The CRUTIAL way of Critical Infrastructure Protection

Alysson Neves Bessani Paulo Sousa Miguel Correia
Nuno Ferreira Neves Paulo Verissimo
Universidade de Lisboa, Faculdade de Ciências – Portugal

Abstract

Today, critical infrastructures like the power grid are essentially physical processes controlled by computers connected by networks. They are usually as vulnerable as any other interconnected computer system, but their failure has a high socio-economic impact. We describe a hierarchy of variations of a novel device for the protection of these infrastructures, the CIS. These devices are used to ensure that incoming/outgoing traffic satisfies the security policy of an infrastructure in face of cyber-attacks. However, a CIS is not a common firewall but a distributed protection device based on a sophisticated access control model. Furthermore, a CIS is intrusion-tolerant and self-healing, seeking perpetual unattended correct operation. A key feature of the proposed architecture is that it does not require any modification of the SCADA/PCS software already in use today.

Introduction

The protection of critical infrastructures (CIs) became a fascinating problem. At its root is the increasing dependence of modern societies on information and communication technologies (ICT). The Internet is today a web of sophisticated devices, fibers and electromagnetic signals, interconnecting computational and electronic systems all over the world, which makes the global society and economy move. However, tips of this web are extending their reach to physical infrastructures like power, water, gas, oil and transportation. Although the insecurity risks posed by Internet are fairly well known, this scenario raises the risk to a new level. The threat is no longer against systems with a limited economic value, like home banking or online shops, but against the infrastructures that support modern life as we know it. Furthermore, more than affecting data integrity or confidentiality, we may be talking about serious physical damage, for example to power generating and distributing infrastructures.

The operational risk of a system is proportional to the level of threat and degree of vulnerability (risk = threat $\times$ vulnerability). The threat to which CIs are subject is conspicuously high. Recent news have brought to light that some of those systems are already being targeted for extortion: their owners were forced to pay to avoid retaliation against their operation [7]. The terrorist threats and foreign country sponsored attacks have also been heard of.

Unfortunately, the level of vulnerability of CIs is not as low as needed due to several factors. CIs are physical/mechanical processes controlled electronically by systems, usually called SCADA (Supervisory Control and Data Acquisition) or PCS (Process Control System), composed by computers interconnected by networks. These control systems used to be based on proprietary solutions, which provided some (a rather weak) form of security, essentially by obscurity. However, nowadays CIs companies are employing standard hardware and software for efficiency reasons. Controllers are now industrial PCs running off-the-shelf operating systems (e.g., Windows or Linux), the networks are often wired (or even wireless) Ethernet, and control and supervision protocols are normally encapsulated on top of UDP/IP or TCP/IP. The defensive phrase often heard that "*the hackers don't know our systems*" is no longer true. Furthermore, our experience

with companies that operate CIs has shown that many times the technologies being utilized – most specially software – are not even best of breed, but older versions that are known to be plagued with vulnerabilities.

CI protection is harder to address than common ICT protection, due to the interconnection complexity of these infrastructures, which can lead to various kinds of problems. Think about the power grid, where there are geographically dispersed production sites, power distribution through different voltage level stations (from higher to lower voltage), until energy eventually flows into our houses. Both the production and distribution sites are typically controlled by SCADA systems, which are (remotely) connected to supervision centers and to the corporate networks (intranets) of the companies managing the infrastructures. The intranets, as expected, are linked to the Internet, to facilitate for instance the communication with power regulators and end clients, creating a path for external attackers. SCADA systems are also accessed remotely for maintenance operations, and sometimes equipment suppliers keep links to the systems through modems. From the point of view of keeping the system working this is all a good idea, but from a security perspective this looks very much like a recipe for serious problems [5]. This conclusion is far from being a speculation, as a few years ago the modem of a supplier was the entry door for an Internet worm into the supervision systems of a nuclear plant, which caused a near disaster.

On a more positive side, CI companies and governments have been showing a high level of concern about this state of affairs, and have been pushing forward a good set of security measures. However, a careful reading of the legion of guidelines being produced (e.g., NIST SP 800-82 [11]) shows a common trend that might be summarized in a couple of ideas: critical infrastructure protection is a problem of network security; it should be handled using secure communication protocols and perimeter protection through firewalls. These mechanisms are well known to be a first level of protection of undeniable utility and importance. However, they put the protection of CIs at the level of ICT security, which is not enough because of their criticality and societal relevance. Moreover, firewalls are known to be vulnerable to intrusions and denial of service attacks. If we consider only the vulnerabilities that may lead to intrusions in commercial firewalls, the National Vulnerability Database reports 9 in 2005, 15 in 2006 and again 15 in 2007. The total number of vulnerabilities reported for these three years is 79. We clearly need more than classical prevention security mechanisms like firewalls.

However, when building newer solutions, there are two important aspects that have to be taken into consideration:

- CIs feature a lot of legacy subsystems and non-computational components, like controllers, sensors, actuators, etc, which can not be modified for operational reasons or others, for a number of years to come.
- The main concern of the companies is to keep the CIs working with the expected level of service. Security mechanisms that stand in the way of operation are unacceptable and will not be deployed.

In this paper, we describe work done in the context of the European project, CRitical UTility InfrastructurAL resilience (CRUTIAL), with a particular emphasis on a novel architecture and protocols that preserve legacy, and on a new protection device called CIS, which provide incremental protection, assuring different resilience to different parts of the infrastructure, according to their criticality.

The CRUTIAL Architecture

One of cruxes of the CI protection problem is the security of the interconnections among the infrastructure providers, regulators, operators and others. To tackle this problem, we need an architecture that models and allow us to reason about this reality.

In the CRUTIAL reference architecture, the whole infrastructure architecture is represented as a WAN-of-LANs. This topology allows simple solutions to hard problems such as legacy control subnetworks, and

interconnection of critical and non-critical traffic. Typically, a critical information infrastructure is formed by facilities, like power transformation substations or corporate offices, modeled as collections of LANs, which are linked by a wider-area network, such as dedicated phone lines or the Internet, modeled as a WAN.

This architecture allows the definition of realms with different levels of trustworthiness, and the problem we need to solve is how to defend realms from one another, i.e., a LAN from another LAN or from the WAN. However, there is virtually no restriction to the level of granularity of a realm, which can go down to a single host. In consequence, this architecture supports solutions both with outsider threats –protecting a facility from the Internet– and insider threats –protecting a highly critical host from other hosts in the same physical facility, by locating them in different LANs.

Protection of a LAN is made by a device called the *CRUTIAL Information Switch* – CIS (see Figure 1). A CIS provides two basic services: the *Protection Service (PS)* and the *Communication Service (CS)*. The PS ensures that the incoming and outgoing traffic in/out of the LAN satisfies the security policy of the organization that manages the LAN. CS supports secure communication between CIS and, ultimately, between LANs. The CS provides secure channels and multicast between CIS. Although the CIS supports these two services, in this paper we will be mainly concerned with the protection service.

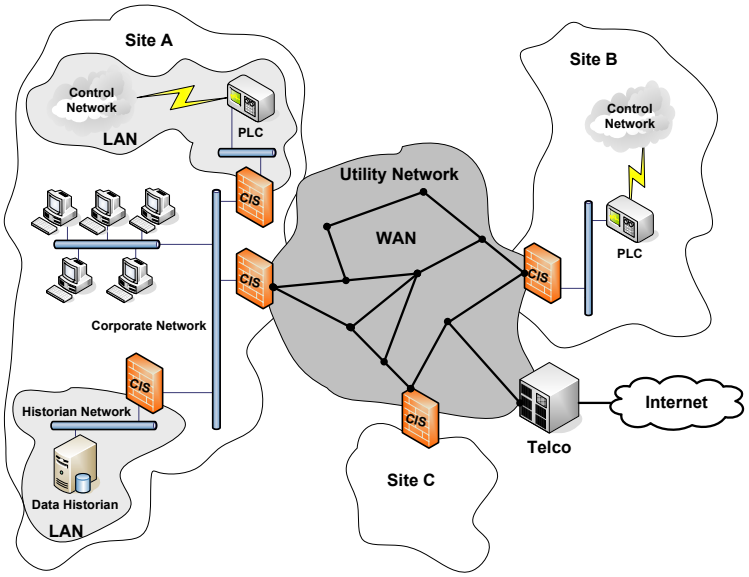


Figure 1: WAN-of-LANs connected by CIS.

At a first glance, a CIS might be instantiated as a firewall. However, as we explained before, this falls short to the challenge. Instead, a CIS is a distributed protection device with the following main characteristics. Firstly, it resembles a *distributed firewall* [3], since it can be deployed not only at the network border but inside the networks to better protect critical equipment. Secondly, the CIS uses a *rich access control model* that takes into account the involvement of different organizations and allows the access control rules to depend on context information [1]. Thirdly, the CIS is *intrusion-tolerant* [13], i.e., it operates correctly even if there are intrusions in some of its components, with the objective of withstanding a high degree of hostility from the environment.

Intrusion-tolerant services have been being proposed in the literature for some years (see [8] for a brief survey), however, the CIS design has to be essentially different from those services for two reasons. The first is that a firewall-like component has to be transparent to protocols that pass through it, so it can not modify the protocols themselves to obtain intrusion tolerance. Second, this also means that internal CIS intrusion tolerance mechanisms have to be transparent to recipient nodes, as such nodes can not protect themselves

from malicious messages forwarded by faulty CIS components not satisfying the security policy. These two challenges are not trivial and our solution requires a different approach than previous works on intrusion-tolerant services.

The CIS Protection Service

From a functional point of view, the CIS Protection Service works like a firewall. It captures packets that pass through it, checks if they satisfy the security policy being enforced, and forwards the approved packets, discarding those that do not satisfy the policy. However, several characteristics of the CIS make it a unique protection device:

Distributed policies. CIS can be used in a redundant way, enforcing the same policies in different points of the network. An extreme case in the SCADA/PCS side is to have a CIS in each gateway interconnecting each substation network, and a CIS specifically protecting each critical component of the SCADA/PCS network. The concept is akin to using firewalls to protect hosts instead of only network borders [3], and is specially useful for critical information infrastructures given their complexity and criticality, with many routes into the control network that can not be easily closed (e.g., Internet, dial-up modems, VPNs, wireless access points).

Application-level semantics. Critical infrastructures have many legacy components that were designed without security in mind, thus do not employ security mechanisms like access control and cryptography. Since these security mechanisms are not part of the SCADA/PCS protocols and systems, which must still be protected, protection mechanisms (such as access control) must be deployed in some point between the infrastructure and the hosts that access it. The CIS has to inspect and evaluate the messages considering application-level semantics because, as already said, the application (infrastructure) itself does not verify it.

Rich access control model. Besides the capacity to inspect application-level messages, the CIS needs to support a rich access control policy that takes into account the multi-organizational nature of the critical infrastructures as well as their different operational states. Taking the power grid as an example, there are several companies involved in generation, transmission and distribution of energy, as well as regulatory agencies, and several of these parties can execute operations in the infrastructure. Moreover, almost all operations are based on a classical state model of the grid [4]. In each state of this model, specific actions must be taken (e.g., actions defined in a defense plan, to avoid or recover from a power outage) and many of these actions are not allowed in other states (e.g., a generator can not be separated automatically when the grid is in its normal state). These two complex facets of access control in critical infrastructures require more elaborated models than basic mandatory, discretionary or role-based access control. To deal with this, in the architecture of CRUTIAL we adopt a more elaborated model, PolyOrBAC (Organization Based Access Control) [1]. It allows the specification of security policies containing permissions, prohibitions, obligations and recommendations, taking into account the role of the subject, who is part of an organization, the action it wants to execute, the target object of this action, and the context in which it is executed. An example: "In context 'emergency', operators from company C can execute maintenance operations on device D."

Intrusion tolerance. The level of security of current systems connected to the Internet is not adequate for the infrastructures we are concerned with, given their criticality. To improve the security and dependability of the CIS, it is designed to be intrusion-tolerant, or Byzantine fault-tolerant: it is replicated in n machines and follows its specification as long as at most f of these machines are attacked and have their behavior

corrupted. This approach works if each system replica is sufficiently diverse to ensure that a similar attack will not affect more than f machines at the same time.

Self-Healing. To enhance the resilience of the CIS, we employ self-healing mechanisms through proactive and reactive recovery. Proactive recovery is useful to periodically rejuvenate the replicas, guaranteeing perpetual correct operation as long as no more than f replicas become faulty in a given recovery period [10]. Reactive recovery is a complement of proactive recovery that allows to deal with compromised replicas more quickly when attackers make conspicuous actions (e.g., a DoS attack) [9].

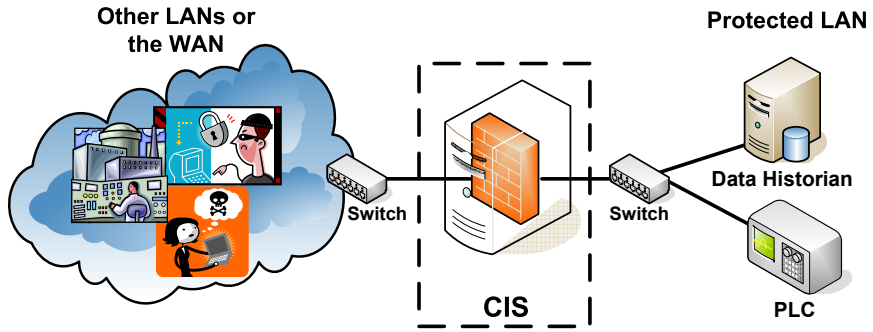
Intrusion tolerance and self-healing can be integrated in the design of the CIS in an incremental way. In the next section we describe three possible deployments of the CIS and their characteristics.

A Hierarchy of CIS Designs

Give the various criticality levels of the CI equipment and the cost of using a replicated device, it is worth defining a hierarchy of CISs designs incrementally more resilient. This section presents this hierarchy and, for each design, we point out what it offers, its cost and limitations.

Non-Intrusion-Tolerant CIS

Figure 2 depicts the design of a non-intrusion-tolerant CIS. The non-intrusion-tolerant CIS is the cheapest design because it only requires one machine just as any classical firewall. Nevertheless, it offers better protection than normal firewalls through the use of application-level policy enforcement and of a rich access control model. These characteristics reduce the probability of an attacker being able to construct a well-crafted message and deceive the CIS to let it go through. Although it is more difficult to mislead the CIS, an attacker may find a way of compromising the machine where the CIS is running (e.g., by exploiting a vulnerability of the operating system) and take control of CIS operation. Notice that the attacker may need days, weeks, or even months to find a vulnerability and deploy such an attack, but from what happened in the past we know that with a reasonable probability she will eventually have success in her quest.



Offers:
- Application-level policy enforcement
- Rich access control model

Requires:
- 1 machine

Limitations:
- Given sufficient time, an attacker may compromise the CIS machine and access the protected LAN

Figure 2: Non-intrusion-tolerant CIS design.

Intrusion-Tolerant CIS

To build a CIS that still works even if some of its components are intruded by a malicious attacker, we need to apply the intrusion tolerance paradigm [13]. To understand the *rationale of the design* of the intrusion-tolerant CIS, consider the problem of implementing a replicated firewall between a non-trusted WAN (or LAN) and the LAN that we want to protect. Further assume that we wish to ensure that only the correct messages (according to the deployed policy) go from the non-trusted side, through the CIS, to the computers and/or devices in the protected LAN. A first difficulty to address is that traffic has to be received by all n replicas, instead of only 1 (as in a normal firewall), to allow every replica to participate in the decisions. A second problem is that up to f replicas can be faulty and behave malicious, both towards the other replicas as to the receiver computers (e.g., the SCADA controllers).

Our solution to the first problem is to use some device (e.g., an Ethernet hub) to broadcast the traffic to all replicas. These verify whether the messages comply with the security policy and do a vote, approving a message if and only if at least $f+1$ different replicas are in favor. This guarantees that at least one correct replica thinks that the message should go through. An approved message is then transmitted by the CIS to the destination by a distinguished replica, the *leader*, so there is no unnecessary traffic multiplication inside the LAN.

Traditionally, intrusion-tolerant mechanisms address the second problem with masking protocols of the Byzantine type, which extract the correct result from the n replicas, despite f being malicious: basically, only results (or messages) that are supported by $f+1$ replicas are accepted. Since a result must be sent to the computers in the protected LAN, this consolidation has to be performed either at the source or at the destination. The simplest and most usual approach implements a front-end at the destination host that accepts a message if: (1) $f+1$ different replicas send it; or (2) the message has a certificate showing that $f+1$ replicas approve it; or (3) the message has a signature generated by $f+1$ replicas using threshold cryptography[1]. This however would require changes to the end hosts and the traffic in the protected LAN would be multiplied by n in certain cases (every replica would send the message), which is undesirable.

Therefore, we should turn ourselves to consolidating at the source and transmit *only one, but correct, message*. What is innovative here is that the source-consolidation mechanism should be transparent to the protected LAN. Moreover, this has to be done with the following difficulty in mind – since a faulty replica (leader or not) has direct access to the LAN, it can send incorrect traffic to the protected computers, which can not distinguish between good and bad messages. This makes consolidation at the source a hard problem.

According to best practice recommendations from expert organizations and governments, the standard protocols that are expected to be generalized in SCADA/PCS systems will utilize IPSEC to secure the communications. Consequently, we can assume that the IPSEC Authentication Header (AH) protocol [6] will be available, and take advantage of it to design our solutions. Even when IPSec is not being used, most current operating systems support it and allow using it in a transparent way, from the point of view of the application, e.g., of the SCADA/PCS software. The basic idea is that the protected computers will only accept messages with a valid IPSEC/AH Message Authentication Code (MAC), which can only be produced if the message is approved by $f+1$ different replicas. However, IPSEC/AH MACs are generated using a shared key[2] K and a hash function, so it is not possible to use threshold cryptography. As the attentive reader will note, the shared key storage becomes a vulnerability point that can not be overlooked in a highly resilient design – there must be some secure component that stores the shared key and produces MACs. This requirement calls for a secure component in an otherwise Byzantine-on-failure environment, which we will

[1] Threshold cryptography is a form of cryptographic that allows to split a key among a set of parties and to define the minimum number of parties that are required to make useful things with the key (see [8]). Unfortunately, it can only be used in combination with public-key cryptography schemes where different keys are used to encrypt and to decrypt, whereas IPSEC/AH uses the same key to produce and verify MACs.

[2] We assume that IPSEC/AH is used with manual key management.

name as a *secure wormhole* [12]. The wormhole can be deployed as a set of local trustworthy components, one per replica, using technologies such as smart-cards or cryptographic boards. Figure 3 depicts the design of an intrusion-tolerant CIS.

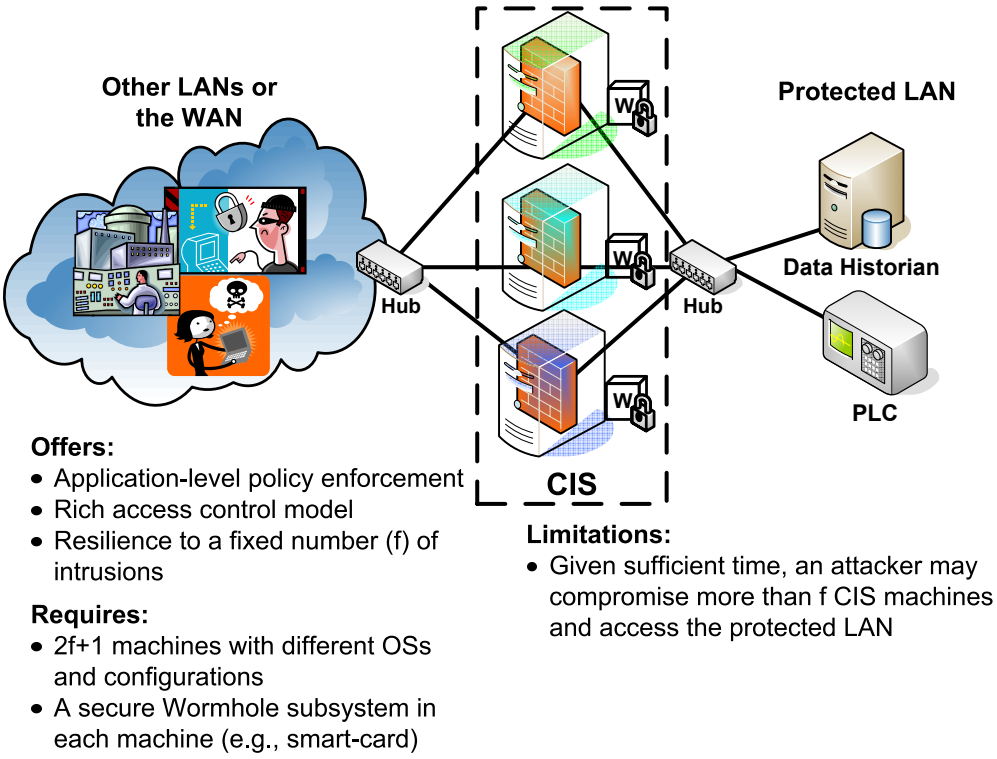


Figure 3: Intrusion-tolerant CIS design.

This CIS design requires $2f + 1$ machines in order to tolerate f intrusions. Thus, the configuration presented in Figure 3 is able to tolerate one intrusion. Notice that such a design only makes sense if the different machines can not be attacked in the same way, i.e., they must not share the same set of vulnerabilities. In order to achieve this goal, each CIS replica is deployed in a different operating system (e.g., Linux, FreeBSD, Solaris), and the operating systems are configured to use different passwords and different services. In addition, each CIS replica is enhanced with a secure wormhole subsystem that stores a secret key and this key is used to produce a MAC for messages that are approved by at least $f + 1$ replicas. Given that the wormhole is secure, no malicious replica can force it to sign an unapproved message.

In practical terms, the intrusion-tolerant CIS is more difficult to compromise than the non-intrusion-tolerant version because an attacker will need to find vulnerabilities and deploy attacks against $f + 1$ diverse replicas and not just one. The task of compromising each replica may take days, weeks, or months, but attackers tend to be patient.

Intrusion-Tolerant and Self-Healing CIS

The most resilient CIS design combines intrusion tolerance with self-healing mechanisms in order to address the limitations explained in the previous section. Self-healing is provided by a proactive-reactive recovery service that combines time-triggered periodic rejuvenations with event-triggered rejuvenations when something bad is detected or suspected [9].

Proactive recoveries are triggered periodically in every replica even if it is not compromised. The goal is to remove the effects of malicious attacks/faults even if the attacker remains dormant. Otherwise, if an

attacker compromises a replica and makes an action that can be detected (e.g., sending a message not signed with the shared key K), the compromised replica is rejuvenated through a reactive recovery. Moreover, the rejuvenation process of a (proactive or reactive) recovery does not simply restore replicas to known states, since this would allow an attacker to exploit the same vulnerabilities as before. The rejuvenation process itself introduces some degree of diversity to restored replicas (changes the operating system, uses memory obfuscation techniques, changes passwords, etc.), so that attackers will have to find other vulnerabilities in order to compromise a replica. Figure 4 depicts the design of an intrusion-tolerant CIS with self-healing mechanisms.

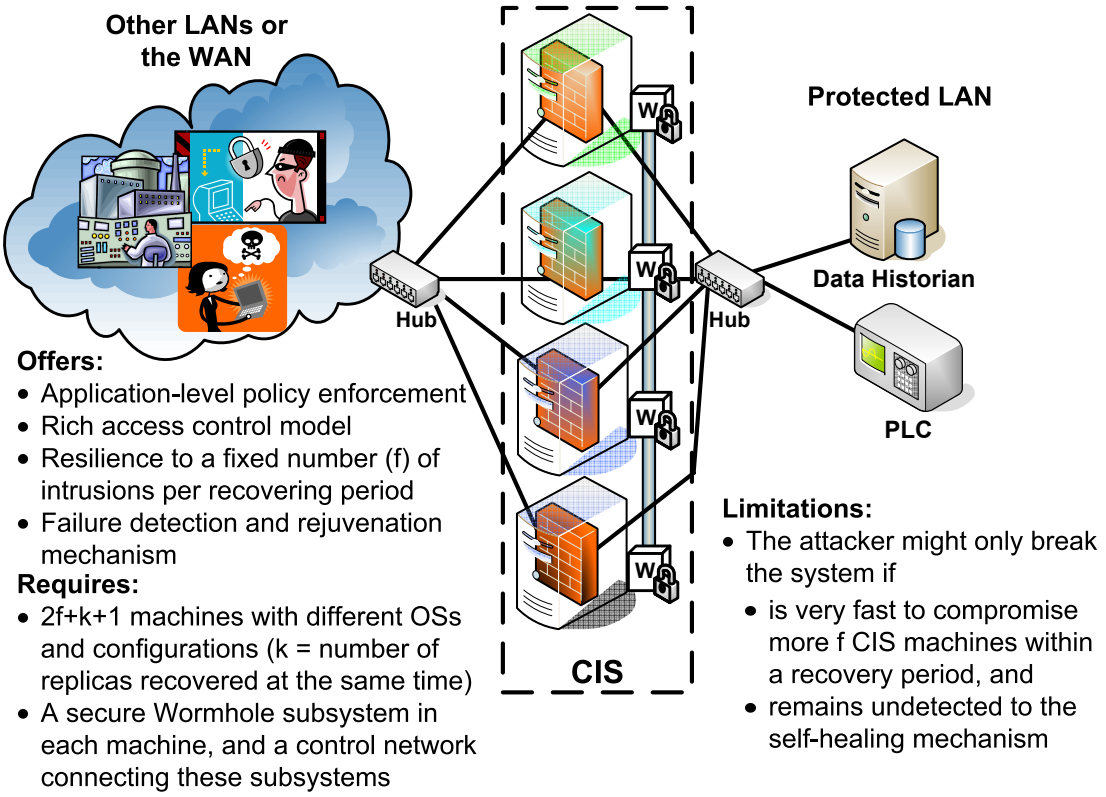


Figure 4: Intrusion-tolerant & self-healing CIS design.

This CIS design requires $2f+k+1$ machines in order to tolerate f intrusions per recovering period (dictated by the periodicity of the proactive recoveries). The new parameter k represents the number of replicas that recover at the same time and its value is typically one. If this parameter was not included in the calculation of the total number of required machines, the CIS could become unavailable during recoveries. Thus, the configuration presented in Figure 4 composed of 4 replicas is able to tolerate one intrusion while another is being rejuvenated. The length of the recovering period corresponds to the sum of the recovery time of each replica. In the experiments performed in our laboratory we were able to recovery each replica in less than 2.5 minutes, which means that the recovering period was less than 10 minutes when using 4 replicas [9]. In this scenario, an attacker would need to find vulnerabilities and deploy attacks against $f+1=2$ replicas within 10 minutes, which seems to be difficult given that each recovery changes the set of vulnerabilities and the attacker has to restart her work. Moreover, we believe that the recovery time of a replica can be reduced to less than one minute using readily available technologies such as solid state disks.

Deployment Space

The intrusion-tolerant CIS (with or without self-healing) presented in the previous sections requires several replicas to be deployed effectively. However, since price is always a major concern, a much more cost effective solution can be attained by resorting to *virtualization* [2]. The various replicas are deployed in the same host, using virtual machines (VM) to isolate the different runtime environments, preventing intrusions from propagating from one replica to the others. Figure 5 presents the main four design options.

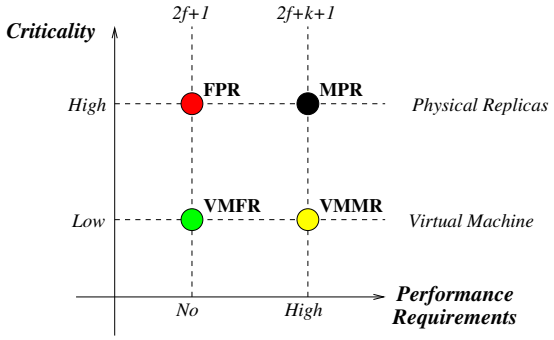


Figure 5: Deployment space for the intrusion-tolerant CIS.

In this figure it is possible to identify two main dimensions in the design space. If the LAN protected by the CIS provides a very critical service, the implementation should be based on the use of different physical machines for each CIS replica, allowing tolerance to physical and software faults. If the protected service is part of an application that requires certain performance guarantees, without tolerance to unexpected delays, there must be at least $2f+k+1$ replicas to ensure safe operation and availability even in case of f faults an k recovering replicas. On the other hand, if occasional delays are not a problem, $2f+1$ replicas suffice to maintain correct operation of the system. The four main options are the following:

- *VMFR (Virtual Machines with Few Replicas)*: $2f+1$ replicas are deployed as virtual machines running in the same host. The system does not tolerate physical faults or bugs in the virtual machine monitor, but offers some level of protection if a virtual machine is subject to an intrusion;
- *VMMR (Virtual Machines with More Replicas)*: There are $2f+k+1$ logical replicas running in different virtual machines. The tolerance to malicious behavior is the same of VMFR but now the system continues to verify and forward messages even with f faulty and k recovering replicas;
- *FPR (Few Physical Replicas)*: Similar to VMFR, but with different physical replicas. There is some tolerance to accidental hardware and software faults as well as intrusions;
- *MPR (More Physical Replicas)*: This is similar to FPR but with k more replicas, to allow self-healing to be used without endangering the availability of the service.

These deployments can be used in different points of the critical infrastructure to protect different applications. For example, a MPR deployment can be used to protect a critical power grid defense system (e.g., **an** Automatic Grid Separation System – that performs automatic grid separations in emergency states [5]) while a VMMR implementation can be adequate to protect a substation data historian network.

Conclusions

Critical infrastructure protection become an extremely important and fascinating problem. Not only these infrastructures are particularly complex, but they include legacy equipment that can not be modified, and their owners are more than reluctant in introducing security mechanisms that may prevent them from doing control operations when they need to.

This paper presents the CRUTIAL Information Switch, a device designed to protect critical infrastructures, tackling this uncommon set of challenges. The CIS is intrusion-tolerant and self-healing, meaning that it remains operating correctly even if there are intrusions in some of its components. It supports distributed security policies based on a sophisticated access control model, PolyOrBAC. The paper presents CIS variants that may be useful in different scenarios, with different levels of criticality: non-intrusion-tolerant CIS, intrusion-tolerant CIS, intrusion-tolerant+self-healing CIS. Furthermore, four configuration possibilities with different costs are discussed: with virtual or physical machines, and few or more replicas.

Acknowledgments

We warmly thank our CRUTIAL project partners for the interesting discussions about the architecture presented in this paper. This work was partially supported by the EC through project IST-2004-27513 (CRUTIAL).

References

[1] A. Abou El Kalam, Y. Deswarte, R. Baida, and M. Kaaniche. Access control for collaborative systems: A web services based approach. In *Proc. of the IEEE International Conference on Web Services*, pages 1064–1071, 2007.

[2] P. Barham, B. Dragovic, K. Fraiser, S. Hand, T. Harris, A. Ho, R. Neugebaurer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. of the 19th ACM Symposium on Operating Systems Principles*, Oct. 2003.

[3] S. M. Bellovin. Distributed firewalls. *;login:*, Nov. 1999.

[4] L. H. Fink and K. Carlsen. Operating under stress and strain. *IEEE Spectrum*, Mar. 1978.

[5] F. Garrone, C. Brasca, D. Cerotti, D. C. Raiteri, A. Daidone, G. Deconinck, S. Donatelli, G. Dondossola, F. Grandoni, M. Kaâniche, and T. Rigole. Analysis of new control applications. Project Deliverable D2, CRUTIAL EC project IST-2004-27513, Jan. 2007.

[6] S. Kent. IP Authentication Header. RFC 4302 (Proposed Standard), Dec. 2005.

[7] SANS Institute. CIA confirms cyber attack caused multi-city power outage. *SANS NewsBites*, 10(5), Jan. 2008. Available at `http://www.sans.org/newsletters/newsbites/newsbites.php?vol=10&issue=5`.

[8] F. B. Schneider and L. Zhou. Implementing trustworthy services using replicated state machines. *IEEE Security & Privacy*, 3(5):34–43, Sept. 2005.

[9] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo. Resilient intrusion tolerance through proactive and reactive recovery. In *Proc. of the 13th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 373–380, Dec. 2007.

[10] P. Sousa, N. F. Neves, and P. Verissimo. How resilient are distributed *f* fault/intrusion-tolerant systems? In *Proc. of the 35th International Conference on Dependable Systems and Networks*, pages 98–107, June 2005.

[11] K. Stouffer, J. Falco, and K. Kent. Guide to supervisory control and data acquisition (SCADA) and industrial control systems security. Recommendations of the National Institute of Standards and Technology. Special Publication 800-82, NIST, Sept. 2006. Initial Public Draft.

[12] P. Verissimo. Travelling through wormholes: a new look at distributed systems models. *ACM SIGACT News*, 37(1), 2006.

[13] P. Verissimo, N. F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, and I. Welch. Intrusion-tolerant middleware: The road to automatic security. *IEEE Security & Privacy*, 4(4):54–62, Jul./Aug. 2006.