CTSC-OSiRIS Collaborative Design Review of OSiRIS Access Assertions

Using the OAuth 2.0 Threat Model (RFC 6819)

31 Mar 2017

For Public Distribution

Jim Basney[1], Jayashree Ajay Candadai, Terry Fleury, Patrick Gossman, Mike Gregorowicz[2], Scott Koranda, Shawn McKee, Ben Meekhof, and Ezra Kissel

[1] CTSC Engagement Lead, jbasney@illinois.edu
[2] OAA Principal Architect, michael.gregorowicz@wayne.edu

About CTSC

The mission of CTSC is to provide the NSF community with a coherent understanding of cybersecurity, its importance to computational science, and what is needed to achieve and maintain an appropriate cybersecurity program.

Acknowledgments

This document is a product of the Center for Trustworthy Scientific Cyberinfrastructure (CTSC). CTSC is supported by the National Science Foundation under Grants Numbered OCI-1234408 and ACI-1547272. For more information about the Center for Trustworthy Scientific Cyberinfrastructure please visit: http://trustedci.org/. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Using & Citing this Work

Please cite as: Jim Basney, Jayashree Ajay Candadai, Terry Fleury, Patrick Gossman, Mike Gregorowicz, Scott Koranda, Shawn McKee, Ben Meekhof, and Ezra Kissel. CTSC-OSiRIS Collaborative Design Review of OSiRIS Access Assertions, March 2017. http://hdl.handle.net/2022/21307

Table of Contents

Executive Summary **3**

1 Engagement Process **3**

2 Factual Summary **3**

2.1 OSiRIS Use Cases 3

2.2 Use of Internet2 COmanage 4

2.3 OAA Design 5

2.4 OAA Threat Model Analysis 5

2.5 OAA Integrations: Ceph, NMAL 8

3 Recommendations **9**

4 Conclusion **9**

Executive Summary

The 2016-2017 CTSC-OSiRIS collaborative design review of OSiRIS Access Assertions produced a set of security recommendations documented in this report that the OSiRIS project plans to implement in its deployed cyberinfrastructure. CTSC identified no significant weaknesses in its review of the initial design of the OSiRIS access control system.

1 Engagement Process

After initial discussions with CTSC staff, representatives of the MI-OSiRIS project submitted a CTSC engagement application in June 2016. From August to October 2016, CTSC and OSiRIS staff developed the engagement plan, with the goal of conducting a joint design review of the OSiRIS Access Assertion (OAA) system. CTSC staff conducted the engagement from October 2016 to March 2017 via a series of hour-long phone calls with OSiRIS staff to discuss and review the OAA design. This report documents the outcomes of those discussions.

OAA design documents were the primary source materials used in the review. At the time of the review, the OAA system was in an early design and implementation phase, giving the group the opportunity to consider a variety of design options and give input to design decisions, in contrast to an after-the-fact security evaluation of an implemented system.

2 Factual Summary

MI-OSiRIS (Multi-Institutional Open Storage Research Infrastructure) is a pilot project funded by the NSF CC*DNI DIBBs program to evaluate a software-defined storage infrastructure for the primary Michigan research universities. OSiRIS is building a distributed, multi-institutional storage infrastructure that will allow researchers at any of three Michigan research university campuses to read, write, manage and share their data directly from their computing facility locations. The OSiRIS Access Assertion (OAA) system performs data access control in OSiRIS, using SAML for authentication and JSON Web Tokens (JWTs) for authorization.

2.1 OSiRIS Use Cases

The engagement team discussed two categories of use cases for the OSiRIS system: 1) distributed access to scientific data using Ceph and 2) network discovery, monitoring, and management using perfSONAR for reliable and high-performance use of Ceph across the network. The former target science users, and the latter target network engineers. The OSiRIS design includes a common authentication and authorization mechanism across these use cases, supporting federated campus authentication via Internet2's InCommon service and group-based access control (with delegated sub-groups) using Internet2's COmanage software.

Ceph provides three core interfaces: object store (S3 API), block device, and filesystem. An OSiRIS web portal enables users to obtain access to these Ceph interfaces and manage access control. Users first set up access via their web browser, then access Ceph interfaces outside the browser, via filesystem mounts, REST APIs, command-line tools, and other applications. OSiRIS also provides Globus endpoints for data sharing/transfer outside the Ceph cluster(s).

OSiRIS manages access to scientific data following the virtual organization (VO) model. OSiRIS operators establish a VO for each supported science domain/project and allocate storage to that VO. VOs manage their own membership, including defining groups for access control. System administrators on shared compute clusters configure Ceph mount points for the VOs that use those clusters, so their compute jobs can access the scientific data according to VO authorization. Science users typically access Ceph via direct mount or S3 API from these shared compute clusters as well as leverage Globus and/or other local mechanisms to transfer data to/from their desktop and the clusters.

The Network Management Abstraction Layer (NMAL) gives network engineers the ability to view operational network information and launch network probes for network discovery and performance analysis. The group of network engineers that support the OSiRIS operational infrastructure form their own virtual organization which determines access rights to the NMAL capabilities. Access control ensures that detailed network topology information and the ability to launch potentially disruptive network probes is restricted to the trusted group of network engineers.

2.2 Use of Internet2 COmanage

Internet2's COmanage software is a collaboration management platform that enables virtual organizations (VOs) to manage groups and roles via extensible enrollment and provisioning workflows. COmanage is multi-tenant, supporting multiple Collaborative Organizations (COs) each with independent user memberships, Collaborative Organization Units (COUs), groups, and roles. To support collaboration across VOs, OSiRIS plans to configure one CO in COmanage with a COU for each science VO. That will enable scientists to enroll once in the OSiRIS CO then join OSiRIS VOs (COUs) according to VO policies and procedures set by VO (COU) administrators.

COmanage implements a central component of the OAA design. Science users and administrators use COmanage to define the groups and roles that control access to their scientific data. Likewise, network administrators define groups and roles in COmanage to

control access to sensitive network data and probes. The Internet2 Shibboleth software authenticates users to COmanage, allowing scientists and network administrators to log in using their campus credentials via the InCommon federation. COmanage provisions user and group information to LDAP for use by other components of the OAA system (described below).

2.3 OAA Design

The OSiRIS Access Assertion (OAA) system consists of two main components: 1) a Resource Authority (RA) that issues digitally signed OSiRIS Access Grants (OAGs) and OSiRIS Access Assertions (OAAs) for access to the resource(s) it controls, and 2) a Central Authority (CA) that issues OSiRIS Access Requests (OARs) on behalf of users to RAs, stores OAAs signed by RAs, and issues digitally signed OSiRIS Access Tokens (OATs) and OSiRIS Refresh Tokens (ORTs) containing OAAs.

OAR	OSiRIS Access Request
OAG	OSiRIS Access Grant
OAA	OSiRIS Access Assertion
OAT	OSiRIS Access Token
ORT	OSiRIS Refresh Token

Table 1. There are five OSiRIS token types.

The CA integrates with COmanage to obtain user group/role information to include in OARs. Based on the user's identity, groups, and roles, the RA issues OAGs indicating what resources are available to the user. The OAG includes the "sub" or subject to which access is being granted, the "kind" of access (e.g., read or write), and the "resource" available for access (e.g., Ceph mount point). The subject of the OAG may be one or more named individuals, groups, or roles.

The OAA system uses concepts from OAuth (RFC6749), including JSON Web Tokens (RFC7519) and the practice of issuing shorter-lived access tokens and longer-lived refresh tokens. This fact inspired the engagement team to use the OAuth 2.0 Threat Model and Security Considerations (RFC6819) as an evaluation framework for the OAA system, as documented in the next section.

2.4 OAA Threat Model Analysis

In applying the RFC 6819 Threat Model to the OAA design we made the following observations:

- OAA clients (e.g., mount.osiris) are analogous to OAuth public clients, since users run OAA clients directly rather than accessing OSiRIS resources via a third-party web application. This is in contrast to OAuth confidential clients, which have their own credentials that are independent of the credentials of the user or resource owner. For this reason our RFC 6819 analysis did not consider topics specific to confidential clients.
- OAA's CA is analogous to OAuth's Authorization Server, which authenticates the user and issues access/refresh tokens.
- OAA's RA is analogous to OAuth's Resource Server, which grants access to resources based on the contents of the client's access tokens.

This correspondence between OAA components and OAuth components enabled a relatively straightforward application of the RFC 6819 threat model to OAA, with considerations for clients (RFC 6819 Section 4.1), user authorization (RFC 6819 Section 4.2), token issuance (RFC 6819 Section 4.3), client authorization (RFC 6819 Section 4.4), token refresh (RFC 6819 Section 4.5), and resource access (RFC 6819 Section 4.6).

Client threats primarily deal with exposure of tokens due to compromised or malicious clients. Based on RFC 6819 Section 4.1 recommendations we made the following observations:

- While OAA clients, being "public", do not have their own secrets, it would still be valuable for clients to locally generate and maintain a unique client identifier to prevent accidental misuse by one client of tokens issued to a different client.
- OAA's CA should include the ability to revoke OATs/ORTs in case of client compromise.
- OAGs, being authorization tokens rather than impersonation tokens, have the ability to limit token scope (i.e., following the principle of least privilege). As an implementation matter, RAs should not issue OAGs with overly broad privileges, to limit the impact of a client compromise.
- Keep OATs in transient memory to protect from disclosure. Use ORTs to obtain new OATs as needed.
- Follow the recommendations from the OAuth 2.0 for Native Apps specification.

User authorization threats focus on the process of the user authenticating to the system and obtaining authorization for the client to access resources. Based on RFC 6819 Section 4.2 recommendations we made the following observations:

- Use of InCommon federated authentication leverages existing university anti-phishing countermeasures. Universities train users to enter campus passwords only at a well-known campus identity provider. Also, many InCommon members are now

deploying multi-factor authentication at their identity providers. OAA use of SAML Web Browser Single Sign-On via InCommon avoids introducing OSiRIS-specific phishing risks.

- To protect against users unintentionally granting too much scope (privileges) to clients:
    - Careful implementation effort should be applied to the OAA user consent interface, to clearly explain to the user what access is being requested/granted.
    - The CA should include sanity checks on scope requests based on client characteristics, e.g., it makes sense for a mount.osiris client to request read/write access but not shell access.

Threats to the token issuance process include eavesdropping attacks and compromise of the token server. Based on RFC 6819 Section 4.3 recommendations we made the following observations:

- Require TLS to protect against token disclosure by network eavesdropping attacks. For non-browser clients, take special care to keep TLS configurations/libraries up-to-date. The current OAA implementation relies on Perl's mozilla-ca package for this.
- To protect against compromise of secrets on the CA (token server):
    - Avoid storing secrets when possible, e.g., store token IDs/hashes for validity checking but do not store sensitive token contents after issuance.
    - Carefully protect the CA's signing key. Support CA signing key revocation. Periodically update the CA signing key using a graceful key roll-over scheme (e.g., storing "previous" signing certificates in LDAP).

Threats to client authorization include malicious client software and browser-based attacks. Based on RFC 6819 Section 4.4 recommendations we make the following observations:

- To protect against malicious client software, the CA should prompt the user for consent when requesting client authorization.
- Prevent clickjacking by using X-FRAME-OPTIONS or JavaScript frame-busting in web interfaces.
- To protect against abuse of browser sessions, consider notifying users by an alternate method (email, text) for significant events (initial registration, new authorizations granted). Make notifications configurable so they can be adjusted based on usage and do not become annoying.
- Give users the ability to view the status of their current valid tokens and information about the clients they are issued to, so conscientious users can detect and report compromises and proactively revoke tokens that are no longer needed.
- To prevent exhaustion of resources, place configurable limits on the number of requests per user and resource.

Due to the longer validity period of refresh tokens, additional mitigations are recommended in RFC 6819 Section 4.5. Specifically for ORTs we make the following observations:

- Bind ORTs to client identity information to make use by other clients more difficult. Consider IP-binding of ORTs for clients with stable IP addresses (e.g., not mobile or using NAT).
- To proactively protect against misuse of old ORTs, re-issue fresh refresh tokens upon use.

When accessing protected resources, threats include malicious resource providers, denial of service attacks, and disclosure of sensitive information. Based on RFC 6819 Section 4.6 recommendations we make the following observations:

- To prevent misuse of bearer tokens by malicious resource providers, clients should verify the resource URI in the token matches the address of the resource before using a token with that resource.
- Use the resource provider's public key to encrypt any sensitive data included in bearer tokens.
- For OAGs that allocate consumable resources, track token serial numbers to prevent overuse by replay attacks.
- Use HTTP cache-control headers to minimize risk of token exposure from HTTP caches.
- Do not log sensitive parameters. Use POST instead of GET for sensitive operations, because GET parameters are often logged by HTTP servers.

2.5 OAA Integrations: Ceph, NMAL

CTSC and OSiRIS staff also discussed scenarios for Ceph and NMAL integration with OAA to understand how the OAA functionality will meet Ceph and NMAL access control needs.

OSiRIS plans to integrate OAA with Ceph via a gatekeeper web service that consumes OAA tokens and creates ephemeral Ceph access keys to grant access at the Ceph layer. The mount.osiris client interacts with a OAA client service daemon process to obtain OAA tokens from the CA (if tokens aren't already cached) then interacts with the gatekeeper service to obtain Ceph access keys (over HTTPS). Then mount.osiris can call mount.ceph to mount the filesystem using the ephemeral Ceph access keys.

OAA can be integrated directly into the NMAL web applications (Unified Network Information Service, Basic Lightweight Periscope Probes, etc.). NMAL access control will be primarily at the

REST endpoint layer (e.g., /services, /metadata, /measurements, etc.). Since NMAL includes network collection agents, running as persistent services, it needs support from OAA for long-lived refresh tokens with "robot" or service identities. These robot identities can be managed via COmanage and provisioned to a separate LDAP search base to keep them separate from OSiRIS user identities.

Overall, the group found that the OAA design provides the functionality and interfaces needed for Ceph and NMAL integration.

3 Recommendations

We supplement the RFC 6819 threat assessment with the following general recommendations:

- Apply standard operational security controls to OAA services, with special focus on protecting the Central Authority's signing key(s). Consult CTSC's Security Commodity IT in Scientific CI Projects: Baseline Controls and Best Practices for guidance.
- This design review is not a substitute for a comprehensive cybersecurity program. CTSC's Guide to Developing Cybersecurity Programs for NSF Science and Engineering Projects provides recommendations and templates.
- For cases where science users do not have a home campus identity provider (IdP) that is part of the InCommon federation, recommend use of an unaffiliated IdP or IdP of Last Resort (IoLR). The REFEDS IoLR working group provides a list.
- OSiRIS should consider applying for a follow-on CTSC engagement after implementation is complete, to review design changes that occurred during implementation and initial deployment.

4 Conclusion

Conducting this review early in the design phase of OSiRIS Access Assertions enabled a collaborative exchange between CTSC and OSiRIS staff, resulting in a design that follows IETF-developed recommendations and addresses a comprehensive threat model. We expect the recommendations in this report to be applicable to other collaborative scientific computing infrastructures that share similarities with OSiRIS.

OSiRIS open source software can be found at https://github.com/MI-OSiRIS.