On the Security of Industrial Control Systems

Defense Strategy for a Representative Testbed

João Almeida*, José Donato†
Departamento de Engenharia Informática
Universidade de Coimbra
*jlalmeida@student.dei.uc.pt, †donato@student.dei.uc.pt

***Abstract*—Industrial Control Systems for critical infrastructures face extremely sophisticated threats, for which a domain-specific defense strategy is needed. This involves an iterative procedure of deploying several layers of complementing security mechanisms, that are continuously monitored for their efficiency in mitigating existing threats, and shortcomings with new ones. In this paper we outline such a strategy for a representative testbed that was previously penetration-tested, with the goal of raising awareness to the main concerns one should have when implementing or iterating upon one.**

***Index Terms*—Industrial control, SCADA systems, Network security, Computer security**

I. Introduction

Modern Industrial Control Systems (ICS) face common ICT threats, due to their adoption of such technologies for increased connectivity, but also highly advanced threats from the criticality of systems they often control.

In this work we propose a defense strategy for the same system that we were previously attacking. This obviously goes against the basic principle of security by design, however this is a highly weak, vulnerable and outdated system, for which action must be taken as soon as possible to prevent many of its possible attacks - for example the over-spinning of its cooling motor.

Our goal is then to raise awareness for those responsible of similar ICS, to take action as soon as possible. As we know, in security, the approach of simply reacting to incidents is common (*"casa roubada, trancas à porta"*). This may still exist in ICS, due to a false sense of security from previous strategies of airgapping and obfuscation, but recent advances towards openness and connectivity have made this infeasible in every way, given the criticality of assets controlled and threats faced.

With this in mind, we tried to come up with a plan that would provide defenses to an existing industrial system without any basic ones set in place, and going off what are currently the best mechanisms available. It is important to remind that security should be a continuous process, of learning from past failures but also looking at the present/future for what could be preempted.

We based ourselves on Lockheed Martin's Cyber Kill Chain shown in Fig.1, which outlines the course of action an attacker must take to successfully carry out the objective:

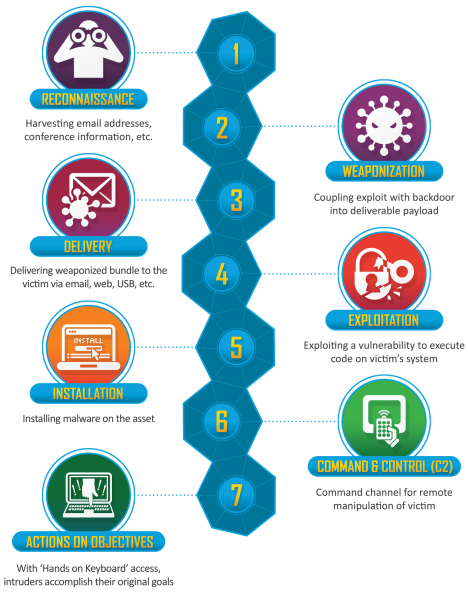

Fig. 1: Cyber Kill chain
Image taken from [3]

.

Our idea is to develop a plan where we cover and protect against all the 7 steps of the cyber attacker chain (Fig.1), since successfully blocking an attack at any one of them, stops it from carrying out the intended disruption - *breaks the chain* [6].

However, we need to be sure that we do not sacrifice the availability of the system with the introduction of performance-degrading security mechanisms, as ICS typically work under strict latency restrictions.

The paper is structured beginning with background knowledge on threats and controls (Sec.II), followed by a description of the testbed and its previously identified vulnerabilities (Sec.III). In IV we described the mechanisms we would deploy for each step of the kill chain. Following this, we focus specifically on what we think is a great addition to industrial systems: virtualization (and more specifically network virtualization with SDN) in section VI. Lastly, in section VII, we configure and evaluate the deployment of OSSEC as a host-based IDS, aiming to complement the tools already studied in the course.

II. Background Knowledge

A. Advanced Persistent Threat

> "Well-resourced and trained adversaries that conduct multi-year intrusion campaigns targeting highly sensitive economic, proprietary, ornational security information. These adversaries accomplish their goals using advanced tools and techniques designed to defeat most conventional computer network defense mechanisms." [4]

This definition is self-explanatory but this type of threat is important to us because they are common enemies of Industrial Control Systems. Because these systems provide critical services, they are the most common targets for cyberterrorism.

One APT involves months of work and combines all the seven steps of the Cyber Kill Chain (Fig.1).

Stuxnet is the perfect example of this [8]. The attacker spends weeks or even months in the first step of reconnaissance, where they tries to stealthily learn everything possible about the system concerned, in some cases through espionage (more on this in Sec.IV-A). The payload that's developed is then specific to the target, often exploiting zero-day vulnerabilities (previously unknown). This describes well how exhaustive this type of attacks is and, normally, with the people behind it behind very well-resourced, eg. governments.

Unlike attacks like ransomware that try to reach everyone, an APT is targeted/focused on a company/government.

B. Security Control

It's "all about defending important assets" [PL1]. Field network devices in ICS like PLCs are responsible for the operation of often critical infrastructures, for which any form of malfunction may risk extreme financial repercussions, as well as the health and safety of some [2].

Alas, some of these assets rely on extremely insecure protocols that have become industry standard, namely ModBus (cleartext communications with no authentication). Morever, the introduction of ICT technology, promoting the benefits of interconnectivity, has left them extremely exposed.

As such, we need to implement a myriad of security mechanisms to reduce (control) the risk of these threats materializing. The common approach of simply reacting post-compromise would be disastrous in these systems [4]. There will obviously be associated costs, for some quite high, but we need to consider the cost induced by a successful attack on this type of infrastructure, which can be devastating.

These controls can follow several courses of action - "detect, deny, disrupt, degrade, deceive, and destroy" [4]. That is, not all need to deny attacks, some could just serve to alert of its (ideally ongoing) occurrence, or others to mislead the attacker (eg. honeypots).

When we consider an APT, we need to be aware that they have the ability to bypass most mechanisms [6]. We will need some to be tailor made for the systems we're working with - ICS - and to follow a domain-specific approach when planning out [2]. Nevertheless this does not mean common mechanisms aren't extremely useful to stop the lower-skilled, but higher-numbered, attackers.

III. Testbed

A. Overview

Testbed models a very simple Industrial Control System in which we have the same constraint: high availability at the expense of security.

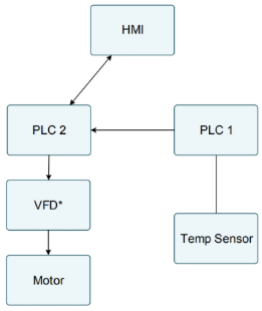

Fig. 2: Testbed topology.
Image taken from [CDIS-PL2].

We have an Oil Temperature sensor, whose measures are used by the PLC (located at .250) to adjust the cooling motor's frequency to achieve certain speeds (rpm). -

This whole process is supervised through the HMI (human-machine interface) located at .10.

From the previous assignment, we also know that a honeypot emulating an Siemens S7 model is deployed at .245.

After the first assignment, the deficiencies of these systems are obvious. Despite the existence of one honeypot (further disclosed by the teacher and effective because we wasted some time researching it), there are too many characteristics that makes industrial systems attractive to attackers.

In the next section, we will talk about the vulnerabilities that stand out (some we find in the first assignment, others are common in these systems).

B. Vulnerabilities

After we were inside the network, everything was possible: starting from bringing down critical and choke point devices like the PLC with simple DDoS attacks to exploiting HMIs using default credentials.

Below there is a list with some of the vulnerabilities we found concerning both PLC and HMIs (the most concerning devices in the system). These have been assigned unique identifiers, to then be associated with the controls described in the strategy outline (Sec.IV).

- **V1 (Exposed Devices):** both but not only these devices are exposed to any scan when we are inside the network. Sometimes also outside the network, a quick search on shodan.io gives us more than 500 results containing the same PLC model exposed in the public internet:

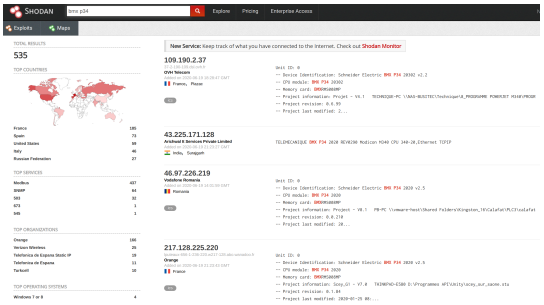

Fig. 3: PLC search in shodan.io

It is important to note that this make the profiling process way easier for the attacker.

- **V2 (Exposed Services):** In addition to these devices being completely exposed, crucial services are also open to anyone such as FTP and web server.
- **V3 (Plaintext Communications):** Even though we have not discovered the entire system topology so far, the fact that devices communicate in plaintext without any authentication further facilitates this process. Sniffing the network, an attacker can easily understand the relationships between the devices and how the system works.
- **V4 (Unauthenticated Communications):** Messages are not authencations. Furthermore, there are no protections regarding the ARP protocol. Caches can be easily poisoned in order to perform Man in the Middle attacks - these hijack communications, freely injecting or tampering with messages.
- **V5 (Fragile Devices):** The fact that these devices have low processing capacity, makes it impossible not only to support secure cryptographic algorithms but also vulnerable to denial of service attacks. We saw that with simple tools such as `nping` are capable enough to make the PLC unavailable as long as the attacker wants.
- **V6 (Default Crendentials):** Normally, services the open services such as FTP or web servers have some kind of authentication (most frequently username+password). This brings a false sense of security since in most cases default credentials easily found in internet (for example in the product's documentation) are used.
- **V7 (Legacy Systems):** The majority of these systems are legacy and vulnerable. For the HMI that uses some version of windows we found vulnerabilities related to SMB (eternal blue). Regarding the PLC, after we found out what kind of linux version we used we also discovered several possible exploits (https://www.exploit-db.com/exploits/40847). This is a major problem since updating these devices require vendor certification and can potentially break them, meanwhile replacing requires money and stopping the industrial process (and stopping production means losses).
- **V8 (System Integrity):** There are not any type of file integrity checking in any device. This makes it possible for an attacker to reprogram the functionality of the devices without being detected (we suggested this in the first assignment when proposed to change the web server in the PLC to contain malicious code and infect the employees).

As if these vulnerabilities were not enough, we still think the following ones are common in these type of systems [9] and therefore we need to fix them as well:

- **V9 (No Virtualization):** There is not any kind of segmentation in the network. This means that the industrial critical process is not separated from the rest at least. If an attacker gets access to one area of the network, can communicate without a problem with all the rest (if the devices are exposed).
-
- **V10 (No Monitoring):** Lack of network audit and monitoring. All actions (and most importantly the suspicious ones) must be logged for further analysis.
-
- **V11 (Lacking Access Control):** Passwords aren't enough... User permissions and access controls are not well defined nor enforced and logged.

IV. Strategy

A single defense layer is often described with the *castle and moat* analogy [5] - where it's considered that all intruders would have been barred entrance from the sole existing gate, and as such no further checks are carried out once inside (eg. only a firewall as defense mechanism).

However, as we know, this is not feasible in computer security. The often huge attack surfaces mean there are several such *gates* we have to defend, and the fact that attacks are ever-changing means that a single check cannot hope to detect and block all malicious attempts.

What we need are several layers of *walls* covering the whole attack surface, where each successive *gate* is tailored to stopping what would be the attacker's next step. Furthermore, a large percentage of attackers are insiders, and so perimeter defense is not enough [2].

Lockheed Martin's Cyber Kill Chain [3] models such steps that an Advance Persistent Threat carries out, helping us understand the behaviours we're going up against.

Our goal is to maximize our chances of intercepting intrusion attempts [5], blocking attacks as soon as possible in the killchain, while being *insured* by the controls implemented for further steps. Also, we want to be able to learn everything we can about the attacker and then further improve the security of the system (this is possible with honeypots explained later in subsection IV-A).

Nonetheless, persistent attackers will eventually find a way through [2].

If the attack can't be blocked, it should at least be possible to detect and delay/contain it long enough for the incident response team to benefit from the insight obtained.

Some controls are applicable with small specificities to several phases, as is the case of virtualization. This one being so important, will be thoroughly discussed in Sec.VI.

A. Reconnaissance

During the recon phase, the attacker attempts to map out the system's attack surface, namely exposed hosts and their services, with network exploration tools.

This is a mostly passive endeavour, in order to find out a target for which to develop the exploiting payload (cf. Weaponization). That is, for which we suspect exists an exploitable vulnerability.

Typical substeps for this stage include:

1) host discovery: could just be randomly scanning the internet, or identifying an organization's IP addresses from DNS records and other public sources. and finding which are responsive.
2) target profilling: port scanning for runnign services which we can fingerprint based on their responses
3) vulnerability search: search repositories that compile vulnerabilities (many eg. cve-search.org, cve.mitre.org, etc). ICS are typically made up of legacy systems, that tend not to be updated for fear of disturbing continuity of service. As such many should be running vulnerable software (including the ones in our testbed III).

Nevertheless, with these tools it may also be possible to straightaway carry out denial of services attacks to exposed devices. This is a great threat if these consist of field network devices like PLCs, with limited computational power and whose unavailability will have severe impact in the whole system.

1) ***Firewall****:* communications with the outside, if necessary, should be completely mediated by a packet filtering router and only be allowed for selected devices - namely those in the enterprise network. One thing's for sure: we do not want our PLCs showing up in Shodan or similar search engines for exposed devices.

For Linux systems one can use IPTables, where simple rules are defined that will match data in the packet headers. It can also keep track of connections with state, through the use of modules, and can integrate with Intrusion Detection Systems (IDS).

2) ***VPN****:* access to field network devices should be controlled (not direct) and authenticated. Also, access control is important (more on this later in Subsection IV-C3).

3) ***NIDS****:* in order to elicit *fingerprintable* responses, tools like Nmap often send malformed packets. There's often no need to support these in normal operation, and furthermore they may break the fragile TCP/IP stack implementations of many field network devices. These tools, like Snort, should be deployed so as not to increase latency, ie. in passive mode and through a monitor port so as to avoid a single point-of-failure for the network's traffic. This can be deployed with a managed switch or a TAP.

4) ***Honeypots****:* we want these to appear as if they were legitimate targets for the attacker, for example by exposing ports of services with similar fingerprints to those found in some PLC/HMI models. Also, they should be integrated with other components, emulating communications, to make them credible.

The goal is then for subsequent steps to be carried out against them instead of the real devices essential to the infrastructure continuity of operation. Meanwhile an incident response team has been made aware of this occurrence, which signifies an intrusion attempt as these devices do not serve any purpose for the actual service provided (no *actual* PLC,etc is expected to communicate with it).

From the attacker's perspective, it's a tradeoff between how much we interact with the device to determine whether it's an honeypot or not.

Conpot lets us deploy extensible ICS/SCADA honeypots [13]. These can be setup to act as close as possible to realistic devices, including the introduction of artificial delays in response times [13]. Together, these emulate complex infrastructures.

Through logs we are then able to reconstruct an attacker's action, to try to derive their skill level and intents. Learning from the attackers makes it possible not only to react to the attack but to improve or develop new defense mechanisms.

Software-Defined Networking can also be used to implement Honeypots. This will be further explained in section VI.

5) ***Tar Pit****:* slow down requests until connection is authenticated. One possible way is to increase the response time exponencial until the other party is authenticated - "takes over unused IP addresses and answers to connection attempts" [CDIS-T4].

B. Weaponization

If successful in identifying a vulnerability, the attacker (if an APT) then develops the payload with which it to carry it out the attempt to compromise some system resource.

Though for less sophisticated attacks, common automated tools will most likely be employed.

If through a soft target, it will not be immediate.

1) ***Secure Services****:* use of secure versions of common services like remote access - eg. `ssh` instead of `telnet`. Legacy services often lacked any security consideration must be ditched for security-focused ones.

2) ***Patched Services****:* even still, vulnerabilities are inevitable in minimally complex software. Furthermore, being secure at a certain point in time, does not make it a given that it will remain as such in the future, since new zero-day exploits are constantly discovered, some exploiting what were not considered as vulnerabilities beforehand. Therefore, we must consider services that are actively supported with patches, and strive to keep them updated as much as possible - unfortunately compatibility issues may arise. In those situations, we need to analyse the risk trade-off and dependent on the calculated risk *vs* cost of change, keep the software out of date or not.

3) ***Threat Intelligence****:* a risk management approach should be adopted, keeping track of the ever-changing threat landscape. APTs may be highly innovative and difficult to predict, but less sophisticated attacks are often trend-based, attempting to replicate previous successes.

*4) **Security Awareness** [5]:* soft targets like employees or subcontractors should made aware of existing threats and follow an organization security policy. Security workshops are mandatory to ensure that everyone in the company is aware of the security problems that exist in these environments.

C. Delivery

In principle, if during recon the attacker was able to identify a given vulnerability, it's because they had to the specific service in that system. The main challenge is then to transfer it while remaining undetected.

Must consider soft targets which are often favoured by attackers due to their easier exploitability compared to 'busting through the front door'. This is the main reason why Security Awareness talked in Subsection IV-B4 is important.

*1) **Airgapping**:* wherein the system is isolated from any outside source, would certainly see an appetizing and seemingly definitive measure. Nevertheless this is extremely impractical - functionalities like remote access are desirable for maintenance - and would just lead to a false sense of security, since the system could still be compromised through the aforementioned soft targets (that end up physically delivering the payload).

*2) **Firewall**:* , application-level firewalls would be desired for exposed services, namely the web servers which were identified in the testbed. For these we could implement web application firewalls in a separate machine operating as a reverse proxy.

*3) **Access Control**:* communications should be authenticated, as well as logged - should not rule out the possibility of an insider being the attacker.

*4) **NIDS**:* an IDS like Snort with signatures for attacks could block common (important caveat) attacks

*5) **Unique credentials**:* - change all default credentials as these are commonly compiled into dictionaries used for automatically bruteforcing authentication. With remote access the attacker is free to deliver a generic payload. Passwords should be strong and unique to each device, so as to make it difficult to compromise each one. Furthermore, failed login attempts should be monitored and throttled.

*6) **USB Drives**:* are a common attack vector, where those owned by employees and/or subcontractors are infected prior to being used inside the infrastructure's network.

*7) **Spam filters**:* like Apache SpamAssassin, should be deployed for the company's email server. Targetted messages aiming to compromise the employees' machines are common (spearfishing), and aim to use them as a launching point for the attack (thus bypassing the firewall).

*8) **Security Awareness**:* to complement these last two controls, including training and email testing for employees [6].

D. Exploitation

*1) **Attack surface**:* should as small as possible, namely by only exposing the absolutely necessary services. With adequate access control in place, one could have the PLC/HMI web servers running on localhost only.

*2) **Vulnerability Management**:* all systems should be continuously assessed with vulnerability scanners. OpenVAS is an example of such a tool, and is based on a database of NVTs (Network Vulnerability Tests) to assess their existence. Scan tasks can be scheduled to periodically run. Reports should be analysed to dismiss false positives, and for the remaining, controls should be selected based on the associated risk.

*3) **Cooperation among Organizations**:* by sharing knowledge learned in the post-incident analysis, is essential so that similar attack vectors aren't consistently reapplied to new targets. Learning the exploit's procedure from system logs is one such example, though some companies may be reluctant to share possibly sensitive information.

E. Installation

The attacker attempts to install a persistent backdoor [6] for remote access, and from which to exfiltrate data for further recon.

*1) **Host-based IDS**:* like OSSEC, whose deployment is described in Sec.VII, can run integrity checks on key files to verify they remain unchanged, detect rootkits, and allow communications with only a select few IPs - which for field network devices should consist solely of local addresses.

If malware is installed to be available through the exposed web servers, that would be a watering hole attack [6] (we'd wait for a legitimate user to go drink from it).

*2) **Antivirus**:* to detect rootkits or other types of malware. It is importante to note that they can only be installed in certain devices because of the latency contraints (devices may not be powerful enough to support this type of software).

Another alternative is application whitelisting, i.e., only allow installations of applications that are whitelisted.

*3) **Least Privilege**:* for services running, especially minimize those running with administrator privileges [6].

*4) **Certificate validation before installations** [6]:* if an attacker is able to poison the DNS cache, packages or updates would be downloaded from malicious sources.

F. Command & Control

From the recon step the attacker will most likely only have a limited view of the system. In this step, it then maps it out with the newly obtained inside access, reporting back gathered intel so as to better coordinate the final attack - the one meant to seriously disrupt the infrastructure.

*1) **Monitoring Traffic**:* log packets, and be particularly wary of outgoing traffic. Much of this step involves exfiltrating the intel gathered back to the attacker, so as to better coordinate the final strike. A common vector for this is DNS tunnelling, as this protocol is often not considered in firewalls. In the field network, communications should be quite monotonous and easy to implement rules for what is their normal behaviour. However, we need to care for false positives, as dropping legitimate traffic could have serious consequences - need to run in passive mode.

*2) **Bumps-in-the-wire**:* to encrypt/authenticate traffic so communications are imperceptible to an attacker. Even though these are legacy devices that may not support it due to protocol/computational restrictions, ad-hoc devices named bump-in-the-wire exist. These can be used to implement IPSec on behalf of hosts [15] (Fig.4) - which encrypts and authenticates traffic at the network/IP level, transparently and tranversally to applications. Though we need to be careful with the latency introduced.

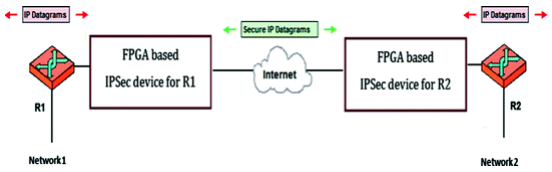

Fig. 4: Bump-in-the-wire deployment
Image taken from [16]
.

*3) **Honeypots**:* are a great tool once again, to fool an attacker with simulated HMI - PLC communications.

*4) **Virtualization**:* to contain potentially compromised devices, network should be segmented, ideally with virtualization as described in Sec.VI. This is because the initially compromised device could be in the business network, for which it'll still have to propagate itself to field devices where the real damage is done to the infrastructure.

G. Actions on Objectives

This is the step where the original goal of the APT is carried out. What ensues depends on the motivation [5], if terrorism then the goal is to disrupt or destroy the system, if financial the goal is to ask for ransom (with the promise of returning sensitive data or getting the process back up and running).

The advantage of our security approach according to the kill chain, is that the attacker has to have been successful in all 6 previous steps, meanwhile we only need to have blocked it at one [6].

*1) **Shadow Security Unit**:* Device placed in parallel with PLC with access to its state and interactions. Therefore, it has access to all the traffic with TAP, knows the exact values of the registers and is able to detect unusual behaviours. The main goal is to detect problems even when the reports to HMI are altered.

*2) **NIDS**:* detect issuing of anomalous commands through SCADA specific rules - for Snort we have (https://github.com/digitalbond/Quickdraw-Snort).

*3) **SIEM**:* use snort and tripwire outputs to feed a SIEM capable to react to attacks in real time. However, this step is hard because one bad choice of the SIEM tool can compromise the availability of the entire system.

*4) **Incident Response Team and Plan**:* if everything else fails and the attack is successful, a team must be in place to contain the impact it as soon as possible.

*5) **Secure Logging**:* to be able to reconstruct steps undertook by the attacker, and so strive to prevent it from happening again.

*6) **Redundancy**:* of PLC to implement voting algorithms, where the majority decides the correct command issued. Thus, if an attacker compromises one to send malicious commands, it will not impact as long as the remaining continue to function normally. It will also signify a failure that can then be investigated to assess whether it was some hardware fault, or rather a malicious attack (and fix later). However, the costs associated with this control can make it impossible to implement. A possible solution would be to virtualize the PLCs despite its possible performance disadvantages.

H. Testbed

We can now map these controls, into the testbed's vulnerabilities previously enumerated in Sec.III-B.

Even though all of those were applicable to our system, we will take a risk management approach, in that only those whose cost outweighs the risk are selected. For this, we must consider the impact and probability of the vulnerabilities being exploited, and the nature of threats we would be facing - quite simple system so unlikely to be the target of any persistent threat (at most attempts to compromise devices to take part in a botnet).

Vulnerability	Step	Control	Comments
V1 (Exposed Devices)	1	Firewall	Sec.IV-A1
V2 (Exposed Services)	2	Secure Services	Sec.IV-B1
V2 (Exposed Services)	2	Patched Services	Sec.IV-B2
V1/V2	2	Network Segmentation	Sec.VI (through virtualization)
V3 (Plaintext Communications)	6	Bump-in-the-wire	Sec.IV-F2
V4 (Unauth'ed Communications)	6	Bump-in-the-wire	Sec.IV-F2
V5 (Fragile Devices)	6	Hardware Virtualization	Sec.VI
V5 (Fragile Devices)	1	Honeypots	Sec.IV-A4 (applicable to other steps) decoys, give time to react
V6 (Default Crendentials)	3	Unique credentials	Sec-IV-C5
V7 (Legacy Systems)	4	Vulnerability Management	Sec.IV-D2
V8 (System Integrity)	5	HIDS	Sec.IV-E1
V9 (No Virtualization)			Sec.VI
V10 (No Monitoring)	6	Monitoring	Sec.IV-F1
V10 (No Monitoring)	7	SIEM	Sec.IV-G3
V11 (Lacking Access Control)	3	Access Control	Sec.IV-C3

TABLE I: Mapping of testbed vulnerabilities to controls

V. Additional Remarks

Before concluding this section, it's important to reiterate that once controls have been selected and deployed, they should not be set in stone.

Their efficiency in stopping attempts should be continuously measured, ideally not relying only on actual attacks, but also on penetration testing and red team exercises performed by outsiders.

These either serve to increase our confidence that everything is well setup, or better yet (and more likely) to identify holes in our defense.

*1) **Post-Analysis**:* is what allows us to learn from previous mistakes. By understanding what went wrong, we can devise a plan to rectify it, once again highlighting the need for reiterating the defense strategy.

Through all the logging mechanisms that should be in place, we can then gather:

- a recreation of the **steps** the attacker went through, so we can better model them. For this we resort to the analysis of logs and packets.
- identify the (soft) targets and their exploited **vulnerabilities**, to understand how they were able to execute one or more of the killchain's steps.
- from their targets we can also try to extrapolate their **motivations** and what in the system they consider as valuable. If the attack was unsuccessful, it is important to understand what the impact would have been.
- through **forensics** we could try to identify the attackers, for example by discovering the C2 infrastructure through malware analysis [6]
- damage assessment, including checking that no device remains compromised

With this improvement process, the goal is for attacks to be stopped increasingly earlier in the killchain over time [7].

*2) **OODA Loop**:* helps us take the best decisions possible, based on what we learned from post-analysis.

It's an acronym of the following steps, executed in a loop:

- **Observe** for information to help us make decisions (particularly the outcomes of previous decisions)
- **Orient** - process this information
- **Decide** on the best option available, but be open to adjust if it proves to not have been the best solution
- **Act** and monitor results to learn from them (do not repeat mistakes)

*3) **Risk Management**:* - a risk management process is essential to identify threats to assets, and the costs the organization may face if compromised. This can be used to motivate corporate leadership to invest in security, which is a common challenge.

*4) **Knowledge Sharing**:* among organizations, of attacks they've faced and conclusions that were taken.

VI. Virtualization

Virtualization is becoming a standard when developing big systems and it would fit perfectly in our plan. We did not include it in the steps because it would require lot of time and money to make it possible in our testbed. Nevertheless, it is important to explain this scenario.

It is important to separate the three security zones of the company each one with different strategies and monitored perimeters:

- IT department
- Operations
- Field

With virtual LANs (VLANs) we can easily implement such scenario, each zone in a different VLANs resulting in a safer and segmented network.

When is needed to communications between zones traffic must pass through firewall and IDS/IPS. An easy scenario of communication between zones is, for example, the communications between the HMI (Operations zone) and the PLC/Sensors (Field zone). This traffic must be monitored to prevent MiTM attacks presented in Subsection III-B, for example.

If the traffic is not whitelisted must set off alarms and wait for security incidents response team. Note that this is not critical since if the communication is not whitelisted it is because the communication is not strictly necessary for the industrial process so it does not jeopardize the availability of the system.

Furthermore, by default field devices are highly interdependent, where failure in one has a cascade effect. This multiplies the attack's potential impact and makes it harder to repair it back up.

Stuxnet attack destroyed one-fifth of Iranian centrifuges [14]. With virtualization and the right configurations the virus could not propagate that fast. Network segmentation with virtual LAN could easily have prevented this.

With this example it is easy to understand that if an attacker gains access to one area of the network, without network segmentation the cascade effect can be catastrophic.

Also, we need to talk about virtualizing the network with Software-Defined Networking (SDN).

With SDN we can implement data diodes [17], i.e., have all the communications only in one direction. Without bidirectional communications, lot of attacks would be prevented.

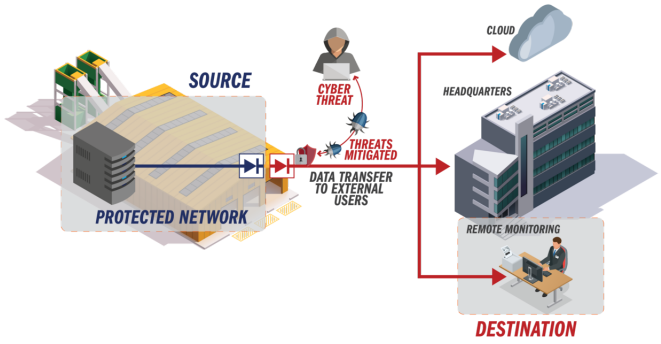

Fig. 5: Data Diode example. Image taken from [18]

In fact, to be compliant with Evaluation Assurance Level 7, the highest Common Criteria level, implementing data-diodes is strictly necessary.

Another perfect use case for SDN are honeypots. When the traffic reaches SDN routers if it is not whitelisted (therefore suspicious) can be immediately redirected to a honeypot (talked in more detail in Subsection IV-A4). The image below explains this:

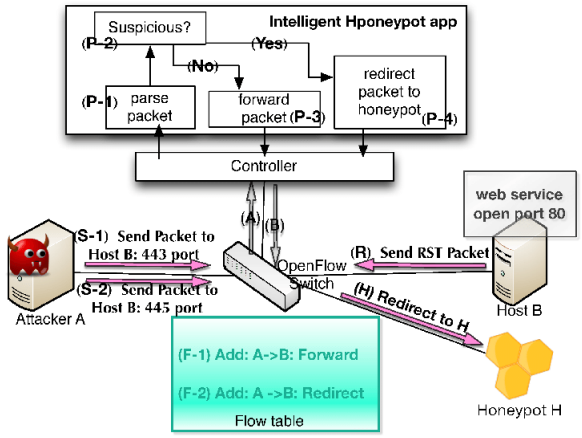

Fig. 6: Honeypot example. Image taken from [19]

Although it is not directly related to the assignment, the advantages of virtualization are not only security-related. If the company grows and there is a higher demand with virtualization is as easy as setting up a new virtual machine or more resources to the ones that already exist. If we need more HMIs because there are more employees we can just clone the virtual machines containing the HMIs. When thinking about the low level devices of the process (PLC and Sensors), the task is more difficult.

VII. OSSEC

A. Configuration

Since we already used both Snort and Honeycon in course classes, we tried to choose a tool that would integrate well with them and help us achieve the steps explained in the previous sections.

After some research, we conclude that OSSEC [10] and tripwire [11] would be great tools (in addition to honeycon and snort) to implement the controls talked so far. "OSSEC is an Open Source Host-based Intrusion Detection System that performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response" [10].

In OSSEC can configure multiple rules, for example, we can implement file integrity rules to check if any file inside /var/www was changed and prevent watering hole attacks:

```
<syscheck>
  <!-- Frequency that syscheck is executed - checking every 2 hours -->
  <frequency>7200</frequency>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories report_changes="yes" realtime="yes" check_all="yes">/etc,/usr/bin,/usr/sbin</
directories>
  <directories report_changes="yes" realtime="yes" check_all="yes">/var/www,/bin,/sbin</dire
ctories>
```

Fig. 7: OSSEC rules to report changes

Since APTs take normally months and this attack is complex, checking every two hours for integrity changes should be enough to detect and stop this attack on time.

```
<rule id="595" level="7" overwrite="yes">
    <category>ossec</category>
    <decoded_as>syscheck_new_entry</decoded_as>
    <description>critical files were changed</description>
    <group>syscheck,</group>
</rule>
```

Fig. 8: OSSEC file integrity rule

.

Tripwire is "a file integrity checker for UNIX systems" [11]. Therefore, tripwire is a simple but powerful tool. We can configure it to detect changes in any folder in the system:

```
(
        rulename = "Web server changes",
        severity= $(SIG_HI)
)
{
        /var/www        -> $(SEC_CRIT);
}
```

Fig. 9: Configurations to detect web server changes

Tripwire is not automatic. Therefore, we can set up a cronjob to run every two hours to detect file changes.

```
sudo crontab -e -u root
0 */2 * * * tripwire --check --email-report #add
    this line to the file
```

Also, we can get email notifications with the reports to the security team with:

```
tripwire --test --email security@team.com
emailto = security@team.com # add this in twpol.txt
    config file
sudo twadmin -m P /etc/tripwire/twpol.txt
sudo tripwire --init
```

SpamAssassin [12] would be a great addition to the system since it would reduce the likelihood of the employees getting attacked with phishing IV-C7.

B. Deployment

Of course we installed OSSEC locally on our machine in the system but in reality we would choose "server" to analyse all critical devices in the network at the same time: PLC and HMI.

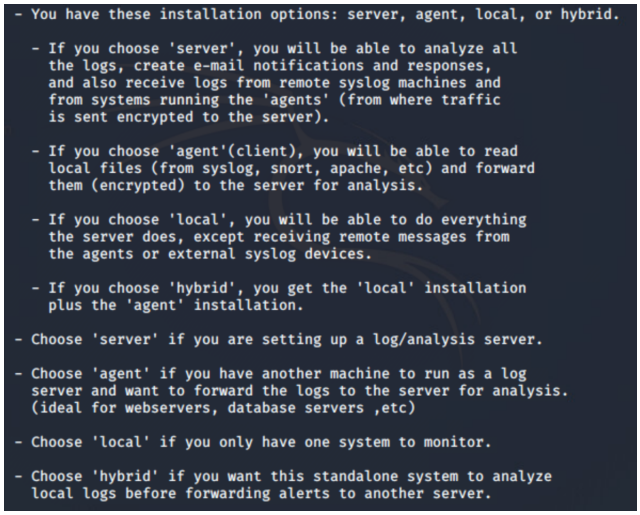

Fig. 10: OSSEC installation options

.

After OSSEC is installed and the configurations from previous section are done, we just need to start it:

```
/var/ossec/bin/ossec-control start
```

To install *Tripwire*, the following commands were used:

```
sudo apt install tripwire
changes in twpol.txt
sudo twadmin -m P /etc/tripwire/twpol.txt
sudo tripwire --init
sudo tripwire --check
```

C. Evaluation

OSSEC uses signatures to detect rootkits which makes it efficient to report them in real time. Therefore, if a rootkit or other similar malware were installed in the devices, OSSEC would promptly detect them and alert the response team. We did not try this because we were worried about the negative effects of installing a rootkit in the machine hosted in the course hypervisor. [20] is a good resource to check this in action.

However, we can see a few useful alerts that OSSEC provides, for example, when users try to perform root operations:

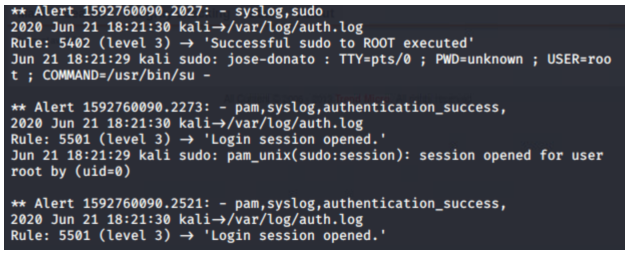

Fig. 11: OSSEC output

To evaluate *tripwire* the process was straightforward and simple. We tried to make changes to the folders that *tripwire* is configured to keep track of. In our case, the obvious use case is the web server folder to prevent watering hole attacks talked before:

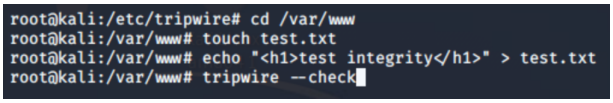

Fig. 12: Tripwire: Changes to evaluate configurations

Lastly, checking *tripwire*'s report we can see that the changes were reported and need further analysis to determine whether they are legitimate or not:

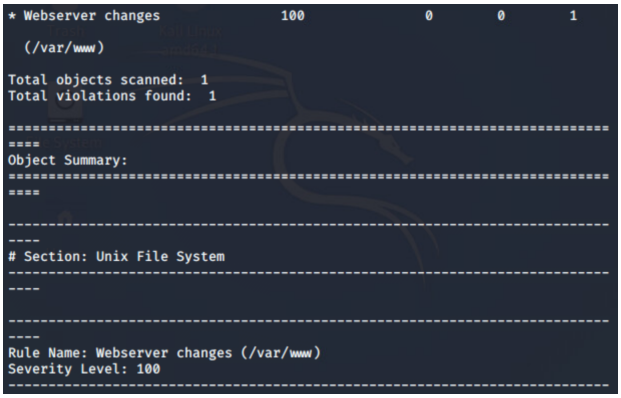

Fig. 13: Tripwire: Changes detected

VIII. Conclusion

In this paper we laid out a security strategy to cover all of the exhaustive steps that an Advanced Persistent Threat follows - the most common and complex attacks against Industrial Control Systems.

Its defense in depth approach maximizes the chances of detecting and stopping an attack, and even though it should ideally be applied at the beginning of the system's design (where there's increased freedom in what/how to implement), it's *never too late*, as security will always remain an iterative process of fine-tuning existing mechanisms and reacting to increasingly complex threats.

References

[1] Rosa, Luís, et al. "Attacking SCADA systems: A practical perspective." 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2017.

[2] Maglaras, L. A., Kim, K. H., Janicke, H., Ferrag, M. A., Rallis, S., Fragkou, P., ... & Cruz, T. J. (2018). Cyber security of critical infrastructures. Ict Express, 4(1), 42-45.

[3] Cyber Kill Chain® | Lockheed Martin https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[4] Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Leading Issues in Information Warfare & Security Research, 1(1), 80.

[5] Lance Spitzner. Applying Security Awareness to the Cyber Kill Chain | SANS Security Awareness https://www.sans.org/security-awareness-training/blog/applying-security-awareness-cyber-kill-chain

[6] Lockheed Martin. Applying Cyber Kill Chain® Methodology https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf
[7] Lockheed Martin. Seven Ways to Apply the Cyber Kill Chain® with a Threat Intelligence Platform https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Seven_Ways_to_Apply_the_Cyber_Kill_Chain_with_a_Threat_Intelligence_Platform.pdf
[8] Assante, Michael J., and Robert M. Lee. "The industrial control system cyber kill chain." SANS Institute InfoSec Reading Room 1 (2015).
[9] Nelson, Trent, and May Chaffin. "Common cybersecurity vulnerabilities in industrial control systems." Control systems security program (2011).
[10] OSSEC. https://github.com/ossec/ossec-hids
[11] tripwire(8) - Linux man page https://linux.die.net/man/8/tripwire
[12] Apache SpamAssassin: Welcome https://spamassassin.apache.org
[13] Conpot. http://conpot.org/
[14] Throwback Thursday: Whatever happened to Stuxnet? — Synopsys https://www.synopsys.com/blogs/software-security/whatever-happened-to-stuxnet/
[15] Keromytis, A. D., & Wright, J. L. (2000, June). Transparent Network Security Policy Enforcement. In USENIX Annual Technical Conference, FREENIX Track (pp. 215-226).
[16] Newe, T., Rao, M., Toal, D., Dooly, G., Omerdic, E., & Mathur, A. (2017). Efficient and High Speed FPGA Bump in the Wire Implementation for Data Integrity and Confidentiality Services in the IoT. In Sensors for Everyday Life (pp. 259-285). Springer, Cham.
[17] de Freitas, Miguel Borges, et al. "SDN-enabled virtual data diode." Computer Security. Springer, Cham, 2018. 102-118.
[18] Oregon Systems — Build the Best Technology with our Expertise https://www.oregon-systems.com/data.html
[19] Shin, Seungwon, et al. "Enhancing network security through software defined networking (SDN)." 2016 25th international conference on computer communication and networks (ICCCN). IEEE, 2016.
[20] OSSEC demo Rootkit detection - YouTube https://youtu.be/Hi22VaVFGbQ