jamk.fi

Securing Access to WinCC OA Control Applications with Apache httpd Proxy

Hannu Kämäräinen

Bachelor's thesis
October 2015
Technology, communication and transport
Degree Programme in Software Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s) Kämäräinen, Hannu	Type of publication Bachelor's thesis	Date dd.mm.2015
	Number of pages 39	Language of publication: English
		Permission for web publication: x

Title of publication
Securing access to Wincc OA control applications with Apache httpd proxy

Degree programme
Software Engineering

Supervisor(s)
Ari Rantala

Assigned by
European Organization for Nuclear Research (CERN)

Description

The traditional SMB/NFS based start-up of WinCC OA control applications at CERN has typically had problems with availability, file permissions, performance, and access for redundant WinCC OA projects.

WinCC OA has an embedded web server, which allows for an alternative method for transferring project files during start-up using the HTTP protocol, however security-wise it is considered lacking. The objective was to implement a web proxy service to secure the http based start-up for WinCC OA applications. Apache httpd was the chosen technology for the proxy due to its maturity, customizability, and wide community support.

All authentication and authorization in accessing the projects is delegated to httpd and its modules, thus providing strong security. Common techniques such as round-robin and wildcard DNS-records were used to provide redundancy for the proxy setup, and to handle dynamic sub-domains and controlling access to different projects. The generation of the necessary httpd configuration files was automated, and software components were produced to prepare WinCC OA projects for the embedded web server use and connections coming through the proxy.

Future use of the proxy may include similar use cases in securing other CERN web applications, and possibly other hardware with embedded web servers, such as high voltage system mainframes.

Keywords (subjects)

Apache, httpd, proxy, WinCC OA

Miscellanous

Contents

FIGURES

TABLES

Terms and Acronyms

Apache httpd

An open source web server by the Apache foundation.

Authz

Short for “authentication and authorization”.

CTRL

Pronounced ”control”, a proprietary scripting language that has a lot of characteristics from the C-language. Main scripting language used in WinCC OA.

HMI

”Human Machine Interface”, typically a ”user friendly” device or piece of software to control machinery, devices or other equipment. The purpose of WinCC OA applications is often to be used as an HMI.

HTTP Proxy

A web server which is purposed to act as a middle-man between the client and the actual destination. Proxies are often used for example for load-balancing high-traffic servers, high-availability (redundancy), and added security (i.e. authentication and authorization), or to hide the location of the client (forward proxy).

JCOP framework

“Joint Controls Project” framework, a collection of WinCC OA components that are produced and maintained by the SCADA section in co-operation with the experiments. A typical JCOP component can include for example datapoint definitions and configurations, device definitions, CTRL libraries, and user interface panels.

Kerberos

A network authentication protocol developed in Massachusetts Institute of Technology.

LDAP

Short for Lightweight Directory Access Protocol, an industry standard protocol for accessing and maintaining information directory services in a network, such as users and user groups in an organization.

Shibboleth

A ”single sign-on” system for authentication and authorization in a network, effective at keeping users logged in between different web services.

SLC6

Scientific Linux CERN 6. A distribution of GNU/Linux that is widely used at CERN.

SMB

SMB (“samba”) is an implementation of the SMB/CIFS network protocol, mainly used for sharing files, printers, ports and other generic communications in a network.

1 Introduction

1.1 The client

The European Organization for Nuclear Research (CERN), located in Geneva, Switzerland, is the largest particle physics research laboratory in the world. Currently CERN employs around 2500 full-time staff members, and around 12 000 fellows, students and other associates, and has 22 member states, and collaborators from more than 600 universities and research facilities. The main purpose of CERN is to provide the infrastructure for high-energy physics experiments, including particle accelerators, computer grids and logistics among other things. (Wikipedia 2015.)

The thesis was written at CERN as a part of a Technical Student contract between 1.5.2014 – 30.6.2015, in the Engineering department, Industrial Controls Engineering group, SCADA section (Supervisory Control And Data Acquisition). The SCADA section focuses on designing, implementing, and maintaining software components for physicists and engineers at CERN to help them develop control systems for the physics experiments and general infrastructure. Currently these components are mostly based around a proprietary SCADA tool called WinCC OA from Siemens/ETM. The programming languages most often used when developing for WinCC OA are the proprietary scripting language CTRL and C++.

1.2 The assignment

Typically, the WinCC OA HMI applications have used a SMB/NFS based startup, however this has historically had problems with availability, file permissions, performance, and access for WinCC OA redundant projects.

As an alternative way for starting these applications, WinCC OA has an embedded web

server which allows transferring project files over the HTTP protocol, however security-wise it is considered lacking. Therefore the SCADA section started looking for a way to secure the WinCC OA web server by placing a proven third party web server to act as a proxy in front of it. The proxy would then take care of authentication and authorization on behalf of the WinCC OA server. In addition to strong security, the objectives for the proxy included support for redundancy for the proxy itself to avoid a single point of failure, and a way to handle redundant WinCC OA projects.

Initial research had shown that the Apache web server would be a suitable candidate due to its maturity, customization options, and wide community support. A secondary objective was to research the suitability of this proxy setup in securing also other potential WinCC OA web solutions, such as the WebUI and Ultralight Client.

2 Technologies: WinCC OA & Apache httpd

2.1 WinCC OA

WinCC Open Architecture is a Supervisory Control And Data Acquisition system (”SCADA”) by Siemens/ETM, officially available for Windows, Linux and Solaris. WinCC OA is widely used at CERN to operate infrastructure, devices and machinery in the physics experiments.

WinCC OA offers the tools to implement user interface panels, launch and use these panels, configure and manage datapoints which act as a kind of database and as an interface between hardware and software, and connecting physical hardware to said datapoints, among other things. WinCC OA works on special “managers”, like the User Interface manager, which runs different types of user interfaces, such as graphical editors and HMI application panels, and the CTRL manager, which can be used for running standalone CTRL scripts.

The user interface manager in WinCC OA is called WCCOAui. It consists of three modules: VISION, PARA, and GEDI. Of these, VISION is used for “using” the panels, PARA is used for configuring and managing datapoints, and GEDI is the built-in graphical editor for implementing user interface panels (Figure 1). Appendices 1-3 contain some screen shots of example WinCC OA applications currently in use at CERN.

The proprietary scripting language CTRL is the main language used for implementing the logic for WinCC OA panels, however, being based on the Qt graphical framework, Qt/C++ extensions are also possible for implementing callable library functions and for creating custom panel widgets.

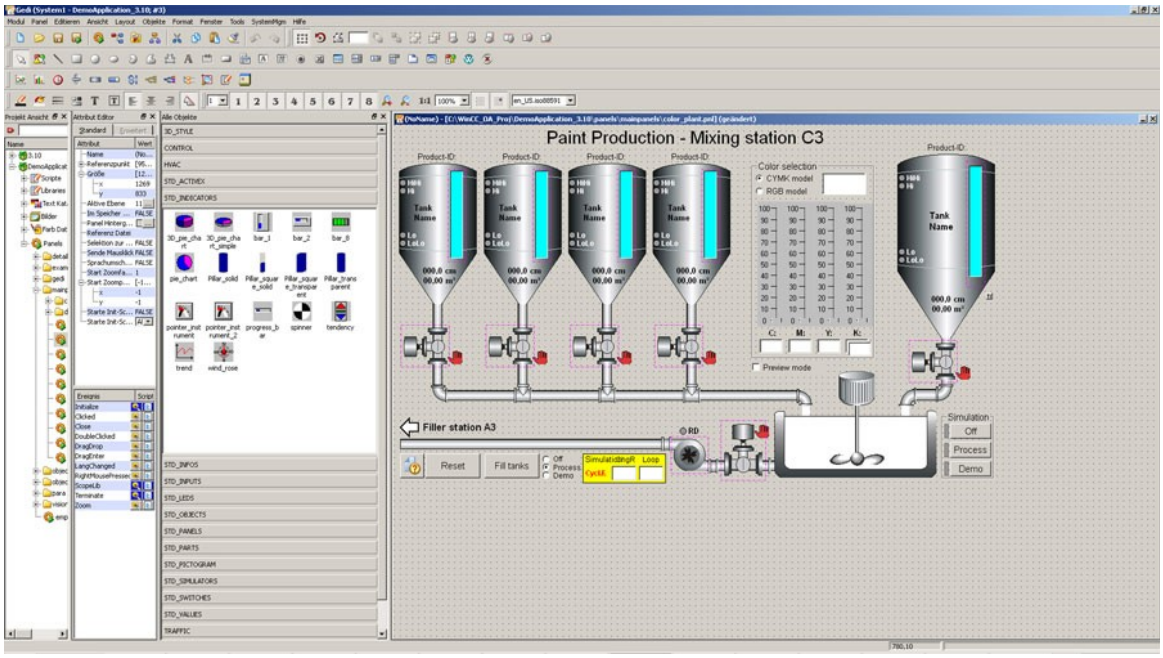


Figure 1. GEDI, the graphical editor used to create user interface panels in WinCC OA. (Siemens SIMATIC WinCC OA 2015)

WinCC OA Projects

At the heart of WinCC OA, there are “projects”. Once a project is created, datapoints can be configured, user interface panels may be implemented, and hardware and devices may be connected etc. Several projects may run on one server, as long as they are configured properly so that no ports are conflicting for example.

A project may be set to run in distributed mode, so that the server is running on one machine (data and event managers), and the user interface clients (UI managers) can be run on remote machines. This is the most common way of setting up a WinCC OA project. Sometimes projects in production are also set to run in redundant mode, meaning there is an identical copy of the project running on a separate server, constantly polling the “master” project, and taking over as needed if the master is detected to be unresponsive. Once the master project has resumed online status, the backup will automatically return to stand-by mode. This redundancy feature of WinCC OA translates into an objective for the httpd proxy setup: Make the proxy “redundancy aware”, in order to properly direct connections to redundant WinCC OA servers as their availability changes.

Access control for these projects is provided by a CERN-developed extension for WinCC OA, which is connected to the CERN centralized user database.

At CERN, all WinCC OA production projects reside inside an internal network and are not accessible from the internet.

WinCC OA embedded web server

Typically, a single UI workstation is used to access multiple projects running on multiple servers, where files to launch the UI panels are accessed through a SMB/NFS based shared network directory.

In practice this happens by executing the command WCCOAui, which starts the UI manager, and for which some basic parameters are provided, e.g. name of the project, panel name to open, possibly debug flags and other options.

A WinCC OA project can be extended to run an HTTP server, implemented completely in CTRL, and which will be running in the project in its own CTRL manager. This web server can be used similarly to a traditional web application back-end, and also to act as a file server to serve the files necessary (panel files which describe the graphical interface, CTRL libraries, image files etc.) to start UI applications using the WCCOAui client.

With the option “-server” to WCCOAui, the client can be instructed to connect to a remote WinCC OA HTTP server. The client has built-in authentication support, however on the server-side (in the WinCC OA HTTP server), the only available mechanism for authentication and authorization is the WinCC OA built-in “project users” database, which is a simple “check and match” system. The client supports HTTPS as well, however in order to make use of it in a normal case, each WinCC OA server running an HTTP server would need to have its own TLS certificate. With around 200 applications to cover, this kind of management effort is to be avoided. An additional benefit of the client is that it will cache all fetched files from the HTTP server, similar to a web browser, which will fasten the startup of applications in consecutive startups. New files will only be downloaded in case the server has a newer version available.

The above is what is available out of the box in WinCC OA. There are good features, but some aspects of them are clearly lacking as well. By itself, the WinCC OA HTTP server already offers an alternative way of starting the applications, even if security is not strong,

and there is some work involved with deploying the TLS certificates.

This web server functionality of WinCC OA is at the root of the proposed HTTP proxy solution.

2.2 Apache httpd

The Apache HTTP server (httpd or “HTTP daemon”) is the most popular web server in the world, in June of 2013 covering over 50% of all web sites (June 2013 Web Server Survey 2013). Launched in 1995, the strengths of httpd include maturity, stability, huge user community, it is well documented and widely supported, and still in active development, lead by the Apache Software Foundation. httpd is open source, and highly customizable with its numerous modules, allowing to extend its functionality for example by choosing from a wide range of different authentication and authorization methods such as LDAP, Kerberos or Shibboleth. Modules are available for augmented security and monitoring capabilities, enabling scripting support for a variety of languages, load balancing, proxying, and other features.

Httpd is readily available in all mainstream Linux distributions, importantly also in SLC6, being a derivative of CentOS.

Http proxy concept

In general a proxy is regarded as something that does a specific task on behalf of someone else, often also hiding or abstracting one party from the other. In the World Wide Web, proxies are used for a variety of purposes, including load-balancing, acting as gateways between networks, HTTPS-to-HTTP offloading, and added security (Wikipedia 2015). An important use-case in today's world is also the ability to hide the identity of the user using a proxy. Proxies are very commonly used for web servers, and naturally Apache httpd is

also capable of fulfilling this job through its available modules.

There are three basic types of HTTP proxies.

- A proxy which does not alter the communications between the client and the end destination in any way is called a gateway proxy. (Apache HTTP Server 2015.)

- A forward proxy is a proxy where the client has control over where the proxy is connecting. This can also be a gateway proxy of the previous type. (Apache HTTP Server 2015.)

- Reverse proxies are proxies where the proxy decides where to direct the connecting client based on some information, and due to its features this is the type of proxy that is the most interesting in this project. General use-cases for reverse proxies include redirecting users to different destinations based on location or other available information, controlling access to protected content, load-balancing, and HTTPS-to-HTTP offloading. No special configurations are needed on the client to use a reverse proxy, and the client does not necessarily know that it is connecting to a proxy of this type at all, as the destination server (decided by the proxy) can be mapped to the address space of the proxy, making it look like the destination content is present in the URL of the proxy. (Apache HTTP Server 2015.)

3 Requirements Specification

Although the assignment started more as a research mission, some primary requirements could immediately be recognized for a solution to even be considered.

- The technology chosen for the proxy should be mature, stable and well supported.

- It should provide strong access control capabilities in the form of authentication and authorization, and be able to interface with the CERN authentication infrastructure.

- Encrypted communications must be supported.

- The proxy has to be made highly available to avoid a single point of failure.

In addition to these requirements, several goals were identified during the research/implementation process as nice to have features, which would considerably ease both use and maintenance of the proposed proxy solution. These would be:

- Automating as much of the proxy configuration process as possible, for example in the case of adding or removing new proxy machines, or adding and removing projects where the proxy can connect to.

- Mitigating the effort of managing TLS certificates per every WinCC OA project (current project count is around 200).

- Better support for redundant WinCC OA projects.

- User friendly access points for projects (URLs).

4 Implementation

4.1 Premise

As the author's initial experience with Apache httpd was limited to setting up web sites, the implementation process started in small steps, setting up a few SLC6 virtual machines and getting basic proxy functionality working, and then gradually extending to learn and use the different features of httpd to approach the goals set for the project.

In httpd, the most basic proxy setup takes very little work, only a few lines in the main configuration file. Even the final solution for the httpd VirtualHost configurations is relatively simple yet effective, and the majority of the effort to get there was a process of trial and error to find the best working options. Attempting to use RewriteRules to do some of the work may seem like a good idea at several points for example, but once the developer becomes familiar with the tools at hand, the developer will know better.

The capabilities of the WCCOAui client and the WinCC OA server were almost completely known beforehand, only some surprise issues surfaced that required action. Therefore the majority of the work involved the proxy layer in finding a way to fulfill the requirements described above.

4.2 Architecture overview

An overview of what the achieved final solution looks like is shown in Figure 2. The proxy (or proxies) between the client and the WinCC OA servers act as a reverse proxy, taking connections from clients, authenticating them, deciding from the domain name that was used to access it where the client should be forwarded to, and also uses this information to authorize the client against CERN's user database.

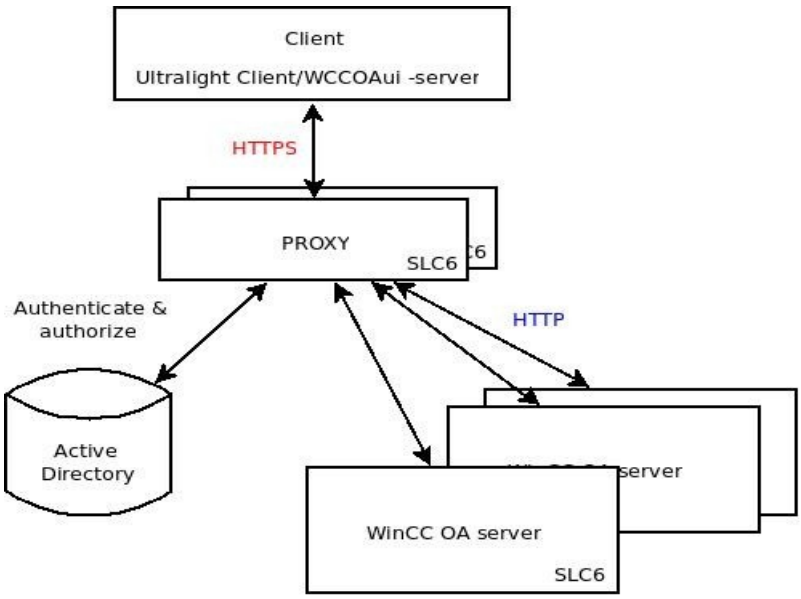


Figure 2. A high-level view of the client-proxy-WinCC OA server architecture.

On successful authorization the proxy starts fetching files from the WinCC OA server and passes them to the client. On the WinCC OA servers, access to the embedded HTTP server, which allows for the alternative start-up method, is restricted by IP-address only to the specified proxies.

HTTPS-to-HTTP offloading

An important task of the proxy is also HTTPS-to-HTTP offloading, which means that between the client and the proxy, all communications is over HTTPS (encrypted), while between the proxy and all the WinCC OA servers, communications happen over plain HTTP. This is acceptable because the internal network is considered to be a secure environment.

The offloading mechanism removes the need to install and maintain TLS certificates on every WinCC OA server, and also offers a minor performance improvement, as all the encryption and decryption takes place on the proxies. This is a common pattern used elsewhere in the industry, where there can be major performance gains by delegating heavy encryption and decryption duty to specialized servers, in an otherwise non-encrypted network.

Event and data manager connections

One noticeable issue is that not all traffic should pass through the proxy, but only the start-up phase. Once the user has been authorized to use the application, project files are downloaded, and the UI is launched. From there, the UI is free to connect to the event and data managers on its own. Event and data managers are where essential system information flows between the client and the WinCC OA project, after the application has been successfully started.

Therefore it follows that in order for the UI panels to work, there needs to be a network connectivity between the UI, and the WinCC OA server: the UI and the server cannot be completely isolated from each other. This requires the use of one additional parameter for WCCOAui, the -noTunnel flag, which tells the UI to connect to the event and data managers directly and not through the proxy, to not hinder the traffic between the client and the managers. If this traffic were to be tunneled through the proxy, it would need to be converted to HTTP protocol, which has a cost, and this is simply not desired.

4.3 Authentication & Authorization

When accessing projects over the WinCC OA HTTP server, the HTTP server itself will do no authentication or authorization effort other than restricting access by IP-address only to the proxy servers. Instead, all authz is delegated to the proxy, which will read

application specific authorization settings from its VirtualHost configuration files, and connect to a centralized database of users, through which access is globally managed to all CERN services and applications over all networks. The underlying technology is based on LDAP, a directory service used for managing users in a hierarchial way, for example assigning users to groups (e-groups) and assigning them roles. A user can be assigned for example to an e-group which holds all the members of an experiment, or all the authorized users of some equipment.

Modules exist for httpd which allow querying LDAP directory servers and using Kerberos and Shibboleth (Apache HTTP Server 2015). These modules enable users to use the same CERN accounts they already have set up to access their projects through the httpd proxy as well.

Three authz methods were researched for the proxy use case, all of which are already in use at CERN: Basic Authentication, Kerberos, and Shibboleth. The next chapters will cover these methods in further detail.

4.3.1 Basic Authentication

Basic Authentication is the most simple authentication method for HTTP web services. Communication between the server and the client takes place through the use of HTTP headers. A web service may indicate that it requires authentication by sending the client a “401 Not Authorized” status and a “WWW-Authenticate” field, to which the client must respond with a HTTP “Authorization” field that is constructed in a certain way, and containing an encoded (not encrypted) username and password (RFC 1945, 1996). The authentication method is then combined with some kind of database to reflect against, which in this case is LDAP based.

The httpd modules required to use Basic Authentication and LDAP as the database are

both included in the default httpd installation available for SLC6. The modules in question are mod_auth_basic (enables Basic Authentication on the server), mod_authnz_ldap (enables use of an LDAP directory as the authentication database), and mod_ssl (enables the use of SSL). (Apache HTTP Server 2015.)

Below is a simple example of using Basic Authentication and LDAP to secure a proxy access point on a server. The configuration could be placed at the bottom of the main httpd configuration file or in its own file which is then included into the main file.

```
# Port number where this configuration will have effect
<VirtualHost *:443>

        # Hostname of the server
        ServerName myhost

        <Proxy *>
                # Everybody is allowed to access on this access point (up to
                # authentication)
                Order deny,allow
                Allow from all

                # Name of the auth domain, purely descriptive. Shown to client
                AuthName "Some Authentication"

                # Select authentication method to use
                AuthType Basic

                # Select authentication provider
                AuthBasicProvider ldap

                # Only use LDAP for authentication. If this fails, don't try others
                AuthzLDAPAuthoritative On

                # LDAP search parameters for the user query
                AuthLDAPURL
                "ldap://cern.ldap.url/OU=Users,OU=OrganicUnits,DC=cern,DC=ch?
                sAMAccountName?sub? (objectClass=person)"

                # Username of an account which can read the LDAP database
                AuthLDAPBindDN "username"
```

```
            # Password of an account which can read the LDAP database
            AuthLDAPBindPassword "password"

            # Specify the authorized ldap groups to access this location.
            Require ldap-group CN=some-allowed-group,ou=e-
            groups,ou=Workgroups,dc=cern,dc=ch
        </Proxy>

        # Maps a remote location to server's local address space
        ProxyPass /accesspoint http://somewhereelse:8080/

        # Keeps remote location's redirects in local context
        ProxyPassReverse /accesspoint http://somewhereelse:8080/
</VirtualHost>
```

According to the above configuration, if navigating for example with a web browser to https://myhost/accesspoint, the client will be asked for authentication, and only successfully authenticated users, who also belong to the specified ldap-group called “some-allowed-group”, will be forwarded to the intended destination which is http://somwehereelse:8080/.

The basic thing about Basic Authentication is that it is simple to use and widely supported, notably also by the WCCOAui client.

4.3.2 Kerberos

Kerberos is a sophisticated authentication protocol building on symmetric key cryptography and requires a trusted third party. Some security features include mutual authentication, where both the client and the server can verify each others' identity, and it provides protection against eavesdropping and replay attacks. (RFC 4120, 2005.)

Support in different applications for Kerberos is more scarce, however it works with web browsers and Curl for example. WCCOAui does not currently support Kerberos. Still the

use of Kerberos is viable for other generic use cases, such as web applications. Web browsers generally require some setting up before being able to use Kerberos authentication. This is done by adding the server host name to the browser's list of trusted URIs (Red Hat Customer Portal 2015).

Kerberos is not available out of the box on httpd, but requires an httpd module to be installed and some additional configuration. The first step is to install the httpd module on the target proxy machine. (Linux @ CERN 2015.)

```
# yum install mod_auth_kerb
```

The default keytab created at /etc/krb5.keytab has to be readable by root only. In addition, another keytab needs to be created to be used specifically when connecting with HTTP. This keytab needs to be made readable by the user apache. The following command will do the trick, creating a new file /etc/krb5.keytab.HTTP . (Linux @ CERN 2015.)

```
# cern-get-keytab --service HTTP –isolate
```

Make sure the produced file is readable by the user apache:

```
# ls -l /etc/krb5.keytab.HTTP
```

And if not, to fix this:

```
# chown apache:apache /etc/krb5.keytab.HTTP
```

Now it is possible to start using Kerberos directives in httpd configuration files. Here Kerberos is used to authenticate the user, and LDAP for authorization (checking e-groups). Most of the configuration content is the same as in the previous chapter discussing Basic Authentication, however for completeness everything is included.

```
# Port number where this configuration will have effect
<VirtualHost *:443>

        # Hostname of the server
        ServerName myhost

        <Proxy *>
                # Everybody is allowed to access on this access point (up to
                # authentication)
                Order deny,allow
                Allow from all

                # Select authentication method
                AuthType Kerberos

                # Name of the auth domain, purely descriptive. Shown to client
                AuthName "Kerberos test auth"

                # Enable SPNEGO protocol; “Simple and Protected GSSAPI Negotiation
                # Mechanism” is enabled to let the client and the server figure out
                # themselves which authentication methods are available for both of them,
                # and choose one accordingly. A web browser for example may use Basic
                # Authentication to provide credentials when the used machine is not part
                # of the Kerberos authentication realm.
                KrbMethodNegotiate On

                # Enable password authentication
                KrbMethodK5Passwd On

                # Kerberos realm to be used for authentication
                KrbAuthRealms CERN.CH

                # Strip the realm ('@CERN.CH') from
                # REMOTE_USER(=username@CERN.CH) to get proper user name
                # for the LDAP query
                KrbLocalUserMapping On

                # Normally Kerberos would use the default krb5.keytab, but here we
                # enforce the use of krb5.keytab.HTTP instead
                Krb5Keytab /etc/krb5.keytab.HTTP

                # From here, LDAP is used for the authorization part, and is the
```

```
                # same as before.

                # LDAP search parameters for the user query
                AuthLDAPURL "ldaps://cerndc.cern.ch/OU=Users,OU=Organic
                Units,DC=cern,DC=ch?sAMAccountName?sub?(objectClass=
                person)"

                # Username of an account which can read the LDAP database
                AuthLDAPBindDN "username"

                # Username of an account which can read the LDAP database
                AuthLDAPBindPassword "password"

                # Specify the authorized ldap groups to access this location.
                Require ldap-group CN=my-e-group-name,ou=e-
                groups,ou=Workgroups,dc=cern,dc=ch
        </Proxy>

        # Maps a remote location to server's local address space
        ProxyPass /accesspoint http://somewhereelse:8080/

        # Keeps remote location's redirects in local context
        ProxyPassReverse /accesspoint http://somewhereelse:8080/
</VirtualHost>
```

4.3.3 Shibboleth

Shibboleth is a single sign-on system used with web browsers to keep users logged in between different web services in an organization. Shibboleth is open source and licensed under the Apache Software License. (Shibboleth Consortium 2015.)

Apache can be configured to use Shibboleth by following to the letter the CERN-specific instructions at http://linux.web.cern.ch/linux/scientific6/docs/shibboleth.shtml. However, since Shibboleth is not supported by WCCOAui -server, it's capabilities in this case were not searched very far. It is perfectly usable in normal proxy use, and still has the potential to be used with other WinCC OA web solutions like WebUI and UltraLight Client, or any other ordinary web application running in a web browser.

4.4 Proxy redundancy

Proxy redundancy was achieved using a DNS round-robin setup. Setting it up and maintaining is very easy: On the CERN internal DNS (Domain Name System) server, we specify the alias scada.CERN.CH to point to the IP-addresses of two or more identical proxy machines running SLC6 and httpd. In the httpd VirtualHost settings, the KeepAlive directive must be turned on, so that the client can keep using the existing connection which it found to be responding. In case performance becomes an issue, the directives KeepAliveTimeout (how long to wait for new requests for a connection before closing it) and MaxKeepAliveRequests (limits the total number of requests allowed per connection) can be adjusted. And that is pretty much all there is to it. (Apache Core Features 2015.)

This way, when a client queries the DNS server for the IP-address of the domain scada.CERN.CH, a list containing the addresses (in random order) is returned instead (The Technology Chronicle 2013). WCCOAui -server and modern web browsers at least will try the other addresses if the first one is not responding, however the decision if the other addresses should be tried during the same request, in the end, as always, depends on client implementation.

In the future if considering this proxy setup for other applications, in case the client does not try the other addresses in the list, the TTL (time to live) for the DNS record (scada.CERN.CH) should be set to a very short time on the DNS server (i.e. one minute), so that the list of available proxies is updated as quickly as possible if a machine has become unreachable (RFC 2308, 1998). Still, application and operating system caching can additionally add delay to this. Web browsers for example by default do DNS caching (however depending on the browser, the TTL can often be altered, or disabled) (Flush DNS 2015). Windows machines do it on the OS level (Ben Anderson 2011), while on Linux it depends on the distribution (SLC6 does it).

4.5 Handling redundant WinCC OA projects

Httpd has a very good built-in support for redundant proxy destinations, making configurations to support redundant WinCC OA projects a breeze. Placed in the VirtualHost configuration files, the directive BalancerMember allows adding as many destination servers for a host name as desired, with different kinds of options such as timeouts and parameters affecting the likelihood of using a specific server over others (Apache HTTP Server 2015). By using this feature, the proxy will have more options to direct the client to a hopefully responding server while using only a single host name.

Generally this directive is used for load-balancing purposes, however with a simple setup it is also suitable for handling a typical redundant server use-case, where only one server is considered functional at a time, with a back-up server being in stand-by. In this use case, only two BalancerMember entries are specified, one for the main server in the redundant WinCC OA server pair, and another one, with an additional parameter which designates the server as a “hot spare”, meaning it will only ever be selected if no other servers are available. Below is a configuration example (without the authz parts) using the BalancerMember directive to achieve the described functionality.

```
<VirtualHost *:443>
        ServerName myhost

        # Declare a group of proxy destinations.
        <Proxy balancer://cluster>

                # Specify primary server to forward connections.
                # Additional parameter sets timeout to 5 seconds,
                # after which other BalancerMembers will be tried.
                BalancerMember http://mainserver:8080 retry=5
```

```
            # Specify backup server.
            # Additional parameter sets the entry as the “hot spare”.
            BalancerMember http://backupserver:8080 status=+H
        </Proxy>
        ProxyPass / balancer://cluster/
        ProxyPassReverse / balancer://cluster/
</VirtualHost>
```

4.6 Connection flow during UI startup from client to a redundant server through the proxy

Figure 3 visualizes how the client would connect into a redundant WinCC OA project to fetch project files during startup in a case where there are two proxy machines, and both proxies and the main server of the redundant WinCC OA project are online. As the client connects to the domain myApp.scada.cern.ch, a list of IP-addresses is received, containing addresses of the two proxy machines in random order. The client will try the first one (e.g. Proxy 1 in Figure 3), and if it is found to be responding, the client will remember it and keep using it as long as it keeps responding. If Proxy 2 happened to get selected first, it would work just as well.

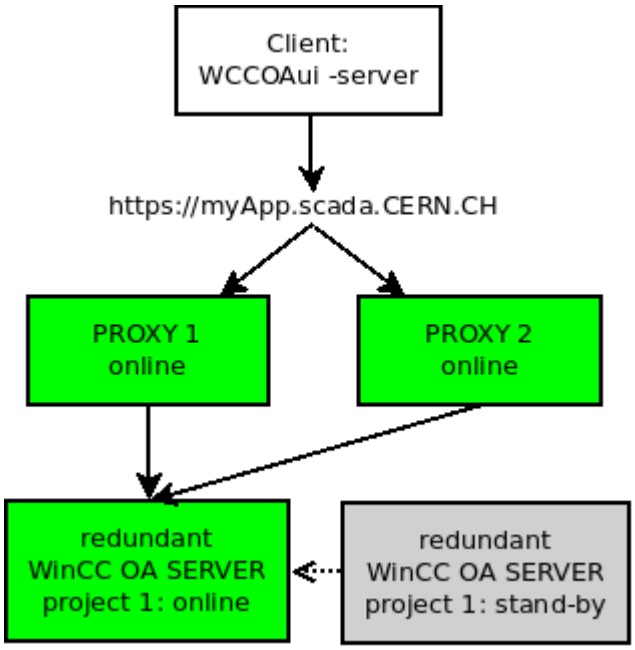


Figure 3. Two proxies and a redundant WinCC OA project.

Upon receiving a new connection, the proxy will request the client to authenticate, and will match the provided credentials against the CERN users database. On successful authorization, the proxy will forward the connection to the main WinCC OA server, and starts passings the files to the client. Meanwhile, the back-up pair of the redundant WinCC OA project is constantly polling the main server, ready to take over in case the main server is found to go offline.

In case Proxy 1 was unreachable for a reason or another, the client may try it once, and then proceed to try connecting to the second proxy, and again forward the connection to the main WinCC OA server, and start retrieving project files (Figure 4).

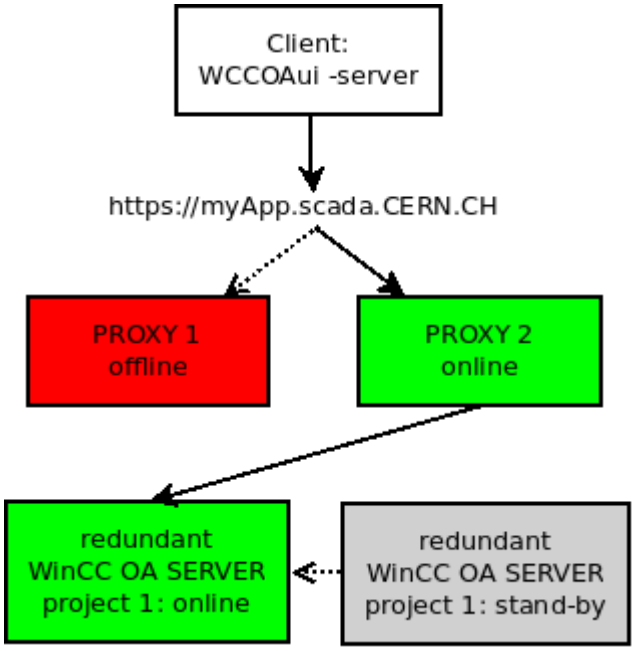


Figure 4. First proxy has become unreachable, all client connections go through Proxy 2.

Finally, in case the main server in the redundant pair has gone offline, the back-up WinCC OA server will automatically take over and start accepting connections (Figure 5). Proxy 2 will know from its VirtualHost settings to try the back-up server when the host name in

the first BalancerMember entry has stopped responding. The back-up server is still polling the main server in case it becomes operational. In such case it will fall back to its back-up status and stops receiving connections, letting the main server take over once again. Proxy 2 will notice that the back-up has stopped responding, and will try the main server again, and the situation returns to what is described in Figure 4.

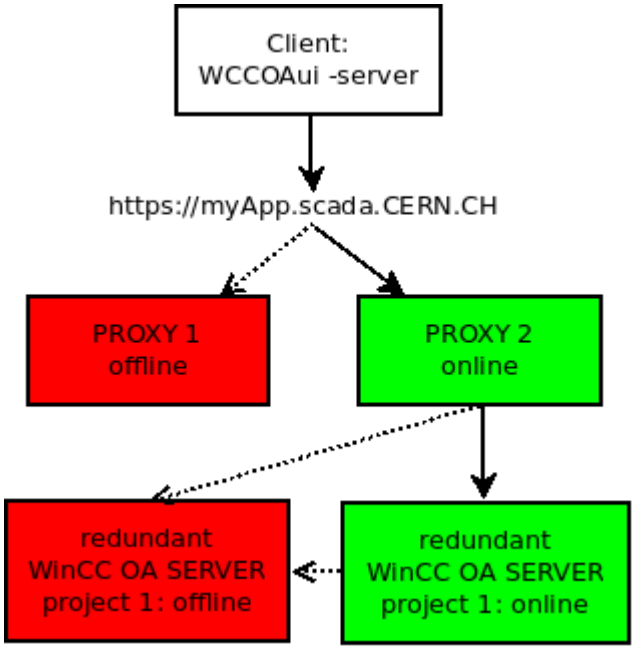


Figure 5. Proxy 1 and the main WinCC OA server have gone offline.

4.7 Dynamic sub-domains

To provide the URLs which are used to access applications through the proxy, the first solution that comes to mind, is to have the proxy host name, followed by a slash and an application name, such as: https://scada.cern.ch/myApp. Unexpectedly, the UI does not allow this approach, because the UI only cares about the host name part, and discards everything after the slash. After researching the options, it turned out that an even better alternative is possible by making use of a so called “wildcard” DNS record on the DNS server itself.

It is possible to set up the proxy so that user-friendly URLs , instead of very long URLs, can be used to connect to any number of machines through the proxy, with a very little amount of configuration needed when adding new proxy destinations or removing old ones.

By using a “wildcard” DNS record, all sub-domains of a domain point to a chosen IP-address or domain (RFC 1034, 1987). Example:

*.scada.CERN.CH -> scada.CERN.CH

The proxy will recognize which sub-domain name was used when connecting, and use the VirtualHost configuration specified for that sub-domain. If the sub-domain is not specified in any configuration, it can be pointed to a default location or hand out an error.

Thus any number of VirtualHost configurations can be specified to make new sub-domain names available, and they can be pointed to different (or same) locations. Figure 6 provides a visual representation of the case.

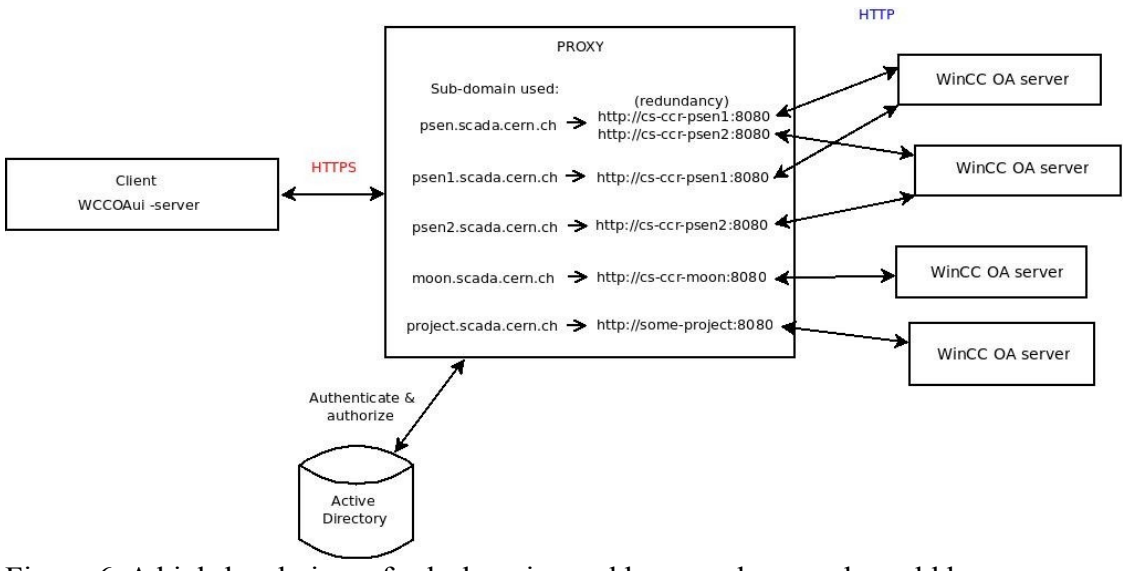


Figure 6. A high-level view of sub-domains and how scada.cern.ch could be mapped to point to different WinCC OA servers.

Adding a new configuration for a sub-domain is simple. One only needs to add a new file to a designated directory, containing a new VirtualHost section, and restart httpd. This file also contains the authentication and authorization settings to be used with the sub-domain. Similarly, removing the configuration only requires removing the file, and a restart of httpd. This is not a custom interface made specifically for this purpose, but just the way httpd configuration files work, making it very convenient to do it like this. With a redundant proxy, these operations need to be made identically on all related machines.

A Python script was implemented for the purpose of automatically generating and updating these files from a database. The script is part of the ”httpProxyConfigurator” component (see chapter 5.1.2).

An example of a full VirtualHost configuration for a single project could look like the following:

```
<VirtualHost *:443>
        ServerName psen.scada.CERN.CH
        <Proxy balancer://cluster>
                        Order deny,allow
                        Allow from all
                        BalancerMember http://cs-ccr-psen1:8080 retry=5
                        BalancerMember http://cs-ccr-psen2:8080 status=+H
                        AuthName "PSEN Authentication"
                        AuthType Basic
                        AuthBasicProvider ldap
                        AuthzLDAPAuthoritative On
                        AuthLDAPURL
                        "ldap://cern.ldap.url/OU=Users,OU=OrganicUnits,DC=cern,DC=c
                        h?sAMAccountName?sub? (objectClass=person)"
                        AuthLDAPBindDN "username"
                        AuthLDAPBindPassword "password"
                        Require ldap-group CN=my-egroup-name,ou=e-
                        groups,ou=Workgroups,dc=cern,dc=ch
        </Proxy>
        RequestHeader unset Authorization
        ProxyPass / balancer://cluster/
        ProxyPassReverse / balancer://cluster/
</VirtualHost>
```

The only new addition compared to the previous configuration examples is the presence of the RequestHeader directive. If the user is not added to the accessed WinCC OA project beforehand, the WinCC OA HTTP server will keep asking authentication repeatedly on every file being fetched, and pressing cancel will result in the error "Host requires authentication". The solution is to strip the Authorization header in the proxy, so that authentication is asked only initially to access the proxy. This is what the setting "RequestHeader unset Authorization" is for (Apache HTTP Server 2015).

4.8 Seconday objectives: WebUI, UltralightClient

WebSockets, which the WebUI makes use of, will work out of the box only on Apache 2.4 and newer, with mod_proxy_wstunnel enabled. Current Apache version on SLC6 machines is 2.2.

Shibboleth SSO (single sign-on) works as expected, making it possible to secure any kind of normal web page or generic web application running in a web browser, such as the UltralightClient.

5 Results

As an overview of the value brought, the proxy + WinCC OA HTTP file-share solution avoids the problems of SMB/NFS file-shares and OWS (Operator Work Station) specific configuration files. Strong authentication and authorization is provided by the existing CERN authentication infrastructure, including e-groups. Transfer of files is secured by HTTPS, and HTTPS-to-HTTP offloading on the proxy level means there is no certificate management required on any of the WinCC OA servers. The solution provides user friendly and easy to remember access points to applications (URLs) and reliable access to

redundant WinCC OA systems. The proxy itself is redundant to avoid a single point of failure and requires no maintenance effort, except when adding new or removing proxy machines, and the generation of httpd configuration files has been fully automated. The solution has been initially overviewed and accepted by IT Security.

5.1 Produced Components

Once establishing a streamlined way of deploying and using the proxy and setting up the WinCC OA project and the embedded HTTP server to accept connections, two components were produced to handle their installation.

In addition to the following components, an existing web service called the ICESAS (Industrial Controls Engineering SCADA Application Service portal) was extended by the application responsible to provide an interface for managing project settings related to the proxy usage (Figure 7), namely the application URL (such as https://myApp.scada.cern.ch) which serves as the application entry point through the proxy, the HTTP port number which will be used by the embedded WinCC OA web server, and e-groups which dictates which groups of users are permitted to access the application.

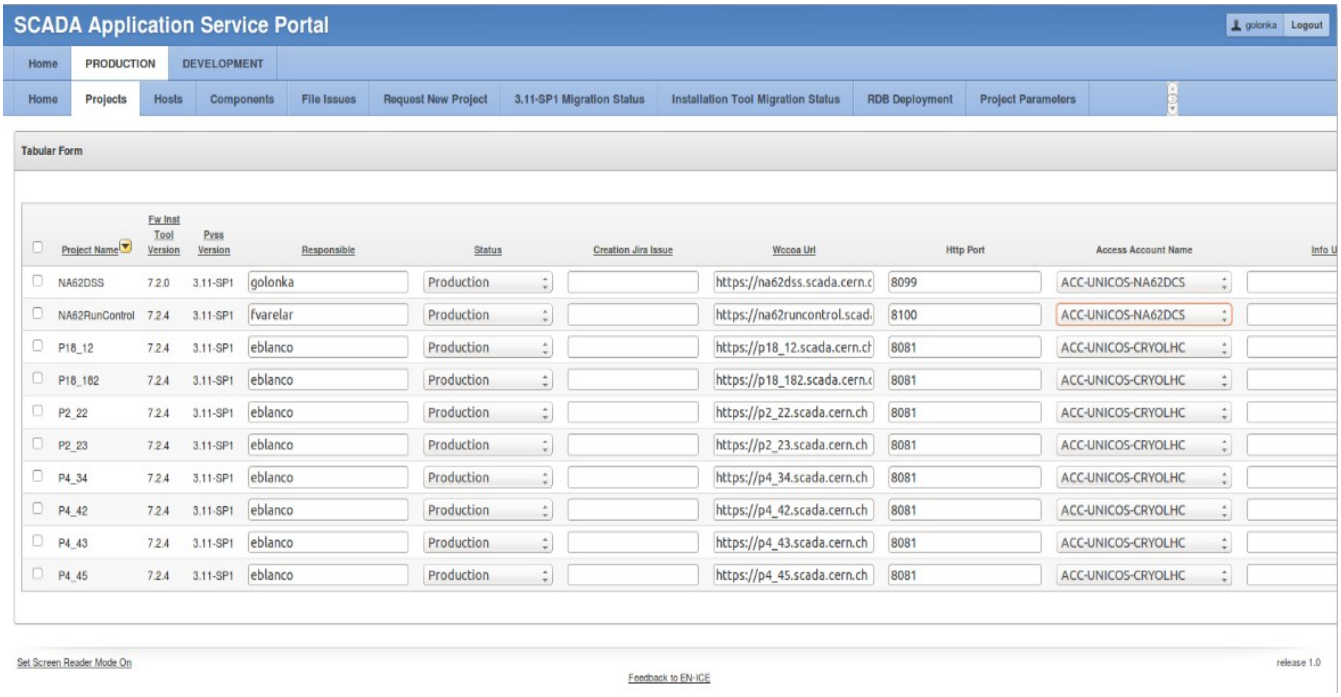


Figure 7. A view of the ICESAS web portal.

5.1.1 fwHttpFileServer

While the HTTP server itself is already built-in in WinCC OA, and can be started by simply executing a provided CTRL function in a CTRL manager, however to augment its usability, it was wrapped inside a component, with the added ability to configure the HTTP server port, some redundancy related checks, and the benefit of having component post-install scripts to apply additional configurations.

The component is now part of the JCOP component framework, and is used to install, configure, and start the HTTP file server in a WinCC OA project. The installation and removal processes of the component are handled by the standard JCOP framework installation tool. There are no dependencies to other framework components, however essentially the httpd proxy service needs to be set up to make use of this component, due

to the changes made into the project during installation (HTTP access to the project will be restricted to the proxy machines).

A special post-install script was written which will be executed after all component files and datapoint definitions have been copied over to the project. In this component, these files include CTRL libraries and scripts for running the WinCC OA HTTP server and making database queries, a panel for manually changing the HTTP server port and restarting the server, and one new datapoint definition for the purpose of storing the HTTP server settings. Currently, only the HTTP server port is being saved into this datapoint.

In the post-install script, the project's HTTP server port is retrieved from a database and saved into a datapoint for use when the HTTP server is started. Additionally a configuration entry is added to the main project configuration file, which will prevent access to the HTTP server from everybody except the proxies. This is done by allowing connections only from specified IP-addresses. Finally, a new CTRL manager is appended (or restarted, if a CTRL manager with a specific id number already exists) to the project for the sole purpose of running the HTTP server.

5.1.2 httpProxyConfigurator

This component can be deployed on a SLC6 Linux server with Apache httpd and TLS module and certificate already installed, to prepare it to act as a proxy for starting up WinCC OA applications.

This component is essentially just a zip file containing a default VirtualHost configuration for all sub-domain names that do not exist in other configuration files, and a default HTML page for them. Additionally, a Python script is included, which will query the database for WinCC OA projects, and generates, updates and deletes VirtualHost

configuration files accordingly as described in previous chapters. In production, running the script should be automated by setting up a cron job for example. An additional parameter may be provided to the script to get additional output of changes in the configurations.

5.2 Performance

A simple test was executed to measure differences in start-up times using the proxy setup and the embedded WinCC OA HTTP server, compared to the traditional SMB/NFS based start-ups. One of the largest SCADA applications, PSEN, which is used to monitor the CERN electrical network, and which consists of a very large number of files, was used for the test. The results of the test can be seen in Table 1.

Table 1. Times measured when opening a typical WinCC OA application using different methods.

Application access method	Seconds to panel open
Samba file-share	34
Local start-up	21
Win WCCOAui -server, no cache	23
Win WCCOAui -server, cache	16
Linux WCCOAui -server, no cache	14
Linux WCCOAui -server, cache	14

From table 1 can be concluded that using the proxy + HTTP based file share is a lot faster than a Samba based startup, and on Windows almost as fast as a local start. Startup from Linux is faster than local startup even, this is because the local launcher performs some additional file and synchronization checks. There is no negative impact whatsoever in terms of application startup time in using the proxy and the WinCC OA embedded HTTP server.

6 Summary

The project was the author's first assignment at CERN, and therefore the project started somewhat slow, as there were simultaneously a lot of new ground to cover: Learning to use the WinCC OA framework, Apache httpd configuration, and authentication and authorization methods. The implementation process was however very goal-oriented, and the various objectives were tackled in a rather organized manner.

All in all the project was a success and no major issues surfaced which would prevent its intended purpose as an alternative start-up method to address the currently standing problems with the previous SMB/NFS file-share approach. The proxy is deployed, and some selected projects are adopting it into initial test use as an alternative start-up method.

The project showed that a solid, proven third party web server may be what it takes to properly secure poor or otherwise simple web servers, which have been recently implemented into products due to new kinds of accessibility requirements (web integration), not only in industrial surveillance software, but in general with the rise of the Internet of Things. As such, future use of the proxy may include similar use cases in securing other CERN web applications particularly in the SCADA section, and possibly other hardware with embedded web servers, such as high voltage system mainframes.

The results are also to be presented at the upcoming ICALEPCS 2015 conference (International Conference on Accelerator and Large Experimental Physics Control Systems) in Melbourne, which can be interpreted as CERN having some confidence in the solution.

References

Apache HTTP Server. 2015. Accessed on 1.10.2015. https://httpd.apache.org/

Apache Authentication and Authorization. 2015. Accessed on 1.10.2015. https://httpd.apache.org/docs/2.2/howto/auth.html

Apache Core Features. 2015. Accessed on 1.10.2015. https://httpd.apache.org/docs/2.2/mod/core.html

Apache mod_proxy. 2015. Accessed on 1.10.2015. https://httpd.apache.org/docs/2.4/mod/mod_proxy

Ben Anderson. Why Web Browser DNS Caching Can Be A Bad Thing. 2011. Accessed on 1.10.2015. http://dyn.com/blog/web-browser-dns-caching-bad-thing/

Flush DNS. Accessed on 1.10.2015. https://www.whatsmydns.net/flush-dns.html

June 2013 Web Server Survey. 2013. Accessed on 1.10.2015. http://news.netcraft.com/archives/2013/06/06/june-2013-web-server-survey-3.html.

Linux @ CERN. 2015. Accessed on 1.10.2015. https://linux.web.cern.ch/linux/docs/kerberos-access.shtml

Linux @ CERN. 2015. Accessed on 10.10.2015. http://linux.web.cern.ch/linux/scientific6/docs/shibboleth.shtml

Siemens SIMATIC WinCC OA. 2015. Accessed on 10.10.2015. http://www.siemens.fi/fi/industry/teollisuus/tuoteuutiset/simatic_wincc_openarchitecture_312.htm.

Red Hat Customer Portal. 2015. Accessed on 1.10.2015.
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/sso-config-firefox.html

RFC 1034 Domain Names. 1987. Accessed on 1.10.2015.
https://tools.ietf.org/html/rfc1034

RFC 1945 Hypertext Transfer Protocol. 1996. Accessed on 1.10.2015.
https://tools.ietf.org/html/rfc1945#section-11.1

RFC 2308 Negative Caching of DNS Queries. 1998. Accessed on 1.10.2015.
https://tools.ietf.org/html/rfc2308

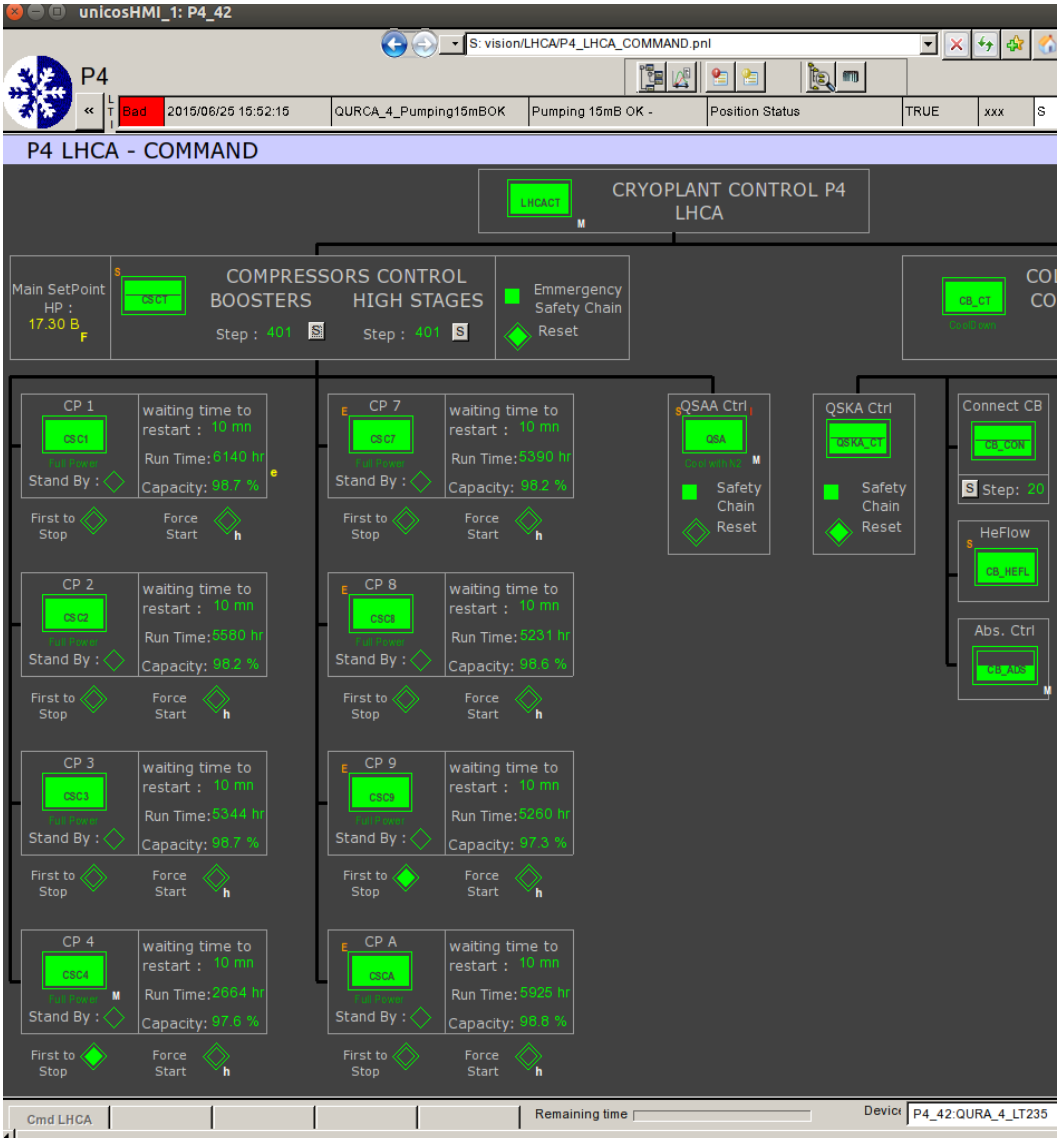
RFC 4120 The Kerberos Network Authentication Service. 2005.
https://www.ietf.org/rfc/rfc4120.txt

The Technology Chronicle. 2013. Accessed on 1.10.2015.
http://thetechnologychronicle.blogspot.in/2013/11/dns-round-robin.html

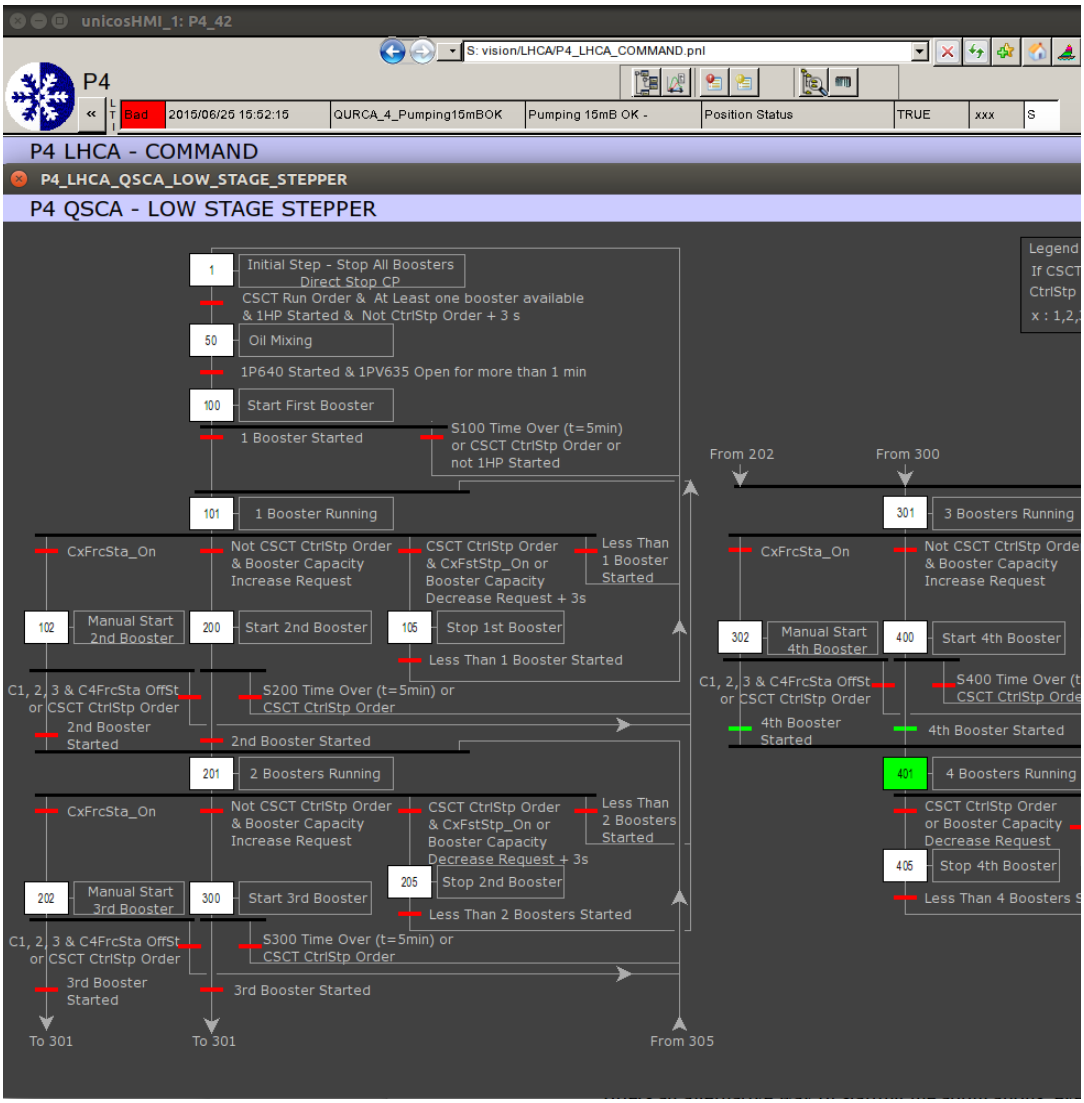
Wikipedia CERN page. 2015. Accessed on 1.10.2015. https://fi.wikipedia.org/wiki/CERN

Appendices

Appendice 1. Example WinCC OA application.



Appendice 2. Example WinCC OA application.



Appendice 3. Example WinCC OA application.