CENTER FOR TRUSTWORTHY
SCIENTIFIC CYBERINFRASTRUCTURE

Suggested Security Practices for SciGaP: A Preliminary Report

June 15, 2014

For Public Distribution

Randy Heiland, Jim Basney, Von Welch

About CTSC

The mission of the Center for Trustworthy Scientific Cyberinfrastructure (CTSC, trustedci.org) is to improve the cybersecurity of NSF science and engineering projects, while allowing those projects to focus on their science endeavors. This mission is accomplished through one-on-one engagements with projects to solve their specific problems, broad education, outreach and training to raise the practice-of-security across the community, and looking for opportunities for improvement to bring in research to raise the state-of-practice.

Acknowledgments

CTSC's engagements are inherently collaborative; the authors would like to thank the SciGaP team for the collaborative effort that made this document possible.

This document is a product of the Center for Trustworthy Scientific Cyberinfrastructure (CTSC). CTSC is supported by the National Science Foundation under Grant Numbers OCI-1234408. 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

Cite this work using the following information:
R. Heiland, J. Basney, and V. Welch, "Suggested Security Practices for SciGaP: A Preliminary Report," Center for Trustworthy Scientific Cyberinfrastructure, trustedci.org, June 2014.

More information about this engagement can be found at the following URL:
http://trustedci.org/scigap

Introduction

The Science Gateway Platform (SciGaP, scigap.org) will provide services to help science domain communities create new and maintain and improve existing Science Gateways. SciGaP, via Apache Airavata[1], will use the Apache Thrift framework (thrift.apache.org), a language-independent, statically typed interface definition language (IDL), to generate the services, as well as the client software development kits (SDKs) to access those services. Several languages are supported: C/C++, Java, PHP, Python, Ruby, and more.

Thrift, originally developed internally at Facebook, became open sourced in 2007 and went on to become a top-level project at the Apache Software Foundation in 2010. As with any Apache project, it encourages and promotes community participation and has a well-defined process for developers who want to contribute code.

Even though Thrift is still a relatively new project, it is used by some well-known applications, including Facebook and Evernote (evernote.com). In another report, we have provided an overview and best practices for the use of Thrift in Evernote. Figure 1 depicts a high-level use case diagram of the Evernote and SciGaP architectures.

[1] Airavata is a toolkit for managing workflows for computational resources.

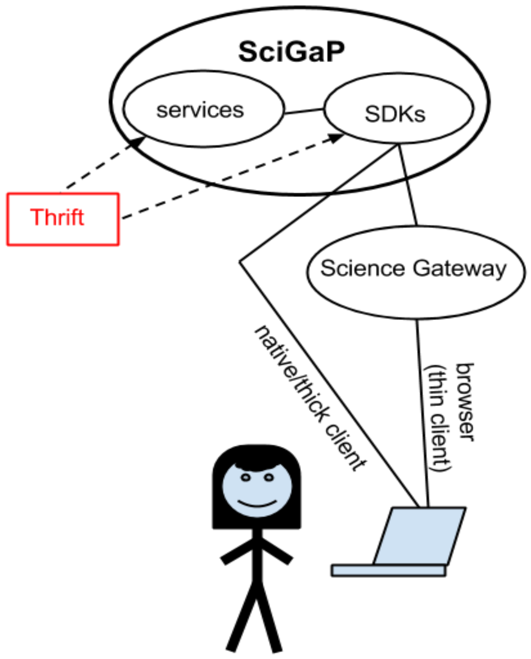

Figure 1. High-level use case scenarios of SciGaP

Differences Between Applications and Gateways

The main difference between an application/thick client and a gateway is that end users (and other entities, perhaps malicious) have access to applications and can decompile or otherwise reverse engineer them. This means that applications, unlike gateways, are not suitable entities for holding secrets or other sensitive information that should not be available to the user of the application.

In practice, this means sensitive information should be stored in services invoked by the application, with the service acting on behalf of the application in an authorized manner based on the authenticated identity of the application user.

Another difference is that science gateways can use browser-based authentication methods, such as OpenID Connect (OAuth) and SAML/InCommon, while thick clients must either use non-browser authentication (password, Kerberos, etc.) or launch a web browser for authentication (i.e., the thick client could launch a web browser to establish OAuth tokens that can subsequently be used by the application outside the browser).

Additionally, the science gateway introduces a layer between SciGaP and the client, so SciGaP is not able to directly authenticate the user at the client. Instead, SciGaP must trust the science gateway to properly authenticate the user, and the science gateway must delegate credentials to SciGaP as needed so SciGaP can act on the user's behalf.

The Science Gateway Security Model

A key component of the science gateway security model is the science gateway acting as a middle tier that mediates the user's access to valuable resources (such as supercomputers). Rather than interacting with the supercomputer directly via the command-line (Figure 2), the user may interact with the science gateway web portal via a web browser (Figure 3). Science gateways can also support thick clients (Figure 4). To maintain the science gateway model, the thick client interacts with a "middleware server", rather than connecting directly from the thick client to the supercomputer. The portal in Figure 3 and the middleware server in Figure 4 serve the same role in the model, namely, to mediate access (i.e., provide a gateway) to the underlying valuable resource. Figure 4 also illustrates how a portal can be layered on top of the middleware server to provide both thick client and browser interfaces to the same science gateway services, which appears to match closely the SciGaP Thrift model. The GridChem science gateway (www.gridchem.org) also combined a thick client and middleware server in this way.

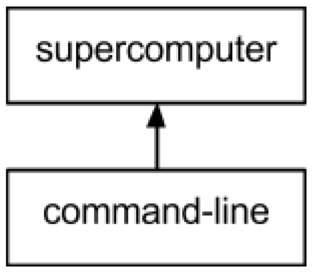

Figure 2: traditional

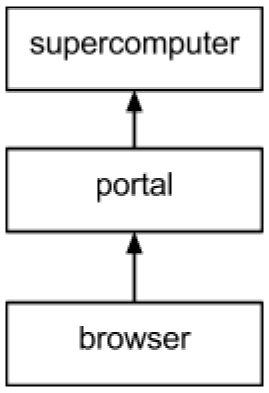

Figure 3: gateway portal

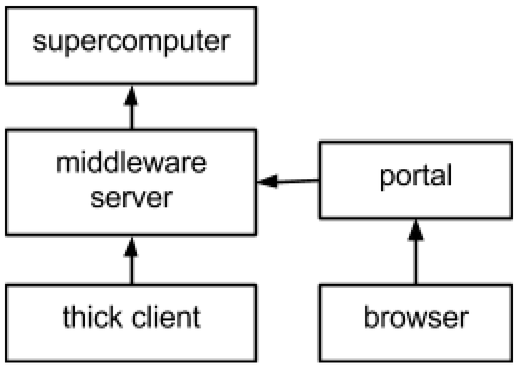

Figure 4: thick-client gateway (plus portal)

Suggested Security Practices

Comprehensive and Simple SDK

The (SciGaP/Airavata) code generated by Thrift (both services and SDKs) should be enhanced to handle as much security as possible, as simply as possible, in order to maximize good security practices by the community using it. Ideally, it would handle all security transparently, only requiring users with exceptional needs to have to override or change its behavior.

Use of Encryption and Server Authentication (HTTPS)

Use of encryption and server authentication (HTTPS presumably) is recommended for all communications to provide privacy and security. Clients must properly authenticate the server identity (i.e., via standard TLS server certificate checks) to ensure the encrypted channel is established with the proper server and not a man-in-the-middle.

Science Gateway Authentication

As shown in Figure 1, SciGaP will have two main use cases, thick clients and science gateways (web portals). For science gateways, there is already a variety of authentication methods in use as described in [1]. A challenge here may be that supporting all these different methods is difficult and SciGaP may need to select a subset to focus on and implement well.

Third-party Application Authentication

Authentication by a third-party application is inherently different than by a science gateway since end-users (and other parties) will have access to the third-party application and can decompile or otherwise reverse engineer it. This means that storing long-term secrets and other sensitive in the third-party application is infeasible from a security perspective.

The typical approach is to have the user of the application authenticate through the application to the service run by SciGaP, the result of that authentication being a token of some sort that provides authentication while relieving the user of having to repeated enter their password. Explicitly, storing of the user’s password by the application is not recommended.

A common mechanism to implement this approach is to use OAuth, with the application acting in the role of an OAuth client to authenticate the user to the Science Gateway (acting as an OAuth service provider). Note that the best practice for this case is to store the OAuth client token on the SciGaP service and have it accessed indirectly by the application [4].

Authentication of Privileged Users

We recommend that privileged users for SciGaP and science gateways utilize two-factor authentication to provide resistance to password compromise.

Community Education

No matter how comprehensive the SDK is, there will be some security practices that the community will need to follow operationally and on which they should be educated. Many of these are described in Section V of [3].

Software Security

All components of the system - applications, Science Gateways, and the SciGaP services - should have proper software implementation to be resilient to attacks. From a practical standpoint, the prioritization, from most to least important, would be SciGaP services, followed closely by Science Gateways and then applications. The SciGaP services and science gateways, by their nature of being online are more susceptible and likely to be attacked. For this reason, we suggest that the SciGaP team perform routine web application scanning (Qualys provides one such application). Applications need to be resilient to malicious services and man-in-the-middle attacks, which are harder to implement. An attack on an application also only impacts a single user.

Since SciGaP involves considerable software *development*, we recommend the following software security practices:

- follow "standard" software engineering best practices, e.g., architecture design, continuous integration (including unit testing), peer review, etc.
- be mindful of secure programming and use an IDE/plugin(s) that can offer static analysis of code, e.g. Eclipse/FindBugs (rf. https://www.owasp.org/index.php/Source_Code_Analysis_Tools)
- run static analysis tool(s) on entire code base[2]

Regarding this last bullet point, there are several options available, both commercial and open source. We list a few in the Related Links section at the end. For example, CTSC took a snapshot of the Airavata code (from github) in March 2014, uploaded it to the SWAMP site and ran the PMD tool (http://pmd.sourceforge.net/) against it. The results included over 40000 "weaknesses" (http://cwe.mitre.org/) that included:

CWE-398 : Indicator of Poor Code Quality
CWE-547: Use of Hard-coded, Security-relevant Constants
CWE-252: Unchecked Return Value
CWE-571: Expression is Always True
CWE-581: Object Model Violation: Just One of Equals and Hashcode Defined
CWE-584: Return Inside Finally Block
CWE-563: Assignment to Variable without Use ('Unused Variable')
CWE-478: Missing Default Case in Switch Statement
CWE-495: Private Array-Typed Field Returned From A Public Method

Most of the weaknesses were labeled as either "low" or "medium" in severity. However, there were also some labeled as "high". CTSC recommends that developers examine and, if necessary, improve code that a static analyzer designates as being highly vulnerable.

For web-based Science Gateways, a web application security scanning tool should be used. A list of both commercial and open source tools can be found at OWASP (https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools).

[2] The SWAMP (https://continuousassurance.org/) offers some of these tools. (Disclaimer: some of the authors of this report are also affiliated with the SWAMP).

Development Support for Applications

Evernote and other services support developers of applications by providing a development environment that allows the application to be developed and tested without putting production services and data at risk due to a possibly immature application. SciGaP should implement a similar development service for application and gateway developers alike.

Application Identification

A situation may arise where a fielded application may be determined to be insecure. By having applications identity themselves to SciGaP services, those services can be configured to recognize such insecure applications and advise users to upgrade and even deny service to applications if the situation warrants.

Trust Model

Defining a trust model, that is a definition of the privileges and responsibilities of all the components of the SciGaP ecosystem, will help both in its design and in the education of the user community making implementation.

Glossary

- **IDL** - Interface Definition Language. A high-level specification language for defining the interface between software components that need to communicate. From the IDL, client and server implementation stubs can be generated for multiple programming languages.
- **Thrift** - a software framework (from Apache) that generates client (SDKs) and server/services code (for multiple languages) from an IDL.
- **Evernote** - a commercial note-taking, archiving, sharing cloud service that uses Thrift for its services and SDKs.
- **OAuth** - an open standard for authorization. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials.

Related Links

https://www.owasp.org - Open Web Application Security Project
https://cwe.mitre.org/ - Common Weakness Enumeration (for software)
https://scan.coverity.com/ - static analysis for C/C++ and Java (free for open source projects)
https://continuousassurance.org/ - Software Assurance Marketplace (SWAMP)
http://brakemanscanner.org/ - static analysis for Ruby on Rails applications
https://issues.apache.org/jira/browse/THRIFT/?selectedTab=com.atlassian.jira.jira-projects-plugin:issues-panel - issues for Thrift developers
http://docs.travis-ci.com/user/languages/java/ - Travis for continuous integration of Java code
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6702697
http://cxsecurity.com/issue/WLB-2014010087
https://www.owasp.org/index.php/Category:OWASP_Java_Project

http://shiro.apache.org/ - Apache Shiro Java Security Framework
http://architects.dzone.com/articles/how-secure-and-apache-thrift - blog on writing a more secure Thrift server & client (in Java)

References

1. Jim Basney, Rion Dooley, Jeff Gaynor, Thejaka Amila Kanewala, Suresh Marru, Marlon Pierce, and Joe Stubbs, "*Integrating Science Gateways with XSEDE Security: A Survey of Credential Management Approaches*," XSEDE Conference, July 2014, Atlanta, GA http://hdl.handle.net/2142/49302
2. Thejaka Amila Kanewala, Suresh Marru, Jim Basney, and Marlon Pierce, "*A Credential Store for Multi-Tenant Science Gateways*, "International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 2014, Chicago, IL http://hdl.handle.net/2022/17379
3. Jim Basney and Von Welch, "*Science Gateway Security Recommendations*," Science Gateway Institute Workshop, September 2013, Indianapolis, IN http://www.ncsa.illinois.edu/People/jbasney/201309-gwsec.pdf
4. Nicolas Viennot, Edward Garcia, and Jason Nieh**,** "*A Measurement Study of Google Play*," Proceedings SIGMETRICS '14, June 2014, Austin, TX http://www.cs.columbia.edu/~nieh/pubs/sigmetrics2014_playdrone.pdf