ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA104

IT INFRASTRUCTURE FOR SAFETY SYSTEMS DEVELOPMENT AT ESS

S. Armanet*, R. Mudingay†, European Spallation Source, Lund, Sweden

Abstract

The Control System Infrastructure team has deployed a dedicated isolated environment to support Safety Systems development at ESS.

We have tried to take advantage of our standardised infrastructure components for controls like virtualization, centralized storage, system orchestration and software deployment strategy. Because we already have all these components in place for our Control System IT infrastructure we have decided to treat engineering workstations as disposable components in an isolated and dedicated virtualized environment. We have designed the environment to control who and when users can access the development environment, from which device, to which workstations and what they can run in this environment.

INTRODUCTION

Safety systems design and development are critical parts of ESS development and operation phases. They are controlled and reviewed by the Swedish nuclear safety authorities and a priority in our cyber security plans. The development process has to follow a well defined security plan and risk assessment and the development environment has to follow these rules. Basically these rules are to answer these questions:

- who can access the development environment?
- when can these users can access it?
- from which devices?
- to which engineering workstations?
- to do what?

In the ICS infrastructure team we have decided to implement engineering workstations has disposable workstations so that we simplify the requirements on managing Windows workstations. These workstations run on a network that is completely stand alone and isolated from any other network.

The only way to access them is using the Remote Desktop Protocol from controlled clients. This environment is fully virtualized which permit us to achieve these goals in a flexible and automated way.

We provide all required tools to program safety systems in a controlled way and we are evaluating a way to interact with external hardware from this environment by using dedicated transfer workstation with ad hoc controls and restrictions.

SYSTEM DESCRIPTION

Infrastructure

Because our orchestration process allows us to easily target where we want to deploy virtual machines, we have decided to split our infrastructure (see Fig. 1) in different virtualization cluster and network layers [1].

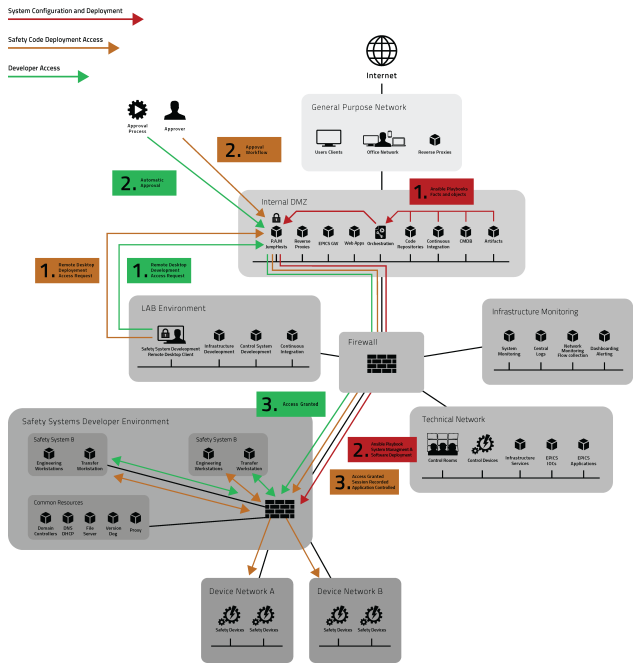

Figure 1: System description.

This way we can easily adapt each cluster to its specific use case and manage network connectivity specifically for each cluster. We use a simple open source Virtualization solution for our clusters and our orchestration tool talk to it's API to deploy virtual machine. This is to avoid complex virtualization solution and keep the management of different cluster easy within a small team.

The virtualization solution supports multi-user, distributed management, 2 factor authentication, High availability and live VM and storage migration. VM location are defined in our CMDB based on network membership, functional group membership or host based definition. Our clusters are composed of at least 3 nodes for redundancy, maintainability and high availability.

We have a shared storage back-end.At the moment our storage is based on replicated ZFS NAS. We use ZFS snapshots send/receive as backup tool for safety systems.

Our plan is to deploy our clusters across 3 data centres/server rooms to avoid split brain scenario. Our storage cluster will also be redundant across these 3 server rooms.

We will share a common storage cluster but with dedicated pools for each environment.

Orchestration

We use Ansible as automation tool. Our roles and playbooks are hosted on an internal gitlab repository and use our CMDB (CSentry) as dynamic inventory.

* stephane.armanet@esss.se
† remy.mudingay@esss.se

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA104

We use git as version control software. All our roles and playbook are automatically tested in the continuous integration pipeline using Ansible Molecule.

We use Artifactory as internal software repository for all kinds of packages (Linux RPM, conda, simple archives ...). AWX is used as job scheduler and allow us to delegate to users the execution of certain tasks.

For instance, safety system engineers are allowed to upload their PLC code reports that include safety code checksum by executing a predefined job using AWX.

Ansible is used to create VM, deploy and configure software, ensure that each system is in the state that has been described by the configuration management process.

The orchestration also permit to automatically configure network switches and assign VLANs according to our CMDB.

Engineering Workstations

Each safety system run on it's own isolated network. Some common resources are shared across these networks for cost and maintenance efficiency. The common resources are also part of the isolated environment on a separate virtual network. An internal firewall sits in the middle of all these virtual networks to prevent safety systems to reach each others as well as permit access to these common resources.

It could also be used as a NATing device to allow communication from a dedicated transfer workstation to external devices for programming and upgrades. This could only be done with enforced connection control and authorisation management to the external devices.

We install Windows workstations and infrastructure services using our orchestration process relying on these tools:

- Ansible for configuration management [2, 3]
- gitlab for code and playbook version control
- CSentry has CMDB and system tuning [4]
- Ansible tower (AWX) as task scheduler and deployment system [5]
- Artifactory as internal software repository [6]

Linux systems are installed and updated by a scripted installation process managed by an Ansible playbook. Windows systems are installed by cloning a template.

Windows systems are managed as disposable systems, we manage updates by re-deploying these workstations. We provide all ad-hoc components into the isolated environment as the orchestration allow us the provision any number of instances for these components:

- Network services: DNS and DHCP. We assign predefined MAC addresses and DHCP reservation to our virtual machine from as defined in our CMDB.
- Authentication and Windows domain membership: we have deployed a dedicated Windows Active Directory Forest that has a trust relationship with our central Forest. This allows the registration/suppression of Windows workstations without the need to open Windows communication outside of the isolated environment. The trust relationship allow users to use their central forest account to authenticated on the engineering workstations.
- Dedicated PLC code repository: this is to allow users to store their project outside of the engineering workstations with version control. It could also act as a source to transfer PLC code to the real devices.
- File Services: as our workstations are disposable, this is to allow users to save documents that are outside of the version controlled projects.This way we can re-deploy their workstations at any time(ie: software updates). This also act as temporary internal software repository, anti-virus server and back end storage to send reports with code checksums outside of the isolated environment.
- HTTP proxy: to allow antivirus updates from the file server.
- Syslog relay: to send all logs to an external logging facility.

User Access

Access to the engineering workstations are only allowed through a privileged access management (P.A.M) or admin bastion solution.

Users can only access their workstations using the RDP protocol. File transfer are disabled and RDP sessions could be recorded. Depending on how critical the system is, we can enforce RDP connection to come from a controlled device.

We use several functions of the P.A.M to achieve a good user control:

- 2 Factor Authentication: to ensure that the user logged in is the right person.
- Per user access rules: a user can only access a specific list of workstations from a well known remote desktop client.
- Adaptive Approval workflow: A user can have automatic approval (with the requirement of filling an access form) to access a development workstation. When there is a need for real device access, the workflow could be enforced to explicit approval by a system owner. In that case the user would have to connect to a specific transfer workstation where these rules would apply.
- Program execution control: even if we control which software has been installed on the engineering workstations, the P.A.M provide the capability to block the execution of applications. This could help to enforce the control of transfer workstations.
- Session recording - 4 eyes monitoring: RDP sessions can be recorded and/or followed by an external administrator. This feature could also be used to secure code transfer to external devices.

We try to avoid users to work from their own workstations (personal computers connected to the Office network). We have deployed hardened linux workstations to be used as remote desktop clients (Thinclient). These workstations are not connected to the office network.

Users can only execute the remote desktop client from these workstations. They can't run any other software or

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA104

open a shell connection. USB storage is disable. Local access (BIOS,boot loader) are password protected.

Users have to be explicitly allowed to connect to one of these Thinclient. Local firewall rules prevent the users to open connection to other remote servers than the P.A.M.

These Thin Clients are supposed to be installed in a locked room only accessible by safety system engineers.

CONTROLS AND MONITORING

Controls

Figure 2 shows the controls and monitoring structure:

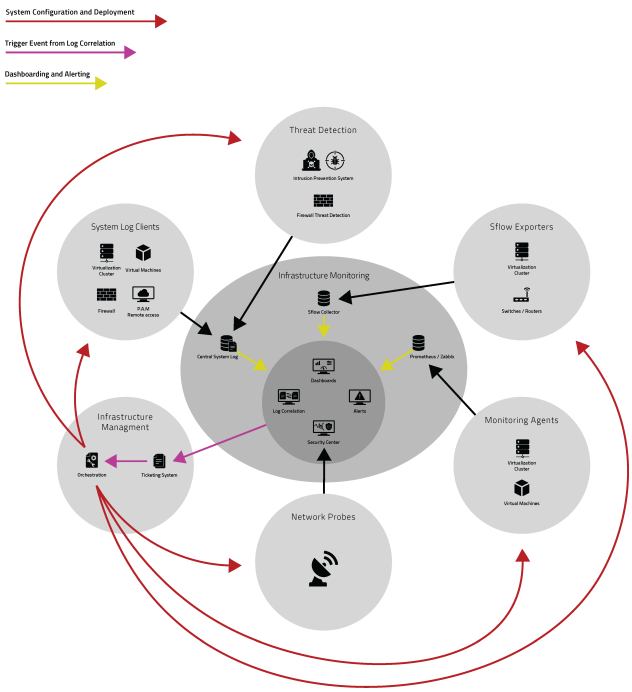

Figure 2: Controls and monitoring structure.

Software Control All software are deployed form a dedicated software repository to each safety system. This way we have a strict control on what can be installed on the engineering workstations and their versions.

Software are installed in a VM template. If there is a need for a new software or for updates, our process is to re-deploy the workstations. All PLC code is version controlled using a dedicated software. This software also acts as a central repository for up to date PLC code. It could be used as source repository for real device programming or firmware upgrades (through dedicated transfer workstation). The PLC code repository can use the file server to host it's database backup.

Users can also save their documents on a personal share on the file server.

Users needs to send report containing PLC code safety checksums. They use a delegation from our orchestration tool to upload they're PDF report outside of the isolated environment. The orchestration provide user authentication and controls on which file can be uploaded. The file server acts as a pivot in that process for virus checking. We try to avoid interaction between system administration and user content. We backup the virtualization plate-form as a whole so that we don't need to get access inside each workstation.

The P.A.M provide application execution control capabilities. We could define a white list of application that a user is able to run on an engineering or transfer workstation. This also give us the capability to alert in case of bad behaviour of a workstation.

Logs are centralized to our central logging solution. This helps to perform correlation analysis and monitoring regarding the technical network as a whole. Network connection inside the isolated environment are managed by the virtualization plate-form. Physical devices network access are automatically assigned by our orchestration process (CMDB + Ansible + Radius).

Monitoring and Threat Prevention

The safety system environment is monitored like all others systems part of the Control System infrastructure. We use standard monitoring and threat detection components and try to centralize as much as possible these information to enable log correlation.

Our monitoring infrastructure is hosted on a dedicated cluster for efficient monitoring and combined different kind of tools:

- log centralisation: we use syslog as much as possible but our central syslog tool is a combination of syslog-ng (archiving) and graylog (indexation, correlation, alerting). Graylog provide some nice features for non-syslog kind of logs (windows, java applications ...)
- System Monitoring: we mainly use Zabbix as monitoring solution but as our infrastructure is growing we are slowly combine it with Prometheus which provide a nice and efficient time series database.
- Threat Detection: we use a L7 next generation type of firewall which provides its own threat prevention capabilities combined with an open-source IPS deployed across most of our networks. Our Privileged Access Management solution also provide in-depth remote desktop session analysis and blocking capabilities. We use Graylog for alerting and dashboarding
- Sflow: we collect sflowsamples from our main switches and routers as well as our virtualization infrastructure to be able to see inter-VMtraffic. We centralize all streams on a powerful open-source collector (Elastiflow) which provides useful dashboards.
- Dashboarding and alerting: we mainly use Grafana connected to the Zabbix and Prometheus databases. We have different kinds of alerting media (Web, slack, email). Alerts are mainly coming from Zabbix, we are currently building alerts based Prometheus.
- Network probes and Security centre: This is an ongoing project. We are currently studying the possibility to deploy network scanner connected to a central security centre. The security centre will be linked to a ticketing

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA104

systems which will help us to schedule system upgrades as well as react to emergency situations. This system will be linked to our orchestration solution (manual or automatic job scheduling).

CONCLUSION

We have built a scalable development environment that can host projects isolated from each others. We have also built a cost effective infrastructure by using mainly open source software and shared resources (virtualisation, compute and storage). Some components are still in under development like log correlation, threat detection and SIEM (Security Information and Event Management), but we are confident that we have a sustainable and secure plate-form to support safety and critical systems developments for ESS.

REFERENCES

[1] *Practical Overview of Implementing IEC 62443 Security Levels in Industrial:* https://www.schneider-electric.com/en/download/document/998-20186845/

[2] Ansible: https://www.ansible.com/

[3] Molecule: https://molecule.readthedocs.io/en/stable/

[4] CSentry: https://gitlab.esss.lu.se/ics-infrastructure/csentry

[5] AWX: https://www.ansible.com/products/awx-project

[6] JFrog Artifactory: https://jfrog.com/artifactory/