Intrusion-Tolerant Eclipse SCADA

André Nogueira · Alysson Bessani · Nuno Neves
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract – The paper describes an open-source SCADA system that was enhanced with intrusion-tolerant capabilities, focusing on the aspects related to the challenges that were addressed and the architecture of the solution. Some preliminary performance results are also included.

1. Introduction

Supervisory Control and Data Acquisition (SCADA) systems form the backbone of most critical infrastructures, including the power grid. To enrich the control of critical infrastructures, modern information and communication technologies are being integrated, sometimes with interconnections to corporate networks. This can cause SCADA systems to become exposed to similar attacks as Internet-based systems, which may lead to operational failures.

Often, companies have depended on firewalls and intrusion detection systems (IDS) to secure their critical infrastructures. These technologies can detect well-known threats and identify suspicious activities by analyzing the network traffic data. However, in most cases, they are unable to defend against novel attacks (associated with zero-day vulnerabilities) that permit attackers to compromise and perform malicious operations. For instance, in most IDS, there is always a vulnerable time window between the discovery of a new threat and its signature being available.

A major thread for SCADA occurs when the attacker gains access to the central component of the system – the SCADA Master. This component manages all operations under the SCADA supervision, and therefore an intrusion can result in a catastrophic scenario. By resorting to appropriate replication techniques, the SCADA Master could be enhanced to operate correctly not only in the presence of accidental faults but also when there are compromises. This new intrusion-tolerant SCADA would contain improved protection capabilities against malicious attacks, complementing more traditional security solutions [1].

We have developed such an intrusion-tolerant approach and applied it to the open-source Eclipse SCADA project. Generically, we resort to State Machine Replication (SMR) to manage multiple replicas of the SCADA Master. The key idea is to make replicas execute deterministically the same sequence of requests in such a way that, despite the failure of a fraction of the replicas, the remaining ones have the same state and ensure correctness of the offered services through a voting on their responses. The protocols were built to address arbitrary (or Byzantine) failures, which include ordinary problems like crashes but also (compromised) replicas that act maliciously.

In this paper, we briefly discuss several issues that must be addressed to integrate an intrusion-tolerant approach with a SCADA Master. We also give an overview of the architecture of our current prototype, where we tried to minimize changes to the existing system[1], and discuss the performance costs of adding the intrusion-tolerance capability based on a preliminary evaluation.

[1] http://www.eclipse.org/eclipsescada/

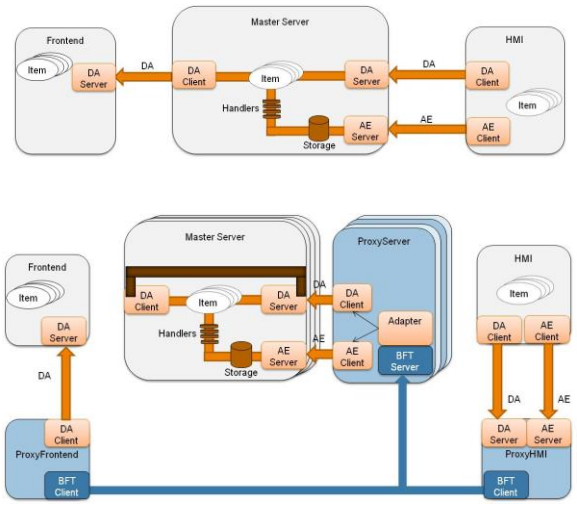

Fig. 1 Simplified view of the architecture of Eclipse SCADA (top), and intrusion-tolerant Eclipse SCADA (bottom).

2. Main Challenges

Here, we identify and discuss some of the main issues related to the integration of a Byzantine fault tolerant (BFT) replication library with the Master server of Eclipse SCADA.

Multiple communications. The Master encompasses several communication entry points (see Figure 1 (top)). To interact with the Frontends, which are connected to the RTUs/IEDs, the Master server uses a data acquisition (DA) protocol. In a similar way, while communicating with the Human Machine Interface (HMI), it resorts both to the DA and the Alarms & Events (AE) protocols. Consequently, the Master can receive simultaneously requests and replies from all these modules, which are executed in some random order. The effect is that in a replicated system, each replica can end up processing messages in a different order even if all replicas get the same set of messages. This architecture does not guarantee that the processing of requests will be deterministic, which a fundamental requirement of our replication approach.

Concurrency in processing. Internally, the Master server does not operate sequentially. The DA and AE subsystems have several modules that execute concurrently with multiple threads, enabling several requests to be processed in parallel. In a context of SMR, this concurrency is a major difficulty because the execution of the requests becomes non-deterministic across replicas.

Timestamps. In the AE subsystem, some modules retrieve information from the operating system to process. For instance, when an event is created, a timestamp is retrieved from the operating system and assigned to the event. In a replicated solution, it is necessary to ensure that all replicas generate the same timestamp for the same event to avoid the violation of determinism.

Asynchrony. Eclipse SCADA implements a publish/subscribe architecture. To receive data or events associated with an item, the HMI must subscribe to that item in the Master. After subscription, the HMI starts receiving messages asynchronously. The Master server can send to the HMI multiple messages in response to a single event reported by a frontend. As the HMI receives messages asynchronously from a set of replicas, the messages must contain some sort of information that enables it to understand in which context that messages were produced. Currently, this context information is not available in Eclipse SCADA messages.

Performance. Some critical infrastructures use Eclipse SCADA nowadays, confirming that its performance is suitable for deployments in the field. The current Eclipse SCADA only supports a single Master server. In a replicated version of the Master server, the performance may be affected due to the overhead introduced by the intrusion-tolerant library.

3. Intrusion-Tolerant Eclipse SCADA

Figure 1 (bottom) shows the architecture of our intrusion-tolerant Eclipse SCADA. It augments the existing system with three new components: *ProxyServer*, *ProxyHMI* and *ProxyFrontend*. These components were designed to address one of the key objectives of our integration, which was to avoid changes in the BFT library and minimize changes in the Eclipse SCADA, while solving the challenges described in previous section.

The *ProxyServer* is responsible to forward all Eclipse SCADA messages that come from the Frontend and the HMI to a Master replica. Each *ProxyServer* has a BFT server module, which is the server-side of the intrusion-tolerant library where a Byzantine fault tolerance protocol runs. This protocol is responsible to guarantee that the received messages are delivered in the same order even if a fraction of the BFT servers is malicious. After the messages are ordered, each BFT server passes to the *Adapter* module, which demultiplexes and forwards messages to the Master server using either the DA or AE client components.

The *ProxyHMI* receives all messages from the HMI and sends them to the *ProxyServer*, using the client-side part of BFT replication library. In this proxy, we have a DA server and AE server which simulate the servers available in the Master server. As a result, the HMI is not aware that it is communicating with a proxy.

The *ProxyFrontend* guarantees the communication between the Frontend and the Master server. This proxy employs the client-side part of the replication library to send all relevant messages that come from the Frontend to the Master server. When the Master server needs to communicate with the Frontend, the *ProxyFrontend* receives the messages from the client-side of intrusion-tolerant library and sends them using the DA client. We also propose a modification in the Master server, which guarantees that all messages that come from the Frontend are placed in the right component of the Master server – DA client. We propose channels between the DA server and the DA clients. This way, when the DA server receives a message from the Frontend, it forwards the message into the DA client of the item placing the message in its channel. A similar approach is followed with the *ProxyHMI*.

All these three new components and the proposed modifications allow us to convert the multiple entry points into a single entry point. This way, the Master server does not receive messages simultaneously and all messages that the Master server processes are ordered, ensuring the determinism necessary for SMR.

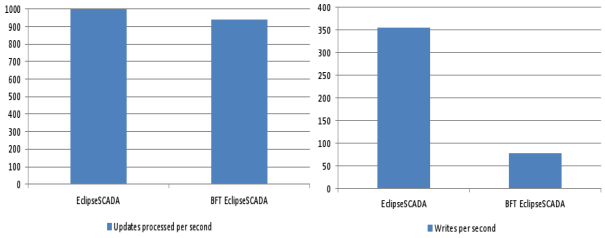

Fig. 2 Performance results for both solutions. Left side is about update processing and right side is related to writes in items.

4. Perliminary Evaluation

To develop the Intrusion-Tolerant Eclipse SCADA (BFT EclipseSCADA), we integrated a fault-tolerant state machine replication library called BFT-SMaRt [2] with the Eclipse SCADA. BFT-SMaRt is an open-source Java-based library implementing BFT state machine replication.

Figure 2 shows the results of some preliminary experiments conducted in a cluster of six machines interconnected by a Gigabit Ethernet switch. The machines have two quad-core 2.27 GHz Intel Xeon E5520, 32 GB of RAM, and run Linux kernel 3.13 and Java 1.7.0. We deployed the Eclipse SCADA in three machines: one Frontend, one Master server and one HMI; each one containing its application. In contrast, we used six machines to deploy the BFT EclipseSCADA solution: one Frontend, four Master servers and one HMI; each one containing its application and its proxy.

We show results for a scenario in which the Frontend contains a set of items that are being updated by a RTU 1000 times per second (left). As a result, the Frontend is sending to the Master server 1000 update messages per second. It is possible to observe a performance drop of 6% in the BFT EclipseSCADA. The causes for such performance drop are the additional communication steps required in the replicated version (6 vs 2 in the conventional system) and the lack of parallelism on message processing (to avoid problems with determinism).

Figure 2 (right) also shows results for a scenario where the HMI performs synchronous writes in an item in the Frontend (which would them be sent to a RTU). This means that, for each write operation, the HMI waits until the write operation is completed. We can observe that the BFT EclipseSCADA introduces an overhead of 78%. We checked that the intrusion-tolerant library only added an overhead of 12%. The remaining 66% was introduced by the necessity of extra communication. While in the Eclipse SCADA it takes 4 communication steps to perform a write operation, in the BFT EclipseSCADA it takes up to 12 steps.

Acknowledgements. This work was partially supported by the EC through project FP7-607109 (SEGRID), and through funds of the Fundação para a Ciência e a Tecnologia (FCT) with reference to UID/CEC/00408/2013 (LaSIGE).

References

[1] P. Veríssimo, N. Neves, M. Correia, Intrusion Tolerant Architectures: Concepts and Design, Architecting Dependable Systems, LNCS 2677, 2003.

[2] A. Bessani, J. Sousa, E. Alchieri. State Machine Replication for the Masses with BFT-SMaRt. Proc. of 2014 IEEE/IFIP Dependable Systems and Networks Conference, 2014.