- Planificación
- Instalación
- Configuración
- Autenticación
- Atributos
- Autenticación multifactor
- SSO & SLO
- Gestión de contraseñas
- Venta de entradas
- Servicios
- Registros y auditorias
- Monitoreo y Estadística
- Interfaz de usuario
- Gestión de flujo web
- Alta disponibilidad
- Protocolos
- Integración
- Desarrollador

Search

1. 5.2.x 2. / planificación 3. / Guia de seguridad

Guia de seguridad

CAS es un software de seguridad que proporciona un inicio de sesión único seguro basado en la Web para aplicaciones basadas en la Web. El inicio de sesión único proporciona beneficios para todos y cada uno en términos de seguridad y conveniencia: reduce la exposición de la contraseña a un agente de credenciales único y confiable, al tiempo que brinda acceso transparente a múltiples servicios sin inicios de sesión repetitivos. El uso de CAS generalmente mejora el entorno de seguridad, pero hay varias preocupaciones de configuración, política e implementación de CAS que deben considerarse para lograr una seguridad adecuada.

Problemas de información
El equipo de seguridad le pide que NO cree temas o publicaciones visibles públicamente para discutir lo que puede considerar una vulnerabilidad de seguridad. Para informar los problemas correctamente y aprender cómo se producen las respuestas, consulte esta guía .

Anuncios

- **Mar 6 2017 Divulgación de Vulnerabilidad**
- **24 de octubre de 2016 Divulgación de Vulnerabilidad**
- **Abr 8 2016 Vulnerabilidad Divulgación**

Consideraciones de seguridad del sistema

Los asuntos de seguridad de infraestructura a considerar pueden incluir los siguientes.

Transporte seguro (https)

Toda la comunicación con el servidor CAS DEBE ocurrir a través de un canal seguro (es decir, TLSv1). Hay dos justificaciones principales para este requisito:

1. El proceso de autenticación requiere la transmisión de credenciales de seguridad.
2. El ticket de concesión de tickets de CAS es un token de portador.

Dado que la divulgación de cualquiera de los datos permitiría ataques de suplantación, es de vital importancia asegurar el canal de comunicación entre los clientes de CAS y el servidor de CAS.

En la práctica, significa que todas las direcciones URL de CAS deben usar HTTPS, pero también significa que todas las conexiones desde el servidor de CAS a la aplicación deben realizarse mediante HTTPS:

- cuando el ticket de servicio generado se envía de vuelta a la aplicación en la url "service"
- cuando se llama un url de devolución de llamada de proxy.

Para ver la lista relevante de propiedades de CAS y ajustar este comportamiento, revise esta guía .

Conexiones a sistemas dependientes

CAS suele requerir conexiones a otros sistemas, como directorios LDAP, bases de datos y servicios de almacenamiento en caché. En general, recomendamos el uso de transporte seguro (SSL / TLS, IPSec) para aquellos sistemas donde sea posible, pero puede haber controles de compensación que hagan que el transporte seguro sea innecesario. Las redes privadas y las redes corporativas con controles de acceso estrictos son excepciones comunes, pero no obstante se recomienda el transporte seguro. La validación de la certificación del cliente puede ser otra buena solución para que LDAP brinde suficiente seguridad.

Como se indicó anteriormente, las conexiones a otros sistemas deben ser seguras. Pero si el servidor CAS se implementa en varios nodos, lo mismo se aplica al propio servidor CAS. Si se ejecuta un registro de tickets basado en caché sin ningún problema de seguridad en un solo servidor CAS, la sincronización puede convertirse en un problema de seguridad cuando se usan varios nodos si la red no está protegida.

Cualquier almacenamiento en disco también es vulnerable si no se asegura adecuadamente. El desbordamiento de EhCache al disco puede desactivarse para aumentar la protección, mientras que el mecanismo de datos de encriptación avanzada debe usarse para el almacenamiento en disco de la base de datos.

Características de seguridad dirigidas a la implementación

CAS admite una serie de funciones que pueden aprovecharse para implementar diversas políticas de seguridad. Las siguientes funciones se proporcionan a través de la configuración de CAS y la integración de clientes de CAS. Tenga en cuenta que muchas funciones están disponibles de forma inmediata, mientras que otras requieren una configuración explícita

Autenticación forzada

Muchos clientes de CAS y protocolos compatibles admiten el concepto de autenticación forzada por el cual un usuario debe volver a autenticarse para acceder a un servicio en particular. Los protocolos CAS admiten la autenticación forzada a través del parámetro renovar . La autenticación forzada proporciona seguridad adicional en la identidad del principal de una sesión de SSO, ya que el usuario debe verificar sus credenciales antes de acceder. La autenticación forzada es adecuada para servicios donde se desea o exige una mayor seguridad. Por lo general, la autenticación forzada se configura para cada servicio, pero la facilidad de administración del servicio brinda cierto soporte para implementar la autenticación forzada como una cuestión de política de seguridad centralizada. La autenticación forzada se puede combinar con la autenticación multifactor Características para implementar políticas de control de acceso arbitrarias específicas del servicio.

Autenticación pasiva

Algunos protocolos CAS admiten la autenticación pasiva cuando el acceso a un servicio protegido por CAS se otorga de forma anónima cuando se solicita. Los protocolos CASv2 y CASv3 admiten esta capacidad a través de la función de puerta de enlace . La autenticación pasiva complementa la autenticación forzada; donde la autenticación forzada requiere autenticación para acceder a un servicio, la autenticación pasiva permite el acceso al servicio, aunque de forma anónima, sin autenticación.

Autenticación de proxy

Proxy authentication, or delegated authentication, provides a powerful, important, and potentially security-improving feature of CAS. Proxy authentication is supported by the CASv2 and CASv3 protocols and is mediated by proxy tickets that are requested by a service on behalf of a user; thus the service proxies authentication for the user. Proxy authentication is commonly used in cases where a service cannot interact directly with the user and as an alternative to replaying end-user credentials to a service.

However, proxy tickets carry risk in that services accepting proxy tickets are responsible for validating the proxy chain (the list of services through which the end-user’s authentication have been delegated to arrive at the ticket validating service). Services can opt out of accepting proxy tickets entirely (and avoid responsibility for validating proxy chains) by simply validating tickets against the /serviceValidate validation endpoint, but experience has shown it’s easy to be confused about this and configure to unintentionally use the /proxyValidate endpoint yet not scrutinize any proxy chains that appear in the ticket validation response. Thus proxy authentication requires careful configuration for proper security controls; it is recommended to disable proxy authentication components at the CAS server if proxy authentication is not needed.

Historically any service could obtain a proxy-granting ticket and from it a proxy ticket to access any other service. In other words, the security model is decentralized rather than centralized. The service management facility affords some centralized control of proxy authentication by exposing a proxy authentication flag that can enabled or disabled on a per-service basis. By default registered services are not granted proxy authentication capability.

Credential Caching and Replay

The ClearPass extension provides a mechanism to capture primary authentication credentials, cache them (encrypted), and replay on demand as needed to access legacy services. While proxy authentication is recommended in lieu of password replay, it may be required to integrate legacy services with CAS. See the ClearPass documentation for detailed information.

Service Management

The service management facility provides a number of service-specific configuration controls that affect security policy and provide some support for centralized security policy. (Note that CAS has historically supported the decentralized security policy model.) Some highlights of service management controls:

- Authorized services
- Forced authentication
- Attribute release
- Proxy authentication control
- Theme control
- Service authorization control
- Multi-factor service access policy

The service management facility is comprised of a service registry containing one or more registered services, each of which specifies the management controls above. The service registry can be controlled via static configuration files, a Web user interface, or both. See the Service Management section for more information.

Authorized Services
As a security best practice, it is strongly recommended to limit the service management facility to only include the list of known applications that are authorized to use CAS. Leaving the management interface open for all applications may create an opportunity for security attacks.

SSO Cookie Encryption

A ticket-granting cookie is an HTTP cookie set by CAS upon the establishment of a single sign-on session. The cookie value is by default encrypted and signed via settings defined in CAS properties. While sample data is provided for initial deployments, these keys MUST be regenerated per your specific environment. Please see this guide for more info.

Password Management Secure Links

Account password reset requests are handled via a secured link that is sent to the registered email address of the user. The link is available only within a defined time window and the request is properly signed and encrypted by CAS. While sample data is provided for initial deployments, these keys MUST be regenerated per your specific environment.

Please see this guide for more info.

Protocol Ticket Encryption

Protocol tickets that are issued by CAS and shared with other applications such as service tickets may optionally go through a signing/encryption process. Even though the CAS server will always cross check ticket validity and expiration policy, this may be forced as an extra check to ensure tickets in transit to other applications are not tampered with and remain to be authentic. While sample data is provided for initial deployments, these keys MUST be regenerated per your specific environment.

Pay Attention
Encrypting and signing a generated ticket will, depending on the encryption method and algorithm used, increase the generated ticket length. Not all CAS clients are equipped to handle lengthy ticket strings and may get upset with you. Evaluate existing integrations before turning this on and consider whether this feature is truly needed for your deployment.

To see the relevant list of CAS properties, please review this guide.

Ticket Registry Encryption

Secure ticket replication as it regards clustered CAS deployments may be required to ensure generated tickets by CAS are not tampered with in transit. CAS covers this issue by allowing tickets to be natively encrypted and signed. While sample data is provided for initial deployments, these keys MUST be regenerated per your specific environment. Please see this guide for more info.

Administrative Pages Security

CAS provides a large variety of web interfaces that are aimed at system administrators and deployers. These screens along with a number of REST endpoints allow a CAS deployer to manage and reconfigure CAS behavior without resorting to native command-line interfaces. Needless to say, these endpoints and screens must be secured and allowed proper access only to authorized parties. Please see this guide for more info.

Ticket Expiration Policies

Ticket expiration policies are a primary mechanism for implementing security policy. Ticket expiration policy allows control of some important aspects of CAS SSO session behavior:

- SSO session duration (sliding expiration, absolute)
- Ticket reuse

See the Configuring Ticketing Components section for a detailed discussion of the various expiration policies and configuration instructions.

Single Sign-Out

Single sign-out, or single log-out (SLO), is a feature by which CAS services are notified of the termination of a CAS SSO session with the expectation that services terminate access for the SSO session owner. While single sign-out can improve security, it is fundamentally a best-effort facility and may not actually terminate access to all services consumed during an SSO session. The following compensating controls may be used to improve risks associated with single sign-out shortcomings:

- Require forced authentication for sensitive services
- Reduce application session timeouts
- Reduce SSO session duration

SLO can happen in two ways: from the CAS server (back-channel logout) and/or from the browser (front-channel logout). For back-channel logout, the SLO process relies on the `SimpleHttpClient` class which has a threads pool: its size must be defined to properly treat all the logout requests. Additional not-already-processed logout requests are temporarily stored in a queue before being sent: its size is defined to 20% of the global capacity of the threads pool and can be adjusted. Both sizes are critical settings of the CAS system and their values should never exceed the real capacity of the CAS server.

Login Throttling

CAS supports a policy-driven feature to limit successive failed authentication attempts to help prevent brute force and denial of service attacks. The feature is beneficial in environments where back-end authentication stores lack equivalent features. In cases where this support is available in underlying systems, we encourage using it instead of CAS features; the justification is that enabling support in underlying systems provides the feature in all dependent systems including CAS. See the login throttling configuration section for further information.

🔗Credential Encryption

To learn how sensitive CAS settings can be secured via encryption, please review this guide.

🔗CAS Security Filter

The CAS project provides a number of a blunt generic security filters suitable for patching-in-place Java CAS server and Java CAS client deployments vulnerable to certain request parameter based bad-CAS-protocol-input attacks. The filters are configured to sanitize authentication request parameters and reject the request if it is not compliant with the CAS protocol in the event that for instance, a parameter is repeated multiple times, includes multiple values, contains unacceptable values, etc.

It is STRONGLY recommended that all CAS deployments be evaluated and include this configuration if necessary to prevent protocol attacks in situations where the CAS container and environment are unable to block malicious and badly-configured requests.

🔗CORS

CAS provides first-class support for enabling HTTP access control (CORS). One application of CORS is when a resource makes a cross-origin HTTP request when it requests a resource from a different domain than the one which the first resource itself serves. This should help more with CAS-enabled applications are accessed via XHR/Ajax requests.

To see the relevant list of CAS properties and tune this behavior, please review this guide.

🔗Security Response Headers

As part of the CAS Security Filter, the CAS project automatically provides the necessary configuration to insert HTTP Security headers into the web response to prevent against HSTS, XSS, X-FRAME and other attacks. These settings are presently off by default. To see the relevant list of CAS properties and tune this behavior, please review this guide.

To review and learn more about these options, please visit this guide.

🔗Spring Webflow Sessions

The CAS project uses Spring Webflow to manage and orchestrate the authentication process. The conversational state of the webflow used by CAS is managed by the client which is then passed and tracked throughout various states of the authentication process. This state must be secured and encrypted to prevent session hijacking. While CAS provides default encryption settings out of the box, it is STRONGLY recommended that all CAS deployments be evaluated prior to production deployments and regenerate this configuration to prevent attacks.

🔗Long Term Authentication

The long term authentication feature, commonly referred to as “Remember Me”, is selected (usually via checkbox) on the CAS login form to avoid re-authentication for an extended period of time. Long term authentication allows users to elect additional convenience at the expense of reduced security. The extent of reduced security is a function of the characteristics of the device used to establish a CAS SSO session. A long-term SSO session established from a device owned or operated by a single user is marginally less secure than a standard CAS SSO session. The only real concern would be the increased lifetime and resulting increased exposure of the CAS ticket-granting ticket. Establishing a long-term CAS SSO session from a shared device, on the other hand, may dramatically reduce security. The likelihood of artifacts from previous SSO sessions affecting subsequent SSO sessions established by other users, even in the face of single sign-out, may increase the likelihood of impersonation. While there is no feasible mitigation for improving security of long-term SSO sessions on a shared device, educating users on the inherent risks may improve overall security.

It is important to note that forced authentication supersedes long term authentication, thus if a service were configured for forced authentication, authentication would be required for service access even in the context of a long-term session.

Long term authentication support must be explicitly enabled through configuration and UI customization during the installation process. Thus deployers choose to offer long-term authentication support, and when available users may elect to use it via selection on the CAS login form.

🔗Warn

CAS supports optional notification of service access during an established SSO session. By default CAS transparently requests tickets needed for service access and presents them to the target service for validation, whereby upon successful validation access to the service is permitted. In most cases this happens nearly instantly and the user is not aware of the CAS authentication process required to access CAS-enabled services. There may be some security benefit to awareness of this process, and CAS supports a warn flag that may be selected by the user on the CAS login screen to provide an interstitial notification page that is displayed prior to accessing a service. By default the notification page offers the user an option to proceed with CAS authentication or abort by navigating away from the target service.