SECURING ACCESS TO CONTROLS APPLICATIONS WITH APACHE HTTPD PROXY

Piotr Golonka, CERN, Geneva, Switzerland,
Hannu Kamarainen, Jyväskylä University of Applied Sciences, Institute of Information Technology, Jyväskylä, Finland

Abstract

Many commercial systems used for controls nowadays contain embedded web servers. Secure access to these, often essential, facilities is of utmost importance, yet it remains complicated to manage for different reasons (e.g. obtaining and applying patches from vendors, ad-hoc oversimplified implementations of web-servers are prone to remote exploit). In this paper we describe a security-mediating proxy system, which is based on the well-known Apache httpd software. We describe how the use of the proxy made it possible to simplify the infrastructure necessary to start WinCC OA-based supervision applications on operator consoles, providing, at the same time, an improved level of security and traceability. Proper integration with the CERN central user account repository allows the operators to use their personal credentials to access applications, and also allows one to use standard user management tools. In addition, easy-to-memorize URL addresses for access to the applications are provided, and the use of a secure https transport protocol is possible for services that do not support it on their own.

INTRODUCTION

Controls applications for numerous systems at CERN's Large Hadron Collider, as well as for technical infrastructure are built with the WinCC Open Architecture SCADA software [1]. Being highly modular and inherently distributed in nature, it allows the separation of the operator consoles (typically in the Control Rooms, or remote access using Terminal Servers) executing the "User Interface" part of the controls application from other critical tasks performed on the control servers. This architecture allows for good scalability in terms of the number of users. A typical problem is however to deliver files, such as panels, libraries or binary plugins, needed at run time, to the operator consoles. These files are application-specific and therefore need to be stored in the control servers where the controls application is running. With over 500 consoles in the control rooms, 20 Terminal Servers and ~200 controla applications that should all be accessible from any console enforcing access control to protect from unauthorized use becomes complex to handle.

In the classical approach applied until now, a file-share is created on every control server, for every application, to make the necessary server-stored files accessible on every console. The fact that files need to be shared to clients running Linux (for consoles) and Windows (for Terminal Servers) operating systems makes it necessary to maintain two parallel file-sharing technologies (NFS and SAMBA), see Figure 1A. Even with an administrative toolbox, support from the operating system administrator, and automated tools to generate application-specific configuration files for the UIs, the management and maintenance of an ever-growing number of applications becomes a time-consuming task. The hardship of resolving numerous performance and stability problems have also prompted us to search for alternative ways of starting the application UIs on operator consoles.

Recent versions of WinCC OA allow for an alternative way of distributing the files to the User Interfaces using the *http* protocol. The files are served by the embedded http server component of WinCC OA, and may be requested as necessary by the clients, who also maintain a cache of the files on a local disk to improve performance. Optionally, the connection to process-data could also be tunnelled by this server, yet we do not use it. In this setup, any machine (with WinCC OA installed) could request the necessary files and hence run the controls application, by simply knowing the appropriate *URL* (address) at which the control server exposes the necessary files. Even though some basic security mechanisms are built into the WinCC OA web server, they are suitable for isolated plants and would not scale for large scale deployments. Moreover, configuring the TLS certificates to obtain a secure https connection is hard to maintain, as well as the fact that there is typically more than a single controls application running on a server. In addition, in the basic setup, we would not be able to assure proper functioning of redundant systems.

In the remaining part of the paper we will present an architecture with a http proxy server in between the server and the UI part of the application, and we will describe the benefits of such arrangement.

ARCHITECTURE WITH A REVERSE PROXY SERVER

A proxy server is a well-known pattern in computer networks, particularly for the World Wide Web: a dedicated server acts as an intermediary between clients and a server that hosts some services. Of our interest are the reverse proxies, that protect the server from uncontrolled access to its services coming from a large network, and perform tasks such as authentication/authorization, load-balancing or certificate handling (SSL/TLS offloading).

The popular Apache *Httpd* web server [2] readily available on Linux platforms, may be configured to act as

ISBN 978-3-95450-148-9

a reverse proxy for http traffic. This mature technology used in over 50% of World Wide Web installations [3] provides a scalable, secure and robust platform adequate for production systems. It can perform user-authentication tasks through the LDAP technology (which allows users to use their standard CERN credentials), and it may be deployed in clustered setups to provide a higher availability.

We installed a pair of proxy servers (in two separate physical machines) to improve the availability of the setup. We configured the servers as well as the DNS (network Domain Name Service [4]) such that the pair of proxies are visible through the same DNS name, and hence they could accept requests in a round-robin way. Having more than one IP address mapped to the same DNS name is a standard feature of the DNS [5], employed in proxy configurations, and in many scenarios where the maximum time to serve a request is not critical it provides sufficient amount of availability, without resorting to very complex setups.

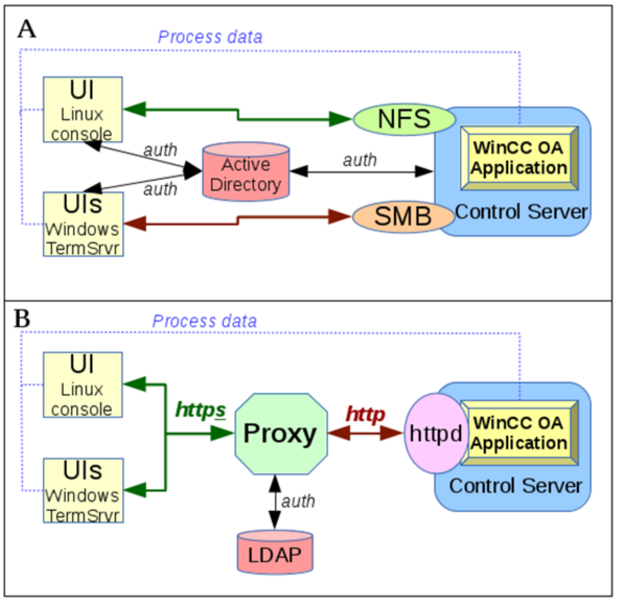

Figure 1: System architecture: (A) with file sharing; (B) with proxy.

NETWORK NAME SERVICE CONFIGURATION

To be able to serve access for many applications the proxy needs to listen for requests at many DNS names; then, by analysing the requested URL, it can determine the proper destination for the request to be forwarded (to one of the embedded WinCC OA web servers running in a controls application on one of the control servers).

Instead of manually registering a new name for every new application, a more advanced setup is used: a wildcard DNS record [4,6], "`*.scada.cern.ch`" pointing to the pair of proxy servers. In effect, a virtual on-demand "scada.cern.ch" subdomain appears in the DNS, allowing us to handle any number of host names.

We are then free to choose the "host name" for every application. The resulting URL that needs to be used to access the application would then have an easy-to-remember form such as `https://MyApplication.SCADA.CERN.CH`.

CERTIFICATES AND SSL/TLS OFFLOADING

A secured variant of the transfer protocol (*https*) should be used by the client and server to assure secure transfer of files and authentication information. This requires TLS certificates to be installed and maintained on the server side. With an already large and growing number of applications, maintenance tasks would become tedious: a signed-certificate would have to be generated and replaced whenever it expires for every single controls application (more than 200).

To sort out this issue we applied the "SSL/TLS offloading" mechanism, with a certificate installed only on the proxy servers. The communication between the client and the proxy (containing sensitive information) is transmitted over the encrypted *https* protocol, whereas the communication between the proxy and the control servers (which can be considered as happening over a trusted channel) is performed over the standard *http* protocol. This way the proxy server *offloads* the control servers not only from the management of their own certificates, but also from the actual task of encrypting/decrypting the communication.

At the technical level, this solution requires the use of a so-called *wildcard certificate* installed on the proxy. This special type of certificate is more complicated to arrange for (it could not be generated by the CERN Certificate Authority), yet it was purchased for us by the CERN Web Service support.

ACCESS CONTROL AND SECURITY

Access to the controls applications needs to be secured and limited to the people who are explicitly granted the necessary permissions. Even though an additional level of access control is provided by the applications themselves, we allow the start of the UI for a controls application only to authorized users.

In the previous approach, the authorization was implemented through file-system privileges of the NFS and SMB network shares, and integrated with operating system authentication and authorization mechanisms (Kerberos, Microsoft ActiveDirectory ™ , CERN egroups), leading to numerous problems with availability, debugging and configuration.

To provide a similar level of configurability and flexibility, we configured the proxy servers to enforce the authorization of use of an application by responding or denying the requests, according to username/password authentication. The standard *mod_ldap* module of Apache httpd has been deployed to integrate with CERN's identity management services over the standard LDAP protocol. User credentials are checked against Active Directory servers, and the authorization decision is based on account membership of the so-called e-groups[7].

For each application, a separate hostname/URL is used for access, and a dedicated entry in the proxy configuration file is placed to define the permissions per application.

To ensure secure access, we configured the WinCC OA embedded http servers to only accept requests from the two proxy servers. We also enforced the use of the secure ***https*** protocol when connecting to the proxies.

In our solution we chose the *data* connection from the UI program to the control servers ***not*** to be mediated through the proxy. The principal reason is to conserve the high performance and robustness of the process-data communication. We consider this communication channel to be secure enough in the isolated CERN Technical Network, and it could also be strengthened by the use of WinCC OA's built-in Kerberos encryption and integrity mechanisms, if needed.

OPERATION WORKFLOW

The sequence of actions that happen when a user starts a new UI for a controls application of their choice, either on a console or in a Terminal Server (see also Figure 1B), is the following:

1. The user starts the WinCC OA User Interface program, pointing it at the application URL. This causes a ***https*** request for the application files to be sent to the proxy.
2. The proxy responds with an authentication request, which results in a window opening on the user's UI asking for a CERN user-name and password.
3. The credentials are sent to the proxy, which forwards them to the CERN LDAP server for verification.
4. The proxy analyses the URL to identify the actual application to which the user is requesting access. Using CERN LDAP it checks the user membership in the e-group that was configured for the application.
5. If the answer is positive, the proxy forwards the request over *http*, to the control server that sends the requested file(s).
6. The file(s) received by the proxy are forwarded to the UI client, which saves them in a cache on a local disk, and then proceeds with standard operation.

In the case where one of the proxy servers is not available, and there is a pending request, the UI automatically retries the request with the other proxy, after some short delay.

The special case of redundant WinCC OA systems (e.g. in [8]) is also treated with additional configuration on the proxy servers. Assume that the UI was connected to the first server (through a proxy) and hence received the files from it up to certain point. If there is a switch to the second server, the data connection is routed to the second server by the WinCC OA redundancy mechanism. However, the proxy will not be able to forward requests to the first server any longer; in such situation, it will find out that a timeout has occured, and then it will fall-back to requesting the same files from the second control server in the redundant pair, as expected. This allows for smooth interventions in redundant systems, where the complete server needs to be shut down. Such a scenario was not possible with the file-sharing architecture (when network share became unavailable, the complete UI session is blocked by a non-accessible file, even though data could flow from the second redundant system).

In case of failed authentication or authorization, or mistyped URL, the proxy server informs the client UI which displays the appropriate information. A fall-back web page will be served if the URL is opened in a web browser, displaying information about the application.

CONFIGURATION AND MANAGEMENT

The system is integrated with the central Configuration Database System Information and SCADA Application Service web portal for management and configuration [9]. They allow to enter/change all the necessary information about managed applications.

The configuration files needed by the proxy server software are generated by a set of Python scripts, which connect to the database and pull the information about all declared applications, namely the requested URL for the application, the information about the target application location (control server name, TCP port number of the embedded http server, or access control). The script, run every few minutes, re-generates the configuration files and signals the proxy to reload the configuration, if any change in the configuration is detected.

Certificate management and the start-up of the http proxy processes, as well as the overall security of the system is a standard operating system administration job.

On the control server side, the necessary configuration is automated. The task of starting and configuring (e.g. TCP port number extracted from the DB) the embedded WinCC OA http server is performed by a dedicated component centrally deployed and managed for all the applications.

On the clients, one needs to make sure that OpenSSL libraries are installed in addition to the standard WinCC OA client installation, to enable the secured *https* transport.

PERFORMANCE

Tests were conducted to compare the performance of serving files using http with that offile-shares (NFS, SAMBA). We measured the time necessary to start up one of our largest applications (the supervision of the CERN electrical network [8]) for which a large number of files need to be loaded by the UI. On Windows consoles where files are shared with SAMBA, it takes 34 seconds,

ISBN 978-3-95450-148-9

when the files are copied to a local disk, this is reduced to 21 seconds . The start-up time when using the http file server with the proxy takes 23 seconds without files in the cache, and 16 seconds when files are cached (Figure 2). This is a perceptible performance improvement for the operators. Simple scalability tests did not indicate any problem with a growing number of clients or applications.

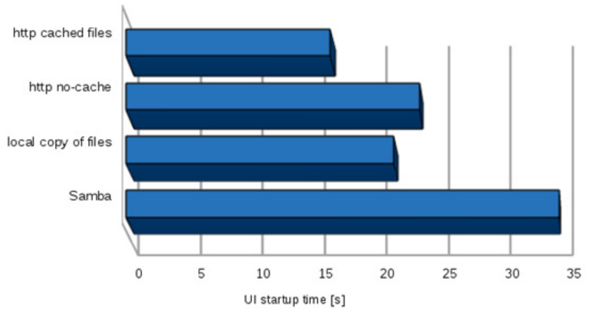

Figure 2: Time to start up the UI for a large SCADA application on a Windows Terminal Server, with Samba and http method.

CONCLUSIONS AND OUTLOOK

The architecture presented in the paper is currently being deployed for pilot use as an alternative way to start UIs for controls applications. We intend to watch the behaviour closely and identify possible limitations (until now the only one found is related to accessing a local cache based on sqlite database, that contains metadata for one of our applications.

The http file transfer provides a good alternative to standard solutions based on file shares, and offers very good performance. It also allows us to address the important case of WinCC OA redundant systems.

Standard, proven technologies and architectural patterns used typically for the World Wide Web prove to be effective also in non-standard applications for control systems. The solution proposed here is generic enough to possibly be implemented using other http proxy software packages (such as Nginx, which is gaining popularity and already takes 25% of web traffic [3]).

By delegating certain aspects of security to a dedicated service that can be maintained independently we will gain on flexibility and increased availability. Thanks to the redundant nature of the proposed architecture, numerous security-related interventions (patches) can now be applied without affecting the running applications.

If needed, the proxy can be used to secure fragile systems with embedded web servers, such as high voltage system mainframes, used in a large number in Detector Control Systems at CERN. It is of particular interest to assure their security for the long-term, as patching firmware to gain security may possibly induce undesired side-effects affecting their stable operation.

The proxy may also work with web sockets (using the `mod_proxy_wstunnel` module, although it requires a newer version of Apache httpd 2.4). This paves the ground for secure access to the future "Web UI" feature, presented also during this conference [10].

REFERENCES

[1] SIMATIC WinCC Open Architecture (previously PVSS) SCADA software from ETM (Siemens subsidiary), http://www.etm.at

[2] Apache HTTP Server Project, http://httpd.apache.org

[3] "Usage of web servers for websites" W3Techs Web Technology Surveys, retrieved on 15 September 2015 http://w3techs.com/technologies/overview/web_server/all

[4] P. Mockapetris,, "Domain Names - Concepts And Facilities", RFC-1034, http://www.rfc-base.org/txt/rfc-1034.txt

[5] T Brisco, "DNS Support for Load Balancing", RFC-1794, http://www.rfc-base.org/txt/rfc-1794.txt

[6] F. Lewis "The Role of Wildcards in the Domain Name System", RFC-4952, http://www.rfc-base.org/txt/rfc-4592.txt

[7] CERN e-groups, http://cern.ch/egroups

[8] J-C Tournier *et al*,"New Electrical Network Supervision for CERN: Simpler, Safer, Faster, and Including New Modern Features", ICALEPCS 2013, October 2013

[9] F.Varela *et al*, "High-level Functions for Modern Control Systems: A Practical Example", ICALEPCS 2013, October 2013

[10] A. Voitier *et al*, "Integrating Web-based User Interface Within CERN's Industrial Control System Infrastructure", ICALEPCS 2015 contribution WEPGF070