einer Anwendung zur nächsten zu übergeben.

Nutzen

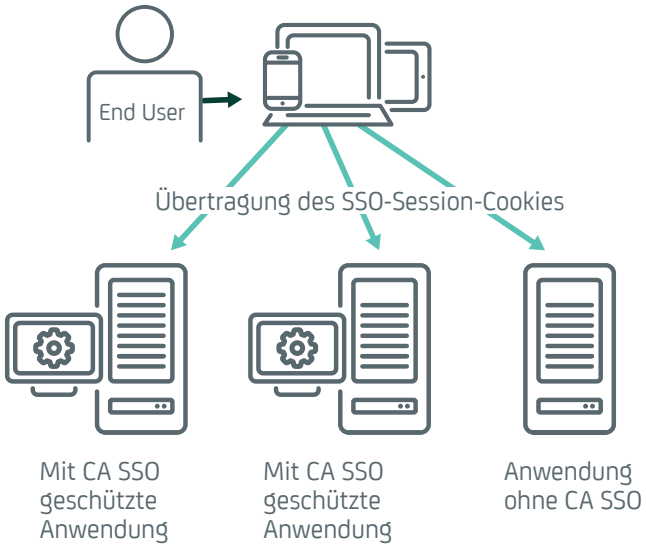
Mit CA SSO können Administratoren per Konfiguration festlegen, wie sich die Architektur bei allen oder einigen ihrer Anwendungen verhalten soll. Durch den Einsatz einer „Host Only“-Architektur kann die Sicherheit erhöht werden, weil sie es Angreifern deutlich erschwert, einen Session-Cookie abzufangen. Und dadurch verringert sich das Risiko, dass eine gestohlene Session an eine bestimmte Anwendung übergeben wird – bis die Leerlauf-Timeouts oder die Session-Absicherung die gestohlene Session auffinden und beenden können.

Abschnitt 1:

Warum der Schutz von CA SSO-Sessions so wichtig ist

„Session Hijacking“, auch bekannt unter der Bezeichnung „Cookie Hijacking“, ist keine neue Bedrohung. Sie ist allerdings zu einem praktisch permanenten Sicherheitsrisiko geworden, seit sich HTTP 1.1 zum Standard herausgebildet hat. Und sie ist nicht auf Session-Tokens von CA SSO beschränkt. Die OWASP Foundation zählt in ihren zehn größten Sicherheitsrisiken für Webanwendungen den Session-Diebstahl zum Angriff der Kategorie „A2 – Fehlerhafte Authentifizierung und fehlerhaftes Session Management“ (Broken Authentication and Session Management). Wenn die Session von einem Angreifer gestohlen wird, kann sie erneut wiedergegeben werden, und die Webanwendungen zeigen dem Angreifer Informationen im Kontext der gestohlenen Identität an. Darüber hinaus werden die Anforderungen des Angreifers als von einem gültigen, authentifizierten Anwender kommend protokolliert. Deshalb ist der Angriff so extrem schwierig aufzudecken.

In den meisten Bereitstellungen ist die CA SSO-Session so konfiguriert, dass sie über alle Webanwendungen hinweg genutzt werden kann, die die gleiche Cookie-Domäne (DNS) verwenden (z. B. jeder Webserver unter „ca.com“). Dies ist die einfachste Art sicherzustellen, dass der Cookie zu allen gewünschten CA SSO-Anwendungen gelangt. Zugleich stellt dies aber auch die größte Schwachstelle dar, da der Web Browser die CA SSO-Session den Anwendungen zur Verfügung stellt, die sie brauchen, und auch denjenigen, die sie nicht brauchen, und weil alle Anwendungen einen gemeinsamen Session-Token verwenden.

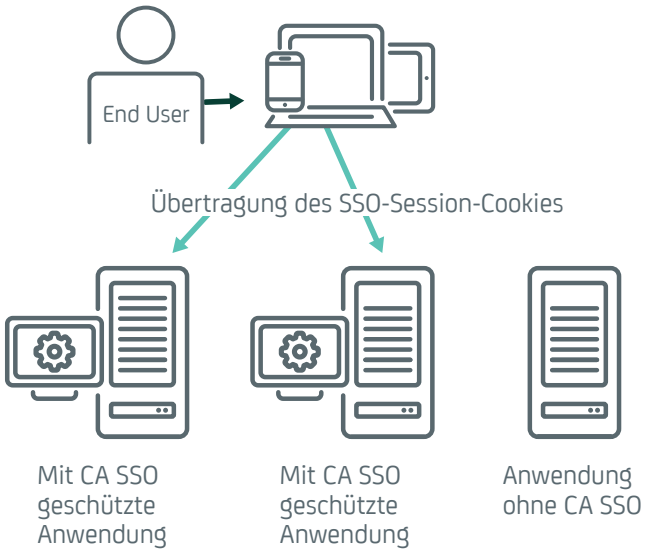


Abschnitt 2:

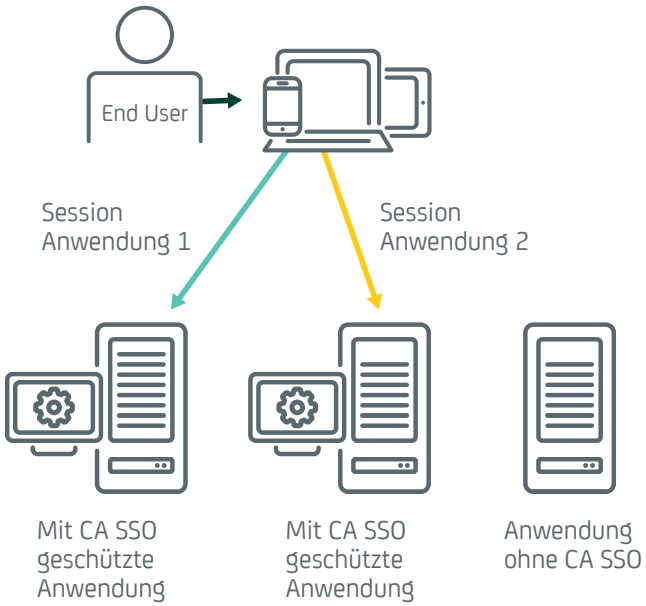
Wichtigste Einstellungen zum Ändern des Verhaltens von CA SSO

Es gibt verschiedene Einstellungen, mit denen sich das Sicherheitsniveau erhöhen lässt, um zu verhindern, dass eine Session ungewollt an Anwendungen übergeben wird, die sie gar nicht benötigen. Durch diese Einstellungen wird das typische Verhalten von CA SSO derart geändert, dass sicherere Szenarien unterstützt werden.

Die erste Verhaltensänderung besteht darin, dass der CA SSO-Cookie nur an die Anwendungen gesendet wird, die ihn benötigen. Dadurch wird verhindert, dass der Cookie an Sites übergeben wird, für die es gar keinen Grund gibt, die Session anzuzeigen. Dazu wird der Umfang der Cookie-Verteilung geändert, das heißt, Cookies werden nur an bestimmte Hosts und nicht an alle Server in einer gemeinsamen Domäne gesendet.



CA SSO kann auch so konfiguriert werden, dass für bestimmte Anwendungen spezifische Sessions verwendet werden, was das Risiko weiter minimiert, dass bei Diebstahl einer Session diese für mehrere Anwendungen genutzt werden kann.



Wichtigste Einstellungen zur Steuerung der Sicherheit von CA SSO-Sessions

Die folgenden Einstellungen gibt es in CA SSO schon seit mehreren Jahren. Sie ermöglichen Konfigurationsänderungen bezüglich des Verhaltens von Agenten und Gateways. Diese Einstellungen befinden sich alle im Agent Configuration Object (ACO).

CookieDomain

Über die Einstellung „CookieDomain" wird der Cookiedomänenwert definiert, der verwendet wird, um Cookies über den HTTP-Antwortheader „set-cookie" zu erstellen. Der Standardwert für diese Einstellung ist eine leere Zeichenfolge, die dem Agenten mitteilt, dass die Cookiedomäne vom HTTP_HOST-Header einer Anforderung basierend auf der unten beschriebenen Einstellung „CookieDomainScope" abgeleitet werden soll. Der Wert „NONE" gibt an, dass kein Cookiedomänenwert festgelegt werden soll. Dies entspricht der Definition eines „Server Only"-Cookies. Alternativ kann auch ein bestimmter Cookiedomänenwert angegeben werden (z. B. „.app.ca.com"). Die Methode der Festlegung eines bestimmten Cookiedomänenwerts sollte jedoch mit Vorsicht angewendet werden, da die für einen Cookie angegebene Domäne mit einem Teil der Domäne der Anforderung übereinstimmen muss, über die er ausgegeben wird. Der Cookiedomänenwert kann also kein beliebiger Wert sein. Wenn ein bestimmter Web-Agent Anforderungen, die an mehrere HTTP-Hosts gesendet werden, verarbeitet, sollte KEIN spezifischer Cookiedomänenwert verwendet werden. Dies ist auch nur in seltenen Fällen erforderlich.

CookieDomainScope

Die Einstellung „CookieDomainScope" steuert den Bereich einer Session, indem festgelegt wird, wie ein Cookiedomänenwert vom HTTP_HOST-Header einer Anforderung abgeleitet wird. Der Standardwert ist „0". Dieser Wert gibt den globalen Bereich an, der die Cookiedomäne auf oberster Domänenebene definiert (z. B. „ca.com"). Der Wert „1" ist nicht zulässig, da „.com" und „.net" usw. keine gültigen Cookiedomänen sind. „2" kommt der Angabe von „0" gleich. Werte über „2" geben einen definierteren Bereich an, sofern die Domäne von HTTP_HOST dies zulässt. So würde beispielsweise der Wert „0" oder „2" dazu führen, dass die Cookiedomäne „.ca.com" für den HTTP_HOST „myserver.security.ca.com" festgelegt wird. Der Wert „1" ist nicht zulässig (wird ignoriert; stattdessen wird der Standardwert „0" verwendet). Der Wert „3" würde zur Folge haben, dass die Cookiedomäne „.security.ca.com" festgelegt wird. Der Wert „4" würde „myserver.security.ca.com" zum Ergebnis haben. In diesem Fall ist es jedoch sinnvoller, für „CookieDomain" wie oben beschrieben den Wert „NONE" anzugeben. „CookieDomainScope" wird ignoriert, wenn „CookieDomain" auf „NONE" eingestellt wurde. Dies weist darauf hin, dass „Server Only"-Cookies verwendet werden sollen. Im Falle von „Server Only"-Cookies ist der Bereich IMMER der vollständige Wert von HTTP_HOST minus jedem angegebenen Portwert.

CookieProvider

Aufgrund der Tatsache, dass wir „Host Only"-Cookies verwenden und trotzdem SSO zwischen Anwendungen implementieren möchten, muss eine zentralisierte Site eines Identity Provider vorhanden sein, der die Session-Informationen an andere Anwendungen übergeben kann. Dies ist der CA SSO-Cookie-Provider. Dabei handelt es sich um einen zentralen Server, der dazu dient, Session-Informationen an andere, dezentrale Webanwendungen zu übermitteln. Als Cookie-Provider kann jedes Gateway bzw. jeder Agent von CA SSO fungieren. Agenten, die diesen Cookie-Provider nutzen, verwenden die Cookie-Provider-URL, die in der ACO-Einstellung angegeben ist.

EnableCookieProvider

Über die Einstellung „EnableCookieProvider" wird einem SSO-Gateway oder -Agenten mitgeteilt, dass es/er als Cookie-Provider fungieren kann. Es wird empfohlen, diese Einstellung für alle Agenten-ACOs zu deaktivieren, ausgenommen für den gewünschten Cookie-Provider. Dies kann einen Angreifer daran hindern, Berechtigungen zu eskalieren, wenn sie eine CA SSO-Session für eine Anwendung gestohlen haben und dann versuchen, diese zu nutzen, um sich Zugriff auf andere Anwendungen zu verschaffen.

StoreSessionInServer

Früher war es so, dass CA SSO-Cookie-Provider die Session in die Daten der HTTP-Abfragezeichenfolge als Teil einer Umleitung zum Endziel einfügten. Durch Einfügen dieser Daten in die Abfragezeichenfolge könnten Angreifer allerdings Zugriff auf die Session erlangen. Stattdessen könnte dem CA SSO-Cookie-Provider mitgeteilt werden, dass die Session in einem zentralisierten Session-Speicher gespeichert und dann ein Einmal-Verweis zur gespeicherten Session an die Abfragezeichenfolge übergeben werden soll. Dann würde der Anwendung, die die Session anfordert, diese Session über den Richtlinienserver bereitgestellt, mit dem sie kommuniziert, anstatt dass sie direkt aus der Abfragezeichenfolge gelesen wird. Dieser Ansatz gleicht dem SAML Artifact Profile.

LimitCookieProvider

Bei Verwendung eines zentralisierten Cookie-Providers kann dieser dazu genutzt werden, neue CA SSO-Session-Cookies für dezentrale Agenten zu erstellen oder es einem dezentralen Agenten zu erlauben, eine neue Session beim Cookie-Provider zu erstellen, sofern der Anwender direkt auf der dezentralen Seite angemeldet ist. Diese Einstellung kann erzwingen, dass alle Authentifizierungen innerhalb der Domäne des zentralen Cookie-Providers stattfinden, und Sessions ablehnen, wenn sie in dezentralen Anwendungen erstellt wurden. Ob diese Einstellung empfehlenswert ist, hängt von den Sicherheits- und Unternehmensrichtlinien ab. Wenn Sie einen zentralen Ort für alle Anmeldeseiten verwenden können, wird empfohlen, diese Einstellung zu nutzen.

TrackSessionDomain

Um sicherzustellen, dass eine CA SSO-Session nur für die Site verwendet wird, für die sie beabsichtigt ist, kann die ACO-Einstellung „TrackSessionDomain“ verwendet werden. Diese Einstellung weist den Web-Agenten an, die gewünschte Domäne einer Session innerhalb des Session-Cookies selbst zu verschlüsseln und zu speichern. Bei anschließenden Anforderungen vergleicht der Web-Agent die im Session-Cookie gespeicherte Domäne mit der Domäne der angeforderten Ressource. Wenn die Domänen nicht übereinstimmen, wird der Session-Cookie vom Web-Agenten abgelehnt.

TrackCPSessionDomain

Ein CA SSO-Cookie-Provider ist für die Verarbeitung der Transformation einer CA SSO-Session von einer Domäne zu einer anderen Domäne zuständig. Damit diese Transformation bei Verwendung von „TrackSessionDomain“ ordnungsgemäß abläuft, muss der Cookie-Provider angewiesen werden, die Domäne innerhalb der Session umzubenennen, sodass sie an anderen Stellen verwendet werden kann. Die Einstellung „TrackCPSessionDomain“ teilt dem Cookie-Provider mit, dass die Domäne seines Cookies vor der Transformation für eine andere Anwendung validiert werden muss. Dadurch wird ein Angreifer daran gehindert, den Cookie-Provider für die willkürliche Transformation von Cookies zwischen Domänen zu missbrauchen (z. B. das Senden eines gestohlenen „.app1.ca.com“-Cookies, der in „.app2.ca.com“ umgewandelt werden soll, an den Cookie-Provider).

ValidTargetDomain

Der Parameter „ValidTargetDomain“ identifiziert die gültigen Domänen und Hosts für dezentrale Systeme während der Verarbeitung. Bevor der Anwender umgeleitet wird, gleicht der Agent die Werte in der Umleitungs-URL mit den Domänen in diesem Parameter ab. Ohne diesen Parameter leitet der Agent den Anwender an Ziele in einer beliebigen Domäne um. Diese Einstellung wird verwendet, um Cross-Site-Angriffe durch Umleitungen zu den Anmeldeseiten, Cookie-Providern und Session-Absicherungs-URLs zu verhindern.

Abschnitt 3:

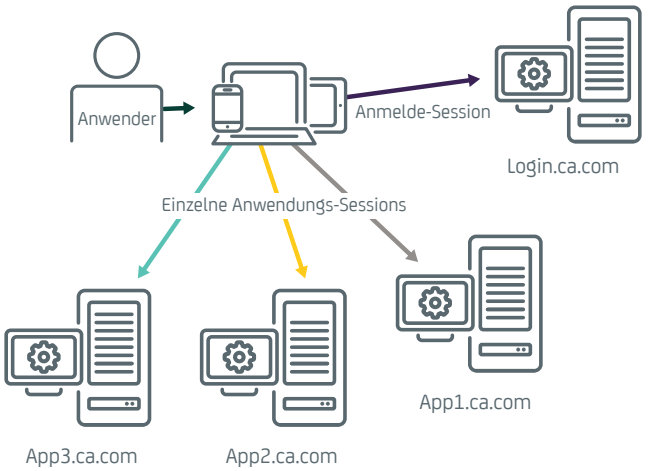
Design einer Architektur für maximale Sicherheit

Die Verwendung dieser Einstellungen begünstigt eine Architektur, in der jede Anwendung erzwungenermaßen eine separate Session hat, die eben nur für diese Anwendung genutzt werden darf. Dazu müssen alle Anwender an einer zentralen Stelle authentifiziert werden, wobei die SSO-Funktionalität aber bestehen bleibt.

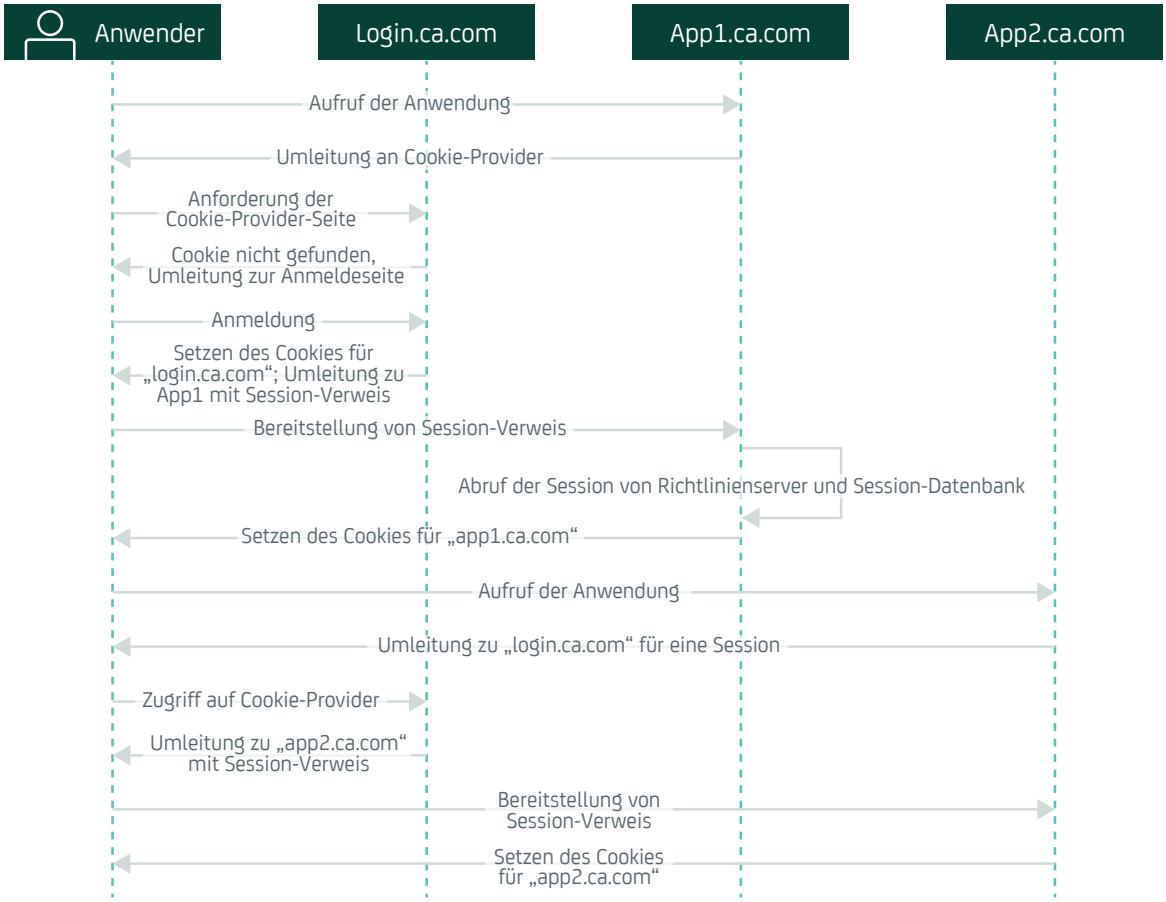
In diesem Beispiel gibt es eine zentrale Site, „login.ca.com“, die als unser Cookie-Provider dient und die Anmeldeseiten sowie mehrere Anwendungen hostet.

	Standardeinstellung	Login.ca.com	App1.ca.com	App2.ca.com	App3.ca.com
CookieDomain	"" (leere Zeichenfolge)	NONE	NONE	NONE	NONE
CookieDomainScope	0 (Bereich oberste Domäne verwenden)	Standard	Standard	Standard	Standard
CookieProvider		Standard	https://login.ca.com/siteminderagent/SmMakeCookie.ccc		
EnableCookieProvider	Ja	Ja	Nein	Nein	Nein
StoreSessionInServer	Nein	Ja	Ja	Ja	Ja
LimitCookieProvider	Nein	Ja	Nein	Nein	Nein
TrackSessionDomain	Nein	Ja	Ja	Ja	Ja
TrackCPSessionDomain	Nein	Ja	Standard	Standard	Standard
ValidTargetDomain	Alle Domänen ("")	App1.ca.com App2.ca.com App3.ca.com	Standard	Standard	Standard

Die obigen Einstellungen resultieren in einer Architektur, die wie folgt ausschaut:



Hier eine abstrakte Ansicht dazu:

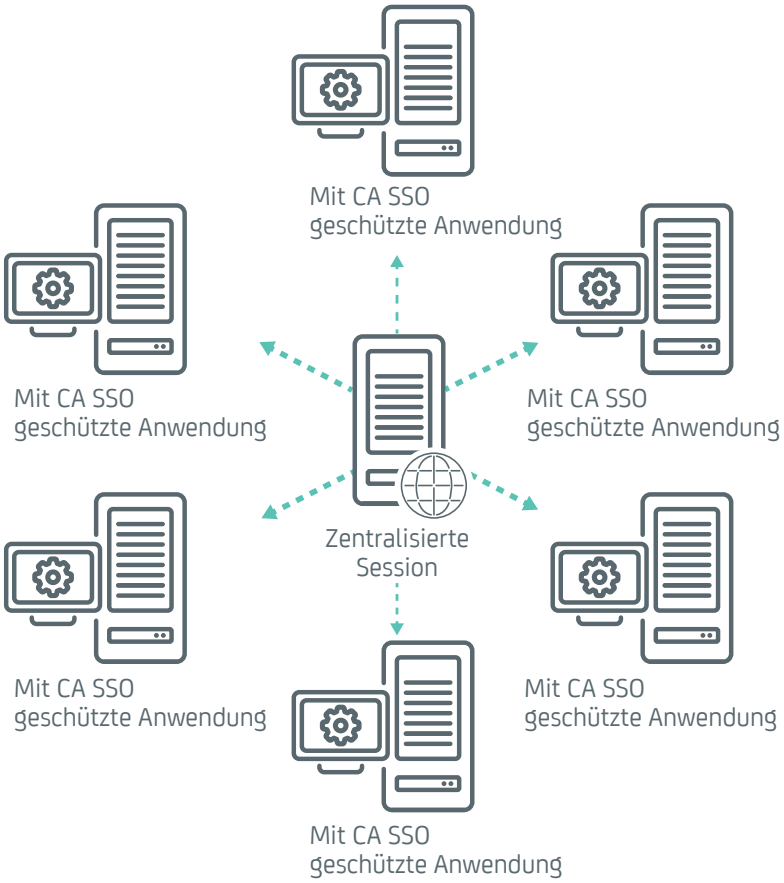


Diese Architektur, kombiniert mit anderen Kontrollen für CA SSO-Sessions (z. B. „SSL Only“- und „HTTP Only“-Cookies, Enhanced Session Assurance mit DeviceDNA für Device Fingerprinting sowie ordnungsgemäßen Richtlinien für einen Leerlauf/Session-Timeout), kann eine CA SSO-Umgebung effektiv absichern und so die Sicherheit der Sessions erhöhen, während die End User dennoch SSO nutzen können.

Abschnitt 4:

Schlussbemerkungen

Da die Unternehmen fortschrittliche risikobasierte Systeme für die mehrstufige Authentifizierung im Einsatz haben, wird die Sicherheit der Session-Tokens, die nach der Authentifizierung bereitgestellt werden, immer wichtiger, denn sie bilden die nächste logische Schwachstelle in der Infrastruktur. CA SSO kann mit „Host Only"-Cookies Single Sign-On und Sitzungsmanagement über verschiedene Sites implementieren und mithilfe von Device Fingerprinting prüfen, ob die Session tatsächlich von dem Host kommt, für den sie ursprünglich ausgegeben wurde. Der Einsatz einer Architektur, die „Host Only"-Cookies verwendet, erschwert den Diebstahl von Session-Cookies erheblich, da eine zentralisierte Sterntopologie anstelle einer einzelnen Session für die gesamte Domäne verwendet wird.



Wenn dennoch eine Session kompromittiert werden sollte, grenzt diese Architektur außerdem die damit verbundenen Risiken ein. Da jede Anwendung über einen separaten Session-Cookie verfügt, kann bei Diebstahl einer Session der gestohlene Session-Cookie nur für die Anwendung genutzt werden, für die er beabsichtigt war. Er kann also nicht missbraucht werden, um Zugriff auf andere Anwendungen zu erhalten, bis die gestohlene Session von der Session-Absicherung, von Timeouts oder anderen Kontrollen für ungültig erklärt wird.

Abschnitt 5:

Literaturhinweise

Top Ten-Liste von OSAWP: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Abschnitt 6:

Informationen zum Autor

Aaron Berman ist derzeit als Seniorberater für die Außendienstorganisation von CA Technologies tätig. Zu seinem Verantwortungsbereich gehören Produkt-, Vertriebs- und Kommunikationsstrategie sowie Tätigkeiten, die das Produktmanagement erweitern. Er verfügt über mehr als 15 Jahre Erfahrung in den Bereichen Problembehebung, Design, Implementierung und Definition von Strategien für Web Access Management-Lösungen, darunter beispielsweise der Entwurf von 100-Millionen-Anwender-Belastungstests für CA Single Sign-On (früher CA SiteMinder) und CA Identity Manager (früher CA IdentityMinder). Darüber hinaus leitete er mehrere Federation Interop-Events. Vor seiner Tätigkeit bei CA Technologies und Netegrity managte Aaron Berman die Support- und Presales-Services für Web Access Management-Lösungen für Raptor Systems/Axent Technology. Zuvor war Aaron Berman VP Principal Architect bei der Service-Organisation von CA Technologies. Aaron Berman hat einen Bachelor of Science in Computerwissenschaften der Syracuse University.

Weitere Informationen finden Sie unter **ca.com/de/secure-sso**

1 Lhttps://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project