Published: 05 October 2020

Citation in BibTeX format

ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security
October 5 - 9, 2020
Taipei, Taiwan

Conference Sponsors:
SIGSAC

RESEARCH-ARTICLE

I came, I saw, I hacked: Automated Generation of Process-independent Attacks for Industrial Control Systems

ESHA SARKAR, New York University, New York, NY, United States

HADJER BENKRAOUDA, NYU Abu Dhabi, Abu Dhabi, Abu Dhabi, United Arab Emirates

MICHAIL MANIATAKOS, NYU Abu Dhabi, Abu Dhabi, Abu Dhabi, United Arab Emirates

I came, I saw, I hacked: Automated Generation of Process-independent Attacks for Industrial Control Systems

Esha Sarkar
Tandon School of Engineering
New York University
esha.sarkar@nyu.edu

Hadjer Benkrouda
Center for Cyber Security
New York University Abu Dhabi
hadjer.benkraouda@nyu.edu

Michail Maniatakos
Center for Cyber Security
New York University Abu Dhabi
michail.maniatakos@nyu.edu

ABSTRACT

Malicious manipulations on Industrial Control Systems (ICSs) endanger critical infrastructures, causing unprecedented losses. State-of-the-art research in the discovery and exploitation of vulnerability typically assumes full visibility and control of the industrial process, which in real-world scenarios is unrealistic. In this work, we investigate the possibility of an automated end-to-end attack for an unknown control process in the constrained scenario of infecting just one industrial computer. We create databases of human-machine interface images, and Programmable Logic Controller (PLC) binaries using publicly available resources to train machine-learning models for modular and granular fingerprinting of the ICS sectors and the processes, respectively. We then explore control-theoretic attacks on the process leveraging common/ubiquitous control algorithm modules like Proportional Integral Derivative blocks using a PLC binary reverse-engineering tool, causing stable or oscillatory deviations within the operational limits of the plant. We package the automated attack and evaluate it against a benchmark chemical process, demonstrating the feasibility of advanced attacks even in constrained scenarios.

CCS CONCEPTS

• **Security and privacy** → **Systems security**.

KEYWORDS

Industrial control systems security, Machine learning, Fingerprinting, process-aware attacks

ACM Reference Format:
Esha Sarkar, Hadjer Benkrouda, and Michail Maniatakos. 2020. I came, I saw, I hacked: Automated Generation of Process-independent Attacks for Industrial Control Systems. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20), October 5–9, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3320269.3384730

1 INTRODUCTION

Industrial Control Systems (ICS) are used to monitor and manage/control physical processes. ICS are often deployed in Critical Infrastructures (CIs) such as nuclear power plants, chemical plants, wastewater treatment facilities, critical manufacturing, transportation systems, etc. These systems serve wide populations and are closely connected to everyday life. Attacks on ICS have scalable consequences and can lead to severe financial losses caused by downtime or physical damages, and can also even lead to human fatalities [43]. The effect of ICS attacks has been heavily researched/proven by academia and repeatedly seen in real-life attacks. Most prominently, Stuxnet [12], an attack that targeted an air-gapped nuclear/chemical plant, caused substantial production issues from damaged equipment while the alarm thresholds remained unaware of the infection. Similarly, the energy sector has also been attacked, the Ukraine blackouts being the prime example [28]. The WannaCry ransomware affected ICS, causing extended downtime and financial losses [6]. The latest incident was an attack on the Schneider Electric Triconex safety instrumented system affecting a major facility in the Middle East [11].

In the past, ICS environments were considered "secure" from attacks originating from cyberspace because the devices in these environments were air-gapped [7]. With the prospect of improved efficiency and better economics, ICS environments started to connect to corporate networks and the internet. This exposed ICS environments to a plethora of cyber-attacks [20]. At the same time, to reduce development and maintenance costs, vendors are increasingly opting for the use of more Commercial Of The Shelf (COTS) components for industrial devices [47]. General purpose computers, for example, host many ICS components such as Human Machine Interfaces (HMIs) and Engineering workstations that interact directly with Programmable Logic Controllers (PLCs)[43]. Therefore, malware affecting commercial microprocessor-based devices can be seamlessly ported to industrial environments. Regular patching, however, is extremely challenging in industrial settings due to the needed downtime, which in many cases could be unacceptable. Furthermore, industrial devices have a lifespan of many decades. This results in a substantial portion of these industrial processes to be based on legacy or deprecated devices, keeping ICS environments exposed to known vulnerabilities even after patches are released.

Although the term "ICS" is used as an umbrella term for systems spanning many sectors, each has its own customized deployment. State-of-the-art literature on ICS vulnerability discovery and deployment have focused on attacks that assume complete or high observability of the target [22, 25, 46]. Although these attacks provide insight into the consequences of plant-wide malicious manipulations of control and sensor data, they do not reflect realistic challenges and scenarios. A more dangerous class of attacks is process-aware attacks, which are operationally stealthy, customized, and result in calculated damages [44]. To design these attacks, a sector-specific analysis has to be carried out.

In all of these approaches of security assessment of ICS, three important considerations are missing: 1) Most attacks assume full visibility of industrial systems or assume complete controllability of at least one type of device (for example, all the controllers in the plant), 2) There is minimal discussion on automation for reconnaissance, attack launch, and attack sustenance phases and 3) There is a scarcity of generic attack vectors applicable to a variety of sectors. The diverse nature of ICS inherently limits the creation of a generic framework for its security evaluation, but we leverage the similarities amongst ICS sectors, processes, and common control algorithms to overcome these limitations and identify boundaries of dynamic attacks. In our investigated scenarios, we consider an attack with constrained access to the industrial plant. Our attack generation begins at a single HMI Windows-based computer, which can be reached by traditional malware. We propose a methodology of automated reconnaissance and attack development, where the adversary can cause stealthy process-aware attacks. We summarize our major contributions as follows:

- **ICS sector fingerprinting using HMI screenshots:** We constructed a database of publicly available images of HMIs and trained machine-learning models that can classify an HMI screenshot to be part of a certain ICS sector, thus successfully identifying the industrial sector.
- **ICS process fingerprinting using ICS binaries:** Once the industrial sector is identified, the specific target process must be fingerprinted. We use a database of publicly available ICS binaries to train machine-learning models to identify the specific process they control within that sector.
- **Generic methodology of designing perturbation-based attacks:** Given a constrained attacker with partial visibility of the process; we leverage control theory to design attacks that can be controlled in behavior. These generic attacks target any system that uses the widely used PID controllers while remaining within operational points, driving the plants into sub-optimal states.
- **End-to-end case study:** We develop an end-to-end demonstration in Tennessee-Eastman, a benchmark chemical process, and present the feasibility of the presented attacks.

After positioning our work in relation to the state-of-the-art in Section 2, the rest of the paper is structured as follows:

- **I came (Section 3):** Describes our threat model and our assumptions about the attacker capabilities and entry points.
- **I saw (Section 4):** Describes our fingerprinting methodology based on images captured from the HMI, as well as binaries extracted from the PLCs.
- **I hacked (Section 5):** Describes a generic, control-theoretic methodology for developing process-aware attacks assuming limited visibility in the process.

The end-to-end case study is presented in Section 6, followed by discussion of related defenses in Section 7.

2 RELATED WORK

To the best of our knowledge, automated reconnaissance using HMI images, and PLC binaries has not been explored in the literature. Analysis of PLC source code [48] and binaries has focused on disassembly, decompilation, finding vulnerabilities, ensuring the safety of critical code, payload design, and general-purpose reconnaissance of controllers. PLC infections can be identified from the analysis of PLC source code, ensuring the security of safety-critical code [49]. For payload generation, SABOT [33] gathers intelligence from source code analysis to automatically understand what kind of code manipulations would lead to attack goals being fulfilled. ICSREF [21] is an open-source framework that allows general purpose PLC binary analysis. In all of these approaches, PLC code analysis has not been used to fingerprint a particular process.

Table 1 summarizes the state-of-the-art process-aware attacks and countermeasures for various ICS sectors. While there are many sector-specific attacks for Energy [19, 23, 29], Water [1, 2, 17, 25, 31, 35], and Chemical [8, 22, 26, 46] sectors, research on generic payload generation [17, 32, 33] is the closest to our work. We differ from ladder logic, bombs [17] in the impact of our attack vector: We cause specific sustainable manipulations whereas it focuses on random manipulations. [32, 33] are more generic because they try to dynamically perform reconnaissance while designing payload but both of them have limitations in scope: [33] only operates on a specific language for PLC programming, Instruction List, which has been deprecated since 2012 and [32] gives a very high-level discussion on how to perform attacks on Boolean type instructions. From the table, we also see there is a scarcity of research on attacks which are generic and have a single entry point. Finally, [8, 31, 32, 46] discuss attacks that cause damage while remaining within the operational limits, but none of these attacks enumerate the generic steps for launching PLC-based stealthy attack which is not dependent on any PLC language.

ICS environments have also been exposed to real-world attacks. Stuxnet targeted the PLCs that controlled the centrifuges of a nuclear plant, subsequently alternating their frequency between preconfigured high and low thresholds to achieve physical destruction [12]. The attackers also remained stealthy by spoofing the sensor data that connects to the SCADA to report normal operation. [28] describes the attack on the Ukrainian power grid, where the adversaries used HMIs in the SCADA system to open the circuit breakers and malicious firmware on serial-to-ethernet devices that ensured that remote commands from operators were disabled. The attackers in Havex inject the target with malicious code that allows data collection such as stealing credentials, taking screenshots and file transfers [37]. This attack can be considered as a reconnaissance attack. An attack on a German steel-mill is known to have infected a controller and impacted its furnace [36]. Trisis gains access to a Schneider Electric's Triconex safety instrumented system (SIS) PLC and aims to change its ladder logic [11].

We observe from the real-world attacks that an adversary may at most infect a single device initially, differing from the academic literature which assumes extensive knowledge, visibility, and control over the process. A real-world attack called Havex [37], is the only known attack that assumes no knowledge of the sector/process (rows 4,5 in Table 1), but it is only a reconnaissance malware with no dynamic payload generation (rows 10,11,12 in Table 1).

3 THREAT MODEL

As mentioned earlier, most of the attacks in the ICS research literature either assume extensive observability and controllability of

Work	[33]	[32]	[46]	[26]	[29]	[17]	[31]	[19]	[23]	[8]	[1] [2]	[25]	[22]	[15]	[12]	[28]	[36]	[37]	[11]	**Our work**
Real/Academic	A	A	A	A	A	A	A	A	A	A	A	A	A	A	R	R	R	R	R	A
Sector demonstrated on	G	G	G	C	E	W	W	E	E	C	W	W	C	E	N	M	G	E	G	G
Target sector knowledge	●	●	◐	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○	●	○
Target process knowledge	◐	◐	○	◐	●	●	◐	●	●	●	●	●	●	●	●	●	●	○	●	○
Point of infection	P	P	P, A	P, A, S	S	P	CN, S	S	CN, P	S	CN, SC, S	S, A	P, S, A	P	GPC, N, P	GPC, N	GPC	N	GPC, P	GPC
Sensors/sensing signal	○	○	●	●	●	○	●	●	○	●	●	●	●	◐	◐	○	○	○	○	○
Controllers/controller variables	●	●	●	●	○	●	○	○	●	○	◐	○	●	●	●	○	●	○	●	●
Actuators actuating signals	○	○	●	●	●	○	○	○	○	○	●	●	●	◐	○	○	○	○	○	○
Attacker defined specification	●	●	○	○	○	○	○	○	○	○	○	●	○	○	●	○	○	○	●	○
Exhaustive vulnerability study	○	○	◐	●	●	○	○	●	○	●	●	○	●	●	○	○	○	○	○	●
Undirected attacks	○	○	○	○	○	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○
Feasibility of stealthy attacks: compromised component	○	●:A	●:A	○	○	○	●:S	○	●:S	◐:S	◐:S	◐:S	◐:S, A	○	○	○	○	○	○	●:P

Table 1: Summary of real-world and academic process-aware attacks. We denote the ICS sectors by W: Water, C: Chemical, E: Energy, N: Nuclear sectors, and G: to represent generic attacks that are not specific to any sector. To denote compromised components, we use P: PLCs, A: Actuators, S: Sensors, N: enterprise Network, CN: Control Network, and GPC: General Purpose Computer. Cyan rows (rows: 4,5): pre-infection knowledge required by the attacker to launch the attack. Gray rows (rows: 7,8,9): the degree of plant visibility assumed in the attack for launch or evaluation. Red rows (rows: 10,11,12): the type of attacks performed in the work. ● represents the completeness of the parameter used in the table.

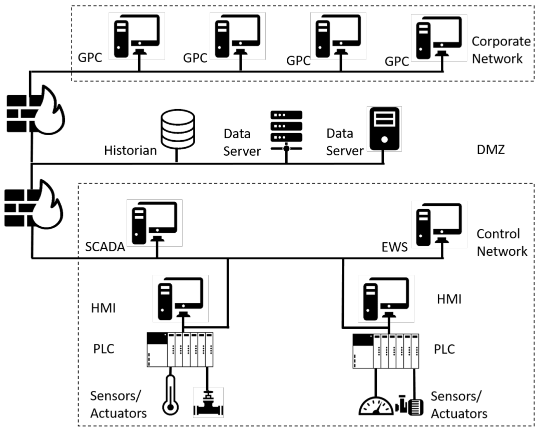

Figure 1: A typical ICS layout [43].

the entire process and/or assume extremely detailed knowledge of the plant while crafting attack vectors. In contrast, for our threat model, we make the following assumptions:

- The attacker has no prior knowledge of the ICS;
- The attacker can reach and execute a program on an HMI;
- The HMI connects to a PLC and its binary can be extracted.

The first assumption is the difference of our work to all related work. In our threat model, the attacker has no knowledge about the process and tries to extract as much as possible automatically.

The second assumption is based on real-world attacks where the attackers used traditional techniques to control a Windows machine. Through some entry vector, such as infected USBs, social engineering, or (spear) phishing, etc., a piece of malware enters the enterprise network and moves laterally, infecting other Windows machines. The Ukraine power grid attack [28] reached the HMI controlling the circuit breakers by stealing the VPN credentials of the substation engineers through social engineering.

The third assumption is required for delivering a process-aware attack using dynamically acquired knowledge. Without it, our attack stays at the reconnaissance stage only. An HMI, however, is designed to give commands and receive data from PLCs, so a network path to PLCs certainly exists. A simple Nmap prioritizing known industrial protocols/ports is sufficient to identify potential targets, followed by a connection to the PLC. Modern PLCs use FTP for file transfer and very frequently allow anonymous access or use default passwords. This can be used to download PLC binaries.

Supervisory Control And Data Acquisition (SCADA) which monitors and controls all components has complete observability of the plant (See Fig. 1) and infecting the computer hosting SCADA is the best-case scenario for an adversary. However, we also consider plants with decentralized systems hosting several HMIs; the attacker in this scenario will have less visibility if only one computer hosting one HMI is infected. The HMIs may monitor a subset of devices in the plant rather than the entire plant. We thus explore the possibility of developing and delivering attacks given such limited visibility to the process. We emphasize that our constrained threat model aims to investigate the requirements and possibilities for an attacker to successfully launch an attack in this scenario.

Essentially we aim to launch process-aware attacks: These attacks are advanced because they collect intelligence to build the final attack vector but do not act to achieve quick financial gains or cause volatile attacks. Instead, the payloads weigh the capabilities, look for an opportune moment, and launch the attack while ensuring stealthiness. These characteristics require complete reconnaissance of the target. While public information about plants [23], device fingerprinting [14], social engineering [4] and model simulation [22] may be used to gain pre-infection knowledge, this leads to crafting manually designed target-specific attack vectors. We, on the other hand, aim to build attack tools that can automatically gain knowledge about the plant and can design payloads accordingly.

As discussed in Section 1, it is difficult to build a generalizable and stealthy payload that can guarantee successful infection across all the different ICS sectors. Therefore, we call the attacks *generalizable* if they leverage integral components that are present in any ICS environment. An example of such components includes HMIs, PLCs and general purpose computers. Furthermore, we define *stealthiness* of the payloads from the process point of view, since random changes in process variables may trigger very simple

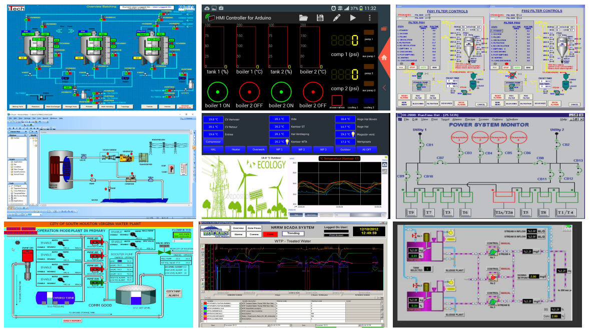

Figure 2: A subset of our constructed dataset. Each row depicts images from the same sector: Chemical sector, Energy sector, and Water sector respectively. The images depict the sharp differences between images belonging to the same sector (along the row) and similarities between images belonging to different sectors (along the column).

threshold-based alarms. Still, experimental results in Section 4.3 clearly present the bounds of the dynamically developed attacks.

In summary, the attack scenario is as follows: An attacker reaches and can execute code on a general purpose computer hosting an HMI. Next, the code takes a screenshot of the HMI and passes it through a pre-trained classifier (Section 4.1). The classifier performs both image-based and extracted text-based classification. When the ICS sector is identified, the code traverses the network to scan for potential target PLCs. After that, when the code reaches a PLC, it downloads the control binary from the PLC and analyzes it to classify the process controlled by that PLC. It also extracts interesting parameters (Section 4.4). Using all the information dynamically extracted, the code can deploy one of many operationally undetectable attacks. These can either be attacks that cause stable perturbation to a physical quantity resulting in non-optimal operation or attacks that cause an oscillating perturbation within upper and lower limits that result in long term physical damages to the system's infrastructure. Another type of attack is a ticking bomb attack that allows the attacker to escape before the attack is deployed just by manipulating the PLC binary using ICSREF.

4 MACHINE LEARNING-BASED RECONNAISSANCE

To effectively attack an unknown industrial setting, we first need to identify the ICS sector and then the specific ICS process monitored and controlled by the HMI. For simplicity, we use the following terms in the context of target fingerprinting:

- **ICS Sector fingerprinting** is the categorization of the plant under one of the 16 Department of Homeland Security (DHS) CI categories [34]. This gives a more generic view of the plant.
- **ICS Process fingerprinting** is the identification of the process being controlled by the infected PLCs. This provides information with more granularity.

To build knowledge, we use publicly available information (images and PLC binaries) to build machine learning models that can efficiently fingerprint and extract data to launch process-aware attacks post-infection. We do not attempt any manual reconnaissance to gain any specific information about the target. ICS sector fingerprinting requires a more generic view of the infected plant:

Sector	No.	Top 3 languages
Chemical	179	English (105), Czech (7), Turkish (7)
Commercial	8	English (7), French (1)
Communications	1	English (1)
Manufacturing	2	English (2)
Dams	4	English (4)
Defense	0	-
Emergency	0	-
Energy	162	English (126), Czech (7), Turkish (3)
Financial	0	-
Food	5	English (5)
Government	0	-
Healthcare	0	-
IT	0	-
Nuclear	8	English (8)
Transportation	8	English (5), Czech (2), Slovak (1)
Water	144	English (123), Czech (9), Thai (3)

Table 2: Summary of our constructed dataset of HMI images.

HMI screenshots can provide that information in a condensed form. Therefore, we mainly use HMI screenshots to learn more about the ICS sector which will be further explained in subsection 4.1. On the other hand, the code running on the infected PLC provides information about the specific process. Thus, we choose PLC control binaries to perform ICS process fingerprinting, as explained in subsection 4.2. We also performed process fingerprinting using HMI screenshots and sector fingerprinting using PLC binaries and observed limited accuracy (these results appear in the Appendix).

4.1 ICS Sector fingerprinting

Constructed dataset: We used `google_images_download` python package to automatically download 1000 images for each of the 16 critical infrastructure sectors, as defined by the US Department of Homeland Security [34] and presented in the first column of Table 2. We further augmented the dataset (with various search strings) using Bing (≈2k), using Flickr (≈40) and searched for ICS screenshots in Shodan [30] (≈20). After careful pruning, we recovered 521 useful HMI images, as summarized in Table 2. Details of dataset construction appear in the Appendix. We decided to focus on Chemical, Energy, and Water sectors for two reasons: a) Most of the real-world attacks target these three sectors (as discussed in Section 2), and b) Since machine learning requires large datasets for robust models, these are the only sectors that can provide meaningful results.

We faced two contradictory challenges in building a generalized classification model. First, training over a small dataset always has a higher chance of overfitting in deeper and larger networks which meant that the classifier would suffer greatly in test accuracy. And second, the images were very diverse (beyond rotation, shear, and zoom modifications, as seen in Fig. 2) which required a deeper network for learning the intricate features. The following subsections describe our methodology followed to design the ML models.

4.1.1 Classification based on raw HMI screenshots. We use various DNNs using Convolutional layers (Conv), ReLU activation layers (ReLU), Max-pooling layers (MP) and Fully Connected (FC) layers to classify the raw HMI screenshots. To finalize our architecture, we empirically evaluated various designs based on high accuracy

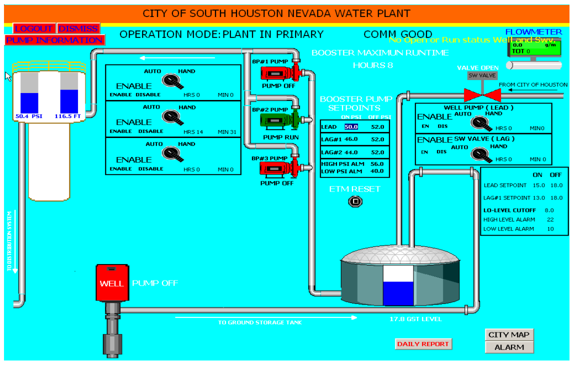

Figure 3: HMI screenshot of a water treatment plant in United States. The Siemens Simatic software-based SCADA was hacked and a screenshot was posted on pastebin [9].

Network architecture	Image size	Training accuracy	Test accuracy	Epochs
DNN-1	128 × 128	98.84	69.12	6
	256 × 256	97.80	71.52	3
	512 × 512	70.35	62.13	29
DNN-2	256 × 256	98.99	71.021	15
DNN-3	256 × 256	92.79	69.221	5
DNN-4	256 × 256	91.32	67.88	95
DNN-5	**256x256**	**99.15**	**80.907**	**51**

Table 3: Experimental results for different image sizes and architectures described in Section 4.1.1 on constructed dataset. The number of epochs in the table represent the iteration at which the highest accuracy was achieved.

and low overfitting, by changing the number and type of layers, pre-processing techniques, and overfitting countermeasures.

Selection of optimal size of images: Resizing images to smaller dimensions may approximate the information while, bigger dimensions may tamper the genuine features, and significantly increase the training time. To find the optimal size, we built a small network of (Conv-32,ReLU,MP) + (Conv-32,ReLU,MP) + (FC-64,ReLU) layers and called it DNN-1. Table 3 summarizes the training and test accuracy for different image sizes. Images of size (128x128) render best training accuracy (98.84%) while images of size (256x256) offer slightly higher test accuracy (≈ 2% higher). However, in both cases, we observe high evidence of overfitting from the difference between training and test accuracy and the smaller number of epochs required to train. Larger image sizes of (512x512) are also unsuitable because of very low training accuracy indicating an inability to learn features and hampering the overall efficiency of the payload. Moreover, using bigger images increases computation during training as well as evaluation. Therefore, we choose (256x256) for further experiments to deduce the final architecture. We take two measures to reduce overfitting- data augmentation and dropout.

Data Augmentation: Data Augmentation has been proven to be extremely effective in forcing the model to learn robust (generic) features. In our case, when we trained the same architecture but without any DA, we achieved our best training and test accuracy in just 3 iterations reflecting extreme overfitting. Some augmentation mechanisms like rotate, vertical flip, and shear are not viable in HMI based sector identification task; therefore, we focused on width and height shift along with zoom range obtaining the following [training, test] accuracy. For height-shift (0.2) we achieved an accuracy of [99%, 69.987%] in 23 epochs and for zoom (0.15), height-shift (0.2) and width-shift (0.2), we achieved [73.35%, 60.15%]. We found height-shift (0.1) and width-shift (0.1) to be the best trade-off between accuracy and overfitting problems, where we achieved [96.54%, 67.03%] in 49 epochs.

Dropout: Since dropout layer removes a percentage of neurons during training, it reduces the co-adaptation of neurons. This prevents the model from overfitting by forcing the neurons to learn features without being dependent on other neurons. This regularization method is extremely useful in preventing overfitting [42]. In our experiments, when DNN-1 was trained for image size (256x256) without dropout layer (DNN-2), a test accuracy of 71.021% was achieved which is almost similar to the test accuracy of DNN-1. But using dropout layer, we were able to stabilize the training, and the best model was achieved after learning through more number of iterations i.e. through 15 iterations instead of 3 as for DNN1.

Transfer learning: The motivation of transfer learning is that a sufficiently trained model is independent of the data with which it was trained and the *knowledge* gained from one dataset can be used to learn features on another dataset. The pre-trained model may further be tuned (by training only a few layers) to cater to the particular dataset. Before applying transfer learning, we trained two slightly deeper networks than DNN-1 to observe the impact of deeper architectures in successfully creating efficient models. We performed experiments with (256x256) sized images for deeper layers: DNN-3 with architecture (Conv-32,Conv-32 ReLU, MP) + (Conv-32, ReLU, MP) + (Conv-64, ReLU, MP) + (FC-64,ReLU) and DNN-4 with architecture (Conv-32, ReLU, MP, Dropout) + (Conv-32, ReLU, MP, Dropout) + (Conv-64, ReLU, MP,Dropout) + (Conv-128, ReLU, MP, Dropout) + (FC-64,ReLU). From Table 3, we infer that the difference between test and training accuracy lowers, and training stabilizes with increment in the number of layers. Thus, we choose a smaller number of trainable layers in our network with a larger number of non-trainable layers with dropout and data-augmentation for improving test accuracy using transfer learning. We performed experiments on transfer learning with VGG16 weights available using the Keras module for Python 3.6.5. The architecture of the final model has the non-trainable layers of VGG16 architecture, followed by a fully connected layer and Dropout layer (DNN-5) trained using data-augmentation. We observed that with transfer learning, we stabilized the training process, reduced the gap between training and test accuracy and improved test accuracy to 80.907% from the best test accuracy of 71.52%.

Insights: For conventional image classification problems, the model learns to use similarities in shapes and edges but here, the same chemical plant HMI may be designed to represent data in different ways (one may be visual, depicting reactors, devices, etc. and other may depict just the sensory information). Moreover, the similarities in colors can misguide the model into using RGB values to classify an HMI as similarly colored images may belong to different classes (Fig. 2). In a nutshell, the differences are more complex than orientation, background, illumination, or color differences between images which prevents high accuracy using raw HMI screenshots.

Parameters			Multinomial Naive Bayes						Support Vector Machine					
No. of Features			Training Accuracy (%)			Testing Accuracy (%)			Training Accuracy (%)			Testing Accuracy (%)		
HMI Text	FB	Strings	HMI Text	FB	Strings	HMI Text	FB	Strings	HMI Text	FB	Strings	HMI Text	FB	Strings
100	30	500	75.70	84	**82.30**	70.21	70.58	**80**	80.30	90.90	91.17	77.6	76	73.33
1000	35	1000	**87.46**	84.85	85.29	**84.04**	76	80	88.74	90.09	91.18	75.53	76.47	73.33
1500	40	1500	87.79	81.82	85.30	81.91	76.50	80	90.02	93.9	97.05	75.53	82.35	73.33
2000	45	2000	89.76	**84.85**	88.23	80.91	**82.35**	80	91.04	93.93	97.05	75.53	82.35	73.33
Feature Selection Statistic														
χ^2			85.67	84.83	97.1	82.97	82.23	80	87.72	93.9	97	79.78	82.35	73.33
Mutual Information			87.46	84.85	82.30	84.04	82.35	80	88.74	93.93	97.05	75.53	82.35	73.33
F_score			86.18	84.8	76.5	80.85	80.23	80	89.25	93.9	79.41	74.46	82.4	73.33

Table 4: Experimental results for different number of features and feature selection statistic for SVM and MNB models for HMI text-based, FB-based and Strings-basedclassification. The first section of the table gives details for selection of number of features. Second part of the table depicts the experiments for feature selection statistic.

4.1.2 Classification based on text from HMI screenshots. Manual inspection of the screenshots reveals that many HMIs provide an interface for the data collected from the field devices (e.g., temperature values) without distinct diagrams to illustrate the measurements (example: Fig. 3). Thus, learning algorithms suffer in accuracy when learning from strictly pixel-based features. We tune the parameters for a Support Vector Machine (SVM) model and Multinomial Naive Bayes (MNB) model on text recovered through Optical Character Recognition (OCR) to finalize the ML parameters for text-based classification and summarize the results in Table 4 (Col:"HMI Text").
Cleaning strings: In our dataset, more than 70% of the images contain English text strings with Czech, Turkish, and Thai being the other popular languages. Translating them sometimes did not yield legible words or combination of letters, especially accents. Thus, we removed any character that was not a letter. We also removed digits because they did not provide any useful information about the sector to which a particular HMI belonged. For text extraction and translation, we use Google Cloud Vision API.
Feature selection: We first use Mutual Information (MI) as a statistic on both models to find the optimal number of features while being careful to avoid overfitting. Then, we used χ^2 statistic, and F-score to find the best correlation between target labels and features using SVM. From Table 4 we see χ^2 statistic (test accuracy: 79.78%) and MI (test accuracy: 75.53%) perform better than F-score (test accuracy: 74.46%) with χ^2 statistic being slightly better in test accuracy. We observe that choosing the top 1000 features from high mutual information with MNB gives the best result (test accuracy: 84.04% and low overfitting).
Model Selection: We experimented with two models, MNB and SVM with linear kernels because both the models are extensively used for text-based classification like sentiment analysis or spam detection [24]. We reduce the impact of overfitting by K-fold cross-validation using 5 folds and we observe that the test accuracy for MNB is higher in all the experiments as compared to SVM. Therefore, we select MNB with mutual information statistic to select the top 1000 features as the final model.
Insights: We investigated the top features out of total 7592 features related to our text-based model and listed the top features in Table 5. Manual inspection of top features reveals that while some features like 'water' for the Water sector, 'kw' (kilowatts) for the Energy sector and 'reactor' for the Chemical sector are intuitive, there are some features that create confusion like 'fit' for the Water sector.

4.1.3 Combined classification model for ICS sector identification. To further improve accuracy, we make our own classification model that leverages the features learned from raw screenshots as well as from the text strings extracted from them. We propose a parallel classification model where the text-based classification model works in conjunction with the pixel-based classification model. Fig. 4 shows the parallel architecture we use in our classification model. In this model, an HMI screenshot is run through the image classification, and a class is predicted with a probability of that screenshot being in that class. In parallel, the image is also run through an Optical Character Recognition (OCR), and text strings are extracted. The text strings are translated if they are in any language other than English. The strings are then cleaned, and the important features are extracted. MNB used in our model then classifies the screenshot with a prediction probability. We then compare the prediction probabilities and choose the sub-model whose prediction probability is higher. We choose the classification scheme based on prediction probability because it is a reflection of the confidence level of prediction from each model. Thus, having a parallel architecture ensures a more accurate prediction. We performed experiments and the test accuracy increased to 88.29% from 80.907%, if using only image-based classification (Table 3) or from 84.04%, if using only text-based classification (Table 4).

4.2 ICS Process fingerprinting

As explained before, an ICS environment in any sector is comprised of control processes. Chemical plants (a sector), for example, involve the control of flow of chemicals, temperature in mixers, pressure in pipes, etc. Each of these processes is typically controlled by a PLC. To attack the system meaningfully, we have to target the right PLC by fingerprinting the selected process.

PLCs use programming languages with both textual and graphical representations, such as ladder diagrams, structured text, instruction list, function block diagrams, and sequential function charts (as described in IEC 61131-3 standard). Engineering workstations run an integrated development environment (for example, Codesys) that is used to compile the control code into binary format and download it to the PLC. We leverage these control binaries

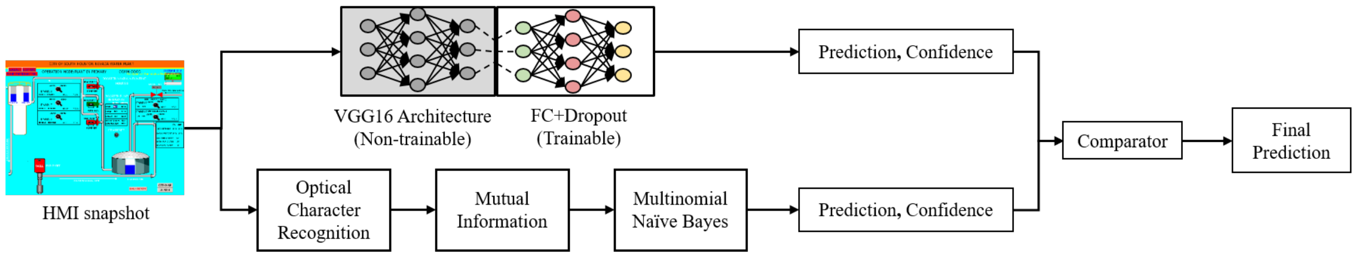

Figure 4: Parallel architecture of the final classifier to fingerprint an ICS sector based on an HMI screenshot.

Sector	Top features
Chemical	reactor, gal, kg, kv, kw, man, mw, pav, power, sp
Energy	cb, kv, kva, kvar, kw, mvar, mw, pf, power, sel
Water	pump, cw, water, fit, flow, hr, kv, kw, mw, sludge

Table 5: Top features for HMI text-based classification.

Sector	Binaries	Top 3 languages
Chemical	18	German(7), English(5), French(3)
Energy	19	German(7), English(6), Cezch(2),
Transportation	6	German(2), English(2), Polish(1),
Water	9	German(2), English(2), Polish(2)

Table 6: Description of the control binaries dataset.

to evaluate their efficiency in classifying the sector that the PLC belongs to and the process running on our target PLC. Specifically, we extract two pieces of information: 1) The Function Blocks (FBs) (Section 4.2.1), and 2) The ASCII strings (Section 4.2.2). FBs resemble functions in imperative programming languages. They are used as black boxes for frequently reoccurring processes, such as control algorithms (PID, Integral, Derivative), timing functions (triggers, timers), and networking functions (MODBUS, TCP) [21]. Extracting FBs from binaries can provide rich semantic information about the process. The strings in a binary are dynamic and include descriptive error messages, input/output prompts, or other information that might be used to identify the process of the PLC.

Constructed dataset: In order to extract FBs from the binaries, we use ICSREF [21]. ICSREF, at its current version (1.0), works only with Codesys v2.3 binaries. Since PLCs control processes are part of the critical infrastructure, real binaries are not easily accessible. Therefore, we turned to publicly available binaries and obtained 69 control binaries without crossing the line of legality. 52 out of the 69 contained useful information and we were able to categorize the binaries into 4 sectors (Chemical, Energy, Transportation, and Water and wastewater management) and 3 control processes (pressure, temperature, and time controls) as seen in Tables 6 and 7, respectively. We used these processes as a proof-of-concept to train and test our machine learning model. To avoid the pitfalls of a small dataset, we ensure diversity in both the training and testing.

4.2.1 Classification based on PLC Function Blocks. **Cleaning FB names:** After automatically extracting the FB names, individually, we had to cleanse the dataset. ICSREF is equipped with a function called *hashmatch* that extracts the FBs in PLC binaries, hashes them to create a signature and then matches them to a built-in library [21]. This function also provides the relative address at which the FB is located. The first stage of data cleaning is removing these addresses as they add no semantic information. Next, because some FBs are generic and can be used in any process (e.g., ETHERNET_MODBUSMASTER_TCP, MD5_DD, RTU_TO_ASCII), they had to be excluded as well. Initially, we ran experiments using all the FB names, including general purpose FB. This yielded limited accuracy, 50%. After removing common FBs, we observed an improved testing accuracy reaching up to 82.35% (Table 4).

Feature Selection: To extract the feature words from the data we used two different methods: The frequency of words, using count, and the importance of a word relative to the entire corpus, using term frequency-inverse document frequency (TF-IDF). Selecting the words using TF-IDF yielded better results. To select the most effective features, we follow the same procedure as subsection 4.1.2. We identify that the *MI* performed the best, by giving the highest testing accuracy and showing the least signs of overfitting. Furthermore, we note that the optimal number of features is 45.

Model Selection: Table 4 shows the variation of training and testing accuracy with different selection parameters and number of features on both the MNB and SVM models. FB based classification performed better using the MNB model with training and test accuracy being 84.85% and 82.35% respectively.

Insights: We extracted the top features (FB names) that contributed to the classification of each binary to the corresponding process. Table 7 shows the list of FB names reciprocal to top features. Even with a limited dataset, it is evident that these FB produce meaningful and unique features (e.g., fb_time, ramp_init) describing each process. There are other features, like blink, for example, that might not aid in classification. During our experiments, the ML model extracted 45 features. Our tests show that for this model, increasing the number of features resulted in higher testing accuracy.

4.2.2 Classification based on PLC binary strings. We also use the ASCII strings embedded in the binary. Using the Unix command *strings*, we were able to find printable strings. We found that more than 75% of our dataset contains meaningful strings.

Cleaning strings: Our algorithm extracts text from control binaries using the *strings* command, and then delimits the words and translates them into English. Our binaries included 10 languages, of which more than 30% were in German (as shown in Table 6). We automatically translate the strings using Google Cloud Vision API. Since *strings* prints all printable strings of 4 characters and longer, this resulted in many meaningless words that we discarded.

Feature Selection: We used same feature selection procedure as in subsection 4.2.1 Although, having 2000 features results in a better training accuracy, the discrepancy between training (88.23%) and testing (80%) accuracies elude to overfitting. We thus choose our optimal number of features to be 500.

Model Selection: From our experiments on SVM and MNB, the MNB model performed better and therefore, we use it for text-based classification of the binary strings. We obtain a training and testing accuracies of 82.30% and 80%, respectively with MI statistic.

Process	No.	Top Features-FB based Classification	Top Features- Strings based Classification
Pressure Control	17	pack, mc3_power_init, mc3_reset_init, mc3_stop_init, hysteresis, fu_linear_2punkt, extract, ramp_int, prot_wait_for_init, blink	pressure, temperature, value, pump, sys, data, time, file, heating, cooling
Temperature Control	17	integral, hysteresis, derivative, pid, ramp_int, fu_linear_2punkt, charcurve,blink_init,hysteresis_init, blink, prot_wait_for_init	state, save, heating, send, manual, hour, alarm, active, heat, temp, cooling
Timer	18	fb_time, fb_stairwelllight1, derivative_init, integral_init, pos_750_635_init, pid, integral, fb_latchingrelay, derivative, pack, fbtimeswitch, prot_wait_for_init, hysteresis_init, blink	value, alarm, active, data, temp, file, set, send, pressure, time, sys, cooling, heating, min, start, open, status

Table 7: Top features that are used in process both fingerprinting methods in control binary based classification.

Fingerprinting with HMI		Test Accuracy (in %)			
ML model	Size	Chemical	Energy	Water	Total
Image only	M: 120MB	98.8	69.7	75.9	80.9
Text only	M: 94.4KB D: 472.2 KB	100	75.8	75.9	84.0
Combined	≈120MB	100	84.8	79.3	88.3
PLC binaries					
ML model	Size	Pressure	Heat	Time	Total
FB only	M: 2.9 kB	87.7	56.2	100	82.4
Strings only	M: 24.7 kB D: 39.6 kB	81.4	76.3	83.3	80
Combined	M: 27.6 kB D: 39.6 kB	100	76.3	100	93.33

Table 8: Boundaries for ICS fingerprinting. Green cells: maximum attack confidence, M: Model, D: Dictionary.

Insights: Strings for process classification showed high training accuracy while the testing accuracy was capped at 80%. This can be attributed to the size of the dataset or the specificity/static nature of processes. It is important to note that increasing the number of features beyond 500 resulted in signs of overfitting; this can also be due to the size and diversity of the dataset. To have a better understanding of how the binaries were classified, we examine the features used for classification. The top features included process-specific words such as pump, pressure, etc. as shown in Table 7.

4.2.3 Combined classification model for ICS process identification. Similar to the combined model used for sector classification (subsection 4.1.3), we also parallelize the two process classification ML models to achieve higher accuracy. Both FB and string ML models are leveraged to classify the process correctly. The combined model assesses the prediction probability from both the FB's and string's based classification models and chooses the class with the higher confidence level. This model is especially useful in cases when the binary does not contain useful information from either source (FB or strings). Our tests show that our combined model results in more than 10% increase in testing accuracy of both models (FB based: 10.98%, Strings based: 13.33%), with a final accuracy of 93.33%.

4.3 Attack boundaries

The proposed reconnaissance methodology can be effectively used in un-targeted attacks on different ICS environments without human intervention. Nonetheless, any statistical (ML) model can introduce a probability of failure which is reflected from the accuracy metric. Table 8 discusses the boundaries of reconnaissance.

We observe that the text-based ML model is much smaller (94.4 KB) than image-based ML model (120 MB) for ICS sector fingerprinting. The dictionary size includes top features and their translation in the detected languages (≈ 27). Although the accuracy of the combined model is 5% higher than a text-based model, the size of the payload is 3 orders of magnitude more. Therefore, we consider the text-based model to be a light-weight version of the payload. The attacker may choose the combined model if her capabilities allow for the payload to remain undetected. Chemical sector, in all the models, has highest reconnaissance probability.

For process fingerprinting, we observe that both the pressure and timer control processes have a high testing accuracy using FB-based and strings-based classification, respectively, with reduced payload size (2.9 kB and 24.7 kB respectively). We also include a dictionary of the top features (500 words) and include their translation to all the languages detected in our dataset (10 total), resulting in a 39.6 kB payload, if multiple languages are targeted. The combined model introduces more than 10% increase in testing accuracy and maintains a practical size of 67.2kB for all languages identified.

4.4 Intelligence collection

Before we move to process-aware attack, we also explore the intelligence that can be collected during the reconnaissance phase.

4.4.1 From HMI screenshots. While extracting text strings from HMI screenshots, we included some attack-aid strings like 'alarm', 'max', 'min', 'set-point', common process-specific variables (pressure, temperature, etc.), their variations and extracted the digits (if any) next to them. This helps the adversary in automatic payload-design and gives insights about the operational limits of the plant. We can see such useful intelligence collection from a real-life example in Fig. 3: Operational limits, alarm thresholds, and plant goals (set-points) can be extracted from the HMI screenshot.

4.4.2 From PLC binaries. PLC binaries also contain similar values like set points and hardcoded values (eg, PID values, Cycle time). PLC control binaries are unique and different for each vendor. Using ICSREF [21], we explored the possibility to use the Global INIT subroutine to extract hard-coded values such as set points and PID values. This analysis is enabled by the reconnaissance steps before. An adversary that gains access to values from the control binaries can use them effectively. In our end-to-end case study, we automatically extract and replace the PID values from the binary.

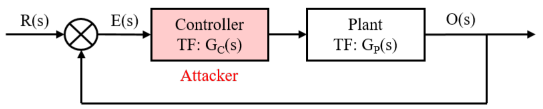

Figure 5: A typical control system with transfer functions and signals of each module. Observability of the attacker is depicted by the highlighted module.

5 CONSTRAINED PROCESS-AWARE ATTACK

A common characteristic of a process-aware attack is to cause sustainable and meaningful damage. Most of the work, however, achieves this attack objective using sensors or actuators[45]. [15] is the only work that discusses the construction of such attacks based on controllers, but there is limited discussion about how it can generalize. Moreover, [15] changes the signals to/from PLCs, essentially spoofing the sensing and actuating signals without any manipulation on PLC variables. In this work, we bridge this gap in the literature by enumerating the steps to cause PLC control code-based attacks that remain within the operational limits of the plant (like causing a sustainable decrease in production). These attacks are performed by just variable value changes in the control code in a generic PID controller, commonly used in ICS.

Since the attacks cause a perturbation (deviation) in the physical quantity, we name them *perturbation-based* attacks. These are two specific types of attack which remain within operational limits of the physical quantity: One that causes a stable change in a physical quantity (like stable decrease in production), and another that causes an oscillating perturbation within upper and lower limits (example, causing turbulence in pipes). Depending on the objectives, the adversary can program the payload to choose. Our formulation of attack methodology follows control theory convention to represent functions: The time- and frequency-domain functions are represented in small and capital letters, respectively.

5.1 Stable Perturbation attack

To induce a stable yet configurable error in a constant physical quantity, we use a control-theoretic approach of manipulating the Proportional-Integral-Derivative (PID) controller. PID-based controller algorithms are standard in ICS environment control and are available as a configurable FB in PLC engineering platforms (Codesys). Moreover, it is possible to extract and manipulate the PID variable values from the binary of the control code [21]. Thus, changing the Proportional (K_P), Integral (K_I) and Derivative (K_D) coefficients, becomes a natural choice to inject stealthy payload.

The transfer function, a ratio between output and input signals in the frequency domain, reflects the characteristics of the process. Let $G_c(s)$ and $G_p(s)$ denote the controller and plant transfer functions respectively, which define the characteristics of each of these modules. Any feedback loop in a control system tracks the desired specification (reference or set-point $R(s)$) with allowable deviation or Error ($E(s)$). From the definition of transfer function, the output $O(s) = G_c(s)G_p(s)E(s)$. A typical control system and the observability of an attacker is summarized in Fig. 5. The error expression $E(s) = R(s) - O(s)$, for any reference input and unity feedback is given by [16]:

$$E(s) = \frac{R(s)}{1 + G_c(s)G_p(s)} \tag{1}$$

Since Laplace transform of a constant-valued function like $r(t) = k$ is k/s, if the set-point is a constant (for example: constant pressure, constant temperature etc. where $r(t)$ is a constant physical quantity), then Eq. 1 becomes:

$$E(s) = \frac{k/s}{1 + G_c(s)G_p(s)} \tag{2}$$

A stable perturbation is analogous to a control error. Therefore, a stable perturbation $e(t)$ is an error measured in steady-state (when $t \to \infty$). From the Final Value Theorem (FVT) of Laplace transform, this is equivalent to measuring $sE(s)$ as $s \to 0$. Thus, the stable perturbation is $E_{SP} = \frac{k}{1+G_c(s)G_p(s)}$. An attacker with direct access to PLC source code, can change $R(s)$ or k directly. For an attacker with access to PLC binary, can only $G_c(s)$ coefficients using ICSREF.

At frequency s, the transfer function of a generic PID controller ($G_c(s)$) becomes [16]:

$$G_c(s) = K_P + K_D s + K_I/s \tag{3}$$

Thus, from Eq. 2 and 3 and expression for E_{SP}, we have two important inferences in terms of designing our attack strategy:

$$E_{SP} = 0 \ \forall K_I \neq 0$$
$$E_{SP} \propto \frac{1}{K_P}$$

We apply these control-theoretic rules to build our attack strategy as summarized in Table 9. First, the payload switches off the integral controller ($K_I = 0$). Then, it slowly decreases the value of proportional gain coefficient (K_P), while getting feedback on attack impact. Note, in our attack strategy, we avoid manipulation of K_D because it makes the system unstable [16].

5.2 Oscillatory perturbation attack

A special case of perturbation-based attacks are when the perturbations change very rapidly but, not beyond operational limits. These kind of attacks are very much dependent on the type of the plant (by extension $G_p(s)$) and may not be applicable to all the control loops in the plant. To achieve oscillatory perturbation, we use the empirical tuning philosophy of a common PID tuning technique called Ziegler-Nichols (ZN) PID tuning [5]. This technique is useful to empirically find the values of K_P, K_D and K_I for plants with simple transfer functions such that the controller meets the design criteria of output response. The method followed is that first the derivative controller and integral controller are switched off. Then, it is required to find the ideal *sign* of K_P, i.e. if a positive change in reference value reflects as a positive or negative change in the output response. We follow this method to find the ideal sign, however, we also experiment with the negative sign of K_P because with negative proportional gain, positive feedback increases oscillations in the system [16], which is the goal of the attack. Then, like in ZN method of PID tuning, we increase the K_P value to the point of sustained oscillations. The value of K_P changes the frequency and amplitude of oscillations and it can be experimentally evaluated to find the value for a particular frequency and amplitude.

5.3 Physically configurable attack-trigger time

These are attacks which get triggered at a configurable time solely based on values manipulated in the control code (by only changing K_P, K_I and K_D coefficients). They do not need any extra trigger logic; our attack model just changes values of existing variables of controller code to prevent getting detected by code injection

Attack type	Attack on	Attack design strategy	Enabler
Stable Perturbation	Set-point	New Set-point = Actual - Target perturbation	PLC source code
	K_P	$K_I = 0$; Stable perturbation $\propto \frac{1}{K_P}$	ICSREF
Oscillatory Perturbation	$\pm K_P$	$K_I = 0$; Increase $\pm K_P$ for sustained oscillations	ICSREF
Physically configurable trigger time	K_P	Trigger time $\propto \frac{1}{K_P}$	ICSREF

Table 9: Summary of payload characteristics.

defense mechanisms. Although both integral and proportional controller gains effect time response, the adversary can only manipulate the proportional controller to perform the attack since both our attack strategies have $K_I = 0$. Thus, in this case the closed-loop transfer function becomes $\frac{1}{1+1/K_P G_p(s)}$. Since, we aim to change the response characteristics of the system, we need to manipulate the time constants of the system. Before we calculate the time constant of the system, it is important to note that inverse Laplace transform of $\frac{1/a}{(1/a)s+1}$ is e^{-at}, where $1/a$ is the time constant. Now, for simplicity, let us consider a first-order control system i.e. power of s in denominator of the transfer function is 1, then the plant transfer function is of the form $\frac{A}{s+a}$ and the closed-loop transfer function is $\frac{K_P A}{s+K_P A+a}$. For such a transfer function, the closed-loop time constant is $\frac{1}{K_P A+a}$. For higher-order systems, similar analysis shows that the time constant is inversely proportional to K_P. Thus, a higher value of K_P will lead to lower value of time-constant which, in turn, leads to faster time-response.

5.4 Selection of attack

Completing the automated attack generation, in this subsection we provide recommendations on the choice of attack for the common ICS processes (from literature) along with the ones found in the constructed dataset of the PLC binaries. We state the following simple rules based on the characteristics of the attack to maximize its impact while adhering to the operational thresholds:

- **Choosing Stable perturbation attack:** Since this attack creates a simple deviation, any attack vector generated will drive the plant to a sub-optimal state. But the adversary must be careful not to force the deviation towards 'explosion' (high speed, high pressure, high temperature, etc.). However, since processes are designed for maximum functional profit, reducing them would incur financial losses. Thus, for pressure, temperature, speed, timer, flow-rate control loops, our general recommendation is to make the process less efficient/slower by selecting stable perturbations.
- **Choosing Oscillatory perturbation attack:** This attack leverages the un-modeled characteristics (e.g., oscillations of a product against the walls of the container/pipes causing quality degradation) and has potential for physical damage (e.g., Stuxnet). Since this attack involves complex parametric oscillations, attack vectors for smaller amplitude should be generated. Prime process targets are production, level, and valve control loops.

In our end-to-end case study, the payload chooses an attack based on the above recommendations. This can also be completely configurable based on the attacker's motivation.

6 CASE STUDY: TENNESSEE EASTMAN

In this section, we give an overview of the attack and present an end-to-end case study on a Hardware-In-The-Loop (HITL) testbed of a chemical plant. Tennessee-Eastman (TE), is a non-linear chemical process which takes in five input products (A, B, C, D, E), performs exothermic reactions, and produces two main products (G, H) and one by-product (F). The process reactions depend on temperature, pressure, and quality (molar concentrations) conditions. These reactions are performed by five major physical components: Reactor, Condenser, Stripper, Separator, and Compressor. Each of these components imposes safety conditions over possibly explosive physical quantities (like pressure, compressor speed, temperature, etc.) and together drive the plant towards a profitable state. [10] is a linearized MATLAB simulation of TE which contains 18 PID loops. It embeds process disturbances which give sufficient complexity in emulation of real-world plants. It also embeds alarms relating to unsafe conditions which the designed attacks aim to avoid. In the real-world scenario, we consider that different PLCs will execute different PID loops. Therefore, in our HITL TE testbed, we migrate two such PID loops to a PLC. The PLC runs two cascading PI loops that take in the values from two sensors and control the reactor pressure and purge rate. The hardware used is shown in Fig. 6. The testbed uses a Wago 750-881 PLC which has a 32-bit ARM CPU that runs on a Nucleus RTOS, and a 32KB non-volatile memory. The PLC was programmed in Structured Text. To communicate with the simulation model hosted on the PC, the testbed deploys a Serial-Interface Board (SIB) that is equipped with analog-to-digital (A/D) and digital-to-analog (D/A) converters. Following the guidelines presented in Section 5, the payload operates as:

ICS Sector fingerprinting: Congruent to our threat model, we assume that in the real world, the adversary would be able to infect the general-purpose computer which hosts the HMI (like the one shown in our setup in Fig. 6). We perform the ICS sector fingerprinting by taking a screenshot of the TE HMI and running it through our parallel classification model. Our 49.1 KB HMI-based fingerprinting script refers to the weights for image-based classification, extracts text-strings for text-based classification, and finally utilizes the parallel architecture to classify an image in 3.09 seconds with $\approx$ 98% class prediction probability as *Chemical Sector.*

ICS Process fingerprinting and intelligence extraction: From the HMI, our code traverses the ICS network to identify PLCs controlled by the HMI. Through an FTP connection, the code extracts the control binary from the PLC. Our payload extracts the FBs and strings from the binary, classifying the binary as a *Pressure Control* process. This classification reflects the actual functionality of our TE testbed. ICSREF, which is 383.1 kB in size and included in our payload, identifies the presence of a PID function block and extracts its values. This process takes 30.57 seconds.

Attack: Post plant reconnaissance, the payload designs attack values for pressure control (identified variable). From the process-aware attack point of view, pressure is a good candidate for attack evaluation because it controls the chemical reactions and also may

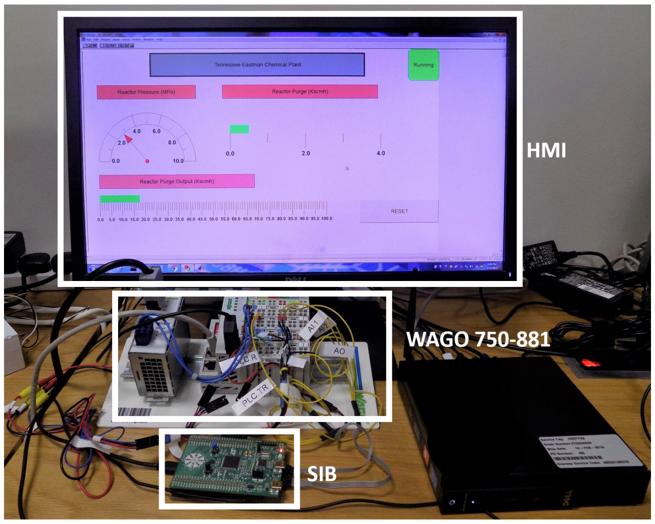

Figure 6: Experimental setup to demonstrate ICS fingerprinting and intelligence collection based on HMI screenshots and PLC binaries. The setup is a Hardware-in-the-loop testbed for Tennessee Eastman Chemical process.

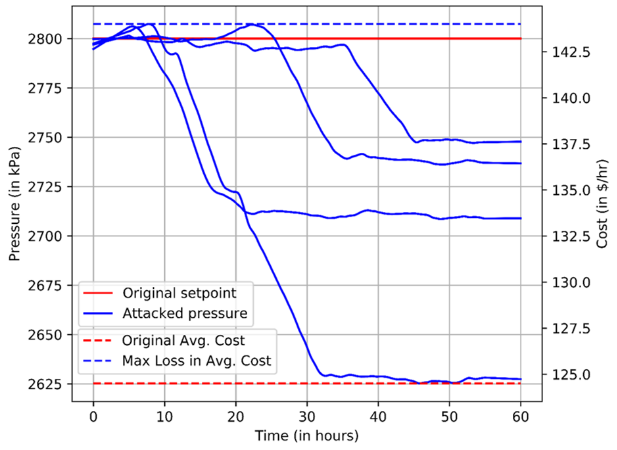

Figure 7: Stable perturbation attack on pressure control.

cause explosions if not controlled within safe limits [22]. We assume that the inherent alarm thresholds of the MATLAB model as the baseline for operational limits of our testbed. For pressure control loop, 3000 kPa is defined as the fail-safe condition. Compromising the *pressure* PLC, we were able to gain all the intelligence regarding operational limits and set-points.

According to the recommendations of subsection 5.4, our automated payload selected a stable perturbation attack setting the lower operational limit to 10% of the original set-point. Under such constraints, the attack could cause a maximum decrease of 280 kPa in pressure from 2800 kPa to 2520 kPa. Fig. 7 describes the results of the experiment. Four successful attack vectors could be generated from principles described in section 5 changing the pressure between 2625 kPa and 2750 kPa. This can result in financial degradation, and it can be seen in the increase in operational cost from $89,000 per month to $103,000 per month. The financial damages caused here attest to our philosophy of attack design; we physically stay within the operational limits but cause damages that accumulate over time. We investigated many loops in TE testbed to find suitable attack vectors (including oscillatory perturbation and physically configurable trigger time) for different alarm thresholds and reported our findings in Appendix.

7 POSSIBLE DEFENSES

Defense against fingerprinting from HMI screenshots: There have been many advances in adversarial machine learning, where undetectable addition of noise to images can change the prediction of an input; These attacks have been demonstrated on facial recognition [39] and text classification using OCR [41]. Recent work on reverting malicious classifications by addition of noise and using majority voting of the fuzzed copies may also be used to confuse fingerprinting mechanisms [38].

Defense against fingerprinting from PLC binaries: Fingerprinting using control binaries depends on the attacker ability to extract PLC binary. A practical method to prevent binary extraction is to disable FTP connections, but the control binary cannot be uploaded remotely anymore. A more realistic solution is to enforce a password-protected FTP connection without default passwords. Semantically, assuming the attacker can have access to the binary, the binary functional blocks and strings can be poisoned by adding random code and text. Adding random code and text is relatively straightforward. We advise against removing descriptive binary strings as important messages in stressful situations can be helpful.

Process-aware defenses: Focusing on defenses applicable to our attacks, one of the most efficient ways to detect malicious manipulations in sensor measurements is by modeling it [27]. Two models that have been used as benchmarks are Auto-regressive (AR) model [18] and Linear Dynamic State-Space (LDS) model [40]. AR oversimplifies ICS processes, sometimes making it difficult to model the sensor readings, and LDS requires extensive ICS domain knowledge to precisely predict a measurement.PASAD [3], a recent work from CCS '18, outperforms the other defense mechanisms by detecting even the most subtle deviations by leveraging the *departure* of ICS dynamics from baseline. However, stealthy adversarial examples on sensor measurements [13] has opened the path to control-theory based attacks leveraging Deep-Learning frameworks to fool state-of-the-art defense mechanisms like PASAD.

8 CONCLUSION

In this work, we consider a constrained threat model where the adversary has no prior knowledge of the target ICS environment. We present a methodology for single-point infiltration through an HMI. For successful identification of the plant, we constructed a dataset using publicly available ICS HMI images and PLC binaries. We then trained several machine learning models to select our final classifier that identifies ICS sector and process. We also use the HMI screenshots and the binaries to extract data to build intelligence for attack design. We leverage control theory to devise generic perturbation-based attacks in ICS. Our results show that, depending on the sector/process/visibility, an adversary can carry-out an end-to-end attack with high accuracy, even with no prior knowledge. This calls for exhaustive security assessment of ICS and the design of robust defense mechanisms that would reduce the attack surface.

ACKNOWLEDGEMENTS

This project was supported by the U.S. Office of Naval Research under Award N00014-15-1-2182, the NYU Abu Dhabi Global PhD Fellowship program, and NYU Abu Dhabi Center for Cyber Security.

RESOURCES

The constructed datasets, and the baseline ML models trained for ICS sector and process fingerprinting are available at https://github.com/momalab/ICS_research_resources.

REFERENCES

[1] S. Amin, X. Litrico, S. Sastry, and A. M. Bayen. 2013. Cyber Security of Water SCADA Systems—Part I: Analysis and Experimentation of Stealthy Deception Attacks. *IEEE Transactions on Control Systems Technology* 21, 5 (2013), 1963–1970.

[2] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen. 2013. Cyber Security of Water SCADA Systems—Part II: Attack Detection Using Enhanced Hydrodynamic Models. *IEEE Transactions on Control Systems Technology* 21, 5 (2013), 1679–1693.

[3] W. Aoudi, M. Iturbe, and M. Almgren. 2018. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *Proceedings of the 2018 ACM CCS (CCS '18)*. 817–831.

[4] W. Ashford. 2018. Social engineering at the heart of critical infrastructure attack. https://www.computerweekly.com/news/252454369/Social-engineering-at-the-heart-of-critical-infrastructure-attack. [Online].

[5] K.J. Astrom and R. M. Murray. 2008. *PID Tuning, Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, USA.

[6] J. Berr. 2017. WannaCry ransomware attack losses could reach $4 billion. https://www.cbsnews.com/news/wannacry-ransomware-attacks-wannacry-virus-losses/. [Online].

[7] E. Byres. 2012. #1 ICS and SCADA Security Myth: Protection by Air Gap. https://www.tofinosecurity.com/blog/1-ics-and-scada-security-myth-protection-air-gap.

[8] A. A. Cárdenas, S. Amin, Z. Lin, Y. Huang, C. Huang, and S. S. Sastry. 2011. Attacks against process control systems: risk assessment, detection, and response. In *AsiaCCS*.

[9] Critical Infrastructure Protection Vigilence. 2011. SCADA Security Evaporates in Texas. https://ciip.wordpress.com/tag/scada-incidents/. [Online].

[10] J.J. Downs and E.F. Vogel. 1993. A plant-wide industrial process control problem. *Computers & Chemical Engineering* 17, 3 (1993), 245 – 255.

[11] Dragos. 2017. TRISIS Malware: Analysis of Safety System Targeted Malware. https://dragos.com/wp-content/uploads/TRISIS-01.pdf. [Online].

[12] N. Falliere, L. O. Murchu, and E. Chien. 2011. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response* (Feb 2011).

[13] C. Feng, T. Li, Z. Zhu, and D. Chana. 2017. A Deep Learning-based Framework for Conducting Stealthy Attacks in Industrial Control Systems. arXiv:cs.CR/1709.06397

[14] David Formby, Preethi Srinivasan, Andrew M. Leonard, Jonathan D. Rogers, and Raheem A. Beyah. 2016. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems. The Internet Society.

[15] L. Garcia, F. Brasser, M. Hazar Cintuglu, A. Sadeghi, O. A. Mohammed, and S. A. Zonouz. 2017. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *NDSS*.

[16] F. Golnaraghi and B. C. Kuo. 2009. *Automatic Control Systems* (9th ed.). Wiley.

[17] N. Govil, A. Agrawal, and N.O. Tippenhauer. 2018. On Ladder Logic Bombs in Industrial Control Systems. Springer International Publishing, 110–126.

[18] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. 2014. Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes. In *Proceedings of the 30th ACSAC (ACSAC '14)*. 126–135.

[19] Y. Huang, M. Esmalifalak, H. Nguyen, R. Zheng, Z. Han, H. Li, and L. Song. 2013. Bad data injection in smart grid: attack and defense mechanisms. *IEEE Communications Magazine* 51, 1 (2013), 27–33.

[20] Kaspersky. 2017. ICS cybersecurity: A view from the field. https://www.kaspersky.com/blog/ics-report-2017/16967/. [Online].

[21] A. Keliris and M. Maniatakos. 2019. ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries. In *NDSS*.

[22] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos, and F. Khorrami. 2016. Machine learning-based defense against process-aware attacks on Industrial Control Systems. In *2016 IEEE International Test Conference (ITC)*.

[23] C. Konstantinou, M. Sazos, and M. Maniatakos. 2016. Attacking the smart grid using public information. In *2016 17th Latin-American Test Symposium (LATS)*. 105–110. https://doi.org/10.1109/LATW.2016.7483348

[24] Kowsari, J. Meimandi, H., Mendu, Barnes, and Brown. 2019. Text Classification Algorithms: A Survey. *Information* 10, 4 (Apr 2019), 150.

[25] M. Krotofil. 2017. Evil Bubbles. https://www.blackhat.com/us-17/briefings/schedule/#evil-bubbles-or-how-to-deliver-attack-payload-via-the-physics-of-the-process-7689. [Online].

[26] M. Krotofil and J Larsen. 2015. rocking the pocket book: Hacking chemical plants for competition and extortion cite. https://www.blackhat.com/docs/us-15/materials/us-15-Krotofil-Rocking-The-Pocket-Book-Hacking-Chemical-Plant-For-Competition-And-Extortion-wp.pdf. [Online].

[27] Marina Krotofil, Jason Larsen, and Dieter Gollmann. 2015. The Process Matters: Ensuring Data Veracity in Cyber-Physical Systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/2714576.2714599

[28] R. M. Lee, M. J. Assante, and T. Conway. 2016. Analysis of the cyber attack on the Ukrainian power grid. *SANS Industrial Control Systems* 23 (2016).

[29] Y. Liu, P. Ning, and M. K. Reiter. 2009. False Data Injection Attacks Against State Estimation in Electric Power Grids. In *Proceedings of the 16th ACM CCS*.

[30] J. Matherly. 2019. SHODAN. https://www.shodan.io/. [Online].

[31] A. P. Mathur and N. O. Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. 31–36.

[32] S. McLaughlin. 2011. On Dynamic Malware Payloads Aimed at Programmable Logic Controllers. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security (HotSec'11)*. USENIX Association, Berkeley, CA, USA, 10–10.

[33] S. McLaughlin and P. McDaniel. 2012. SABOT: Specification-based Payload Generation for Programmable Logic Controllers. In *Proceedings of the 2012 ACM CCS (CCS '12)*. New York, NY, USA, 439–449.

[34] Department of Homeland Security. [n.d.]. Critical Infrastructure Sectors. https://www.dhs.gov/cisa/critical-infrastructure-sectors. [Online].

[35] Prashant Hari Narayan Rajput, Pankaj Rajput, Marios Sazos, and Michail Maniatakos. 2019. Process-Aware Cyberattacks for Thermal Desalination Plants. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (Asia CCS '19)*. Association for Computing Machinery, New York, NY, USA, 441–452. https://doi.org/10.1145/3321705.3329805

[36] SANS Institute. 2014. German Steel Mill Cyber Attack. https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf. [Online].

[37] SANS Institute. 2016. The Impact of Dragonfly Malware on Industrial Control Systems. https://www.sans.org/reading-room/whitepapers/ICS/impact-dragonfly-malware-industrial-control-systems-36672. [Online].

[38] E. Sarkar, Y. Alkindi, and M. Maniatakos. 2020. Backdoor Suppression in Neural Networks using Input Fuzzing and Majority Voting. *IEEE Design Test* (2020), 1–1. https://doi.org/10.1109/MDAT.2020.2968275

[39] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM CCS (CCS '16)*. 1528–1540.

[40] Y. Shoukry, P. Martin, Y. Yona, S. Diggavi, and M. Srivastava. 2015. PyCRA: Physical Challenge-Response Authentication For Active Sensors Under Spoofing Attacks. In *Proceedings of the 22Nd ACM CCS (CCS '15)*. 1004–1015.

[41] C. Song and V.Shmatikov. 2018. Fooling OCR Systems with Adversarial Text Images. arXiv:cs.LG/1802.05385

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958.

[43] Keith A. Stouffer, Joseph A. Falco, and Karen A. Scarfone. 2011. *SP 800-82. Guide to Industrial Control Systems (ICS) Security: Supervisory Control and Data Acquisition (SCADA) Systems, Distributed Control Systems (DCS), and Other Control System Configurations Such As Programmable Logic Controllers (PLC)*. Technical Report. Gaithersburg, MD, United States.

[44] D. I. Urbina, J. Giraldo, A. A. Cardenas, J. Valente, M. Faisal, N. O. Tippenhauer, J. Ruths, R. Candell, and H. Sandberg. 2016. *Survey and new directions for physics-based attack detection in control systems*. US Department of Commerce, NIST.

[45] David I. Urbina, Jairo A. Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1092–1105. https://doi.org/10.1145/2976749.2978388

[46] A.R. Winnicki, M. Krotofil, and D. Gollmann. 2017. Cyber-Physical System Discovery: Reverse Engineering Physical Processes. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security (CPSS '17)*. 3–14.

[47] T. Yardley. 2008. SCADA: issues, vulnerabilities, and future directions. https://www.usenix.org/system/files/login/articles/258-yardley.pdf. [Online].

[48] M. B. Younis and G. Frey. 2006. UML-based Approach for the Re-Engineering of PLC Programs. In *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*. 3691–3696.

[49] S. Zonouz, J. Rrushi, and S. McLaughlin. 2014. Detecting Industrial Control Malware Using Automated PLC Code Analytics. *IEEE Security Privacy* (2014).

APPENDIX

A CONSTRUCTION OF DATASET

In this work, we propose to fingerprint a process on the basis of an HMI screenshot but a collection of HMI screenshots is not available as a public database. To facilitate method of fingerprinting using ML, we collected diverse images belonging to various Critical Infrastructure (CI) sectors. The Cyber + Infrastructure unit of Department of Homeland Security, which is responsible for protecting CI from cyber and physical attacks, enlists 16 different sectors for categorization of functionalities, threats and, vulnerabilities [34]. The subsubsection describes the methodology followed in the collection of dataset.

A.1 Collection of HMI screenshots

We wanted to leverage the data publicly available on the internet. Some vendors advertise their products and share screenshots of their HMIs and some utilities advertise their projects sharing pictures of the installed CI devices and SCADA systems. A variation of the second case was used by Stuxnet to find information about Iranian Nuclear Power Plants. We used search engines like **Google** and **Bing** to find such images. A Python package called google_images_download was used along with chromedriver to download images in bulk. Once a few viable images were found, more images were found using image search option in these search engines. The following search strings were used:

Name of each sector: Each sector was searched by adding keywords like 'scada', 'hmi' and 'human machine interface' using google_images_download package. For example, chemical sector images were searched for with 'chemical hmi scada human machine interface' as the search string. Each <space> in the search string is interpreted as '+' and thus, all the combinations were searched using the api. We set the limit of download to 1000 per sector but many images faced download errors due to broken links, unknown file types or url-errors. We expected a total of 16k images but around 6.3k images were downloaded using this method. This method downloaded the maximum number of images in bulk.

Popular vendor names and their product names: We targeted 5 popular CI vendors: Siemens, General Electric, ABB, Schneider Electric, Yokogawa and their corresponding SCADA softwares to use in search strings like 'SIMATIC WINCC' from Siemens, 'iFix' from General Electric, 'Wonderware' from Invensys (now Schneider Electric). Apart from SCADA, we also searched for products that had HMI interfaces: iRIO PLC from Schneider Electric Telecontrol runs a software called xFlow which has a web interface.

Utilities: We also looked at select Utilities for some countries like 'National Thermal Power Corporation', or 'Saudi Arabia Desalination.'

Random search: standalone search strings like 'Scada', 'HMI', 'Distributed Control System', 'PLC interface' were used for collecting more images.

Using search engines, we downloaded more than 10k images but not all the images were useful. Many images were cartoons, many depicted SCADA/HMI in concept and many were different images of the words 'SCADA' and 'HMI'. We cleaned the dataset for usable images and we found less than 5 % of the images were actually HMI screenshots. The usable dataset after cleaning resulted into 500 images belonging to different sectors.

In our second methodology, we used a Search Engine that finds IoT deices on the internet, **Shodan** [30]. We used Shodan Images, a dedicated API in Shodan that looks for images in IoTs, to search for screenshots in Industrial Control System (ICS) (search string: 'screenshot.label:ics'). As an exhaustive search, we also looked for images running VNC service (search string: 'has_screenshot:true RFB'). Most of the screenshots were those of the login screen and 21 images were usable for constructing our dataset.

A.2 Image Annotation

Only three sectors, Chemical Sector, Water and Waste Water Management Sector, and Energy sector, had more than 100 usable HMI images. Since small number of images in a sector would bias the algorithm extremely, we chose these three sectors for classification. We performed manual labelling of data following the steps listed below:

Visual inspection: Presence of key elements like water tanks, common chemical names, electrical switches and connections are strongly indicative of the sector a process belongs to.

Visiting websites and collecting metadata: Visiting the source of the image reveals more information about the image. Other meta-data used for classifying images were filenames, description, language and location. We also used the meta-data downloaded using google_images_download to help in classification.

Sub components of a sector: Critical infrastructure sectors are themselves composed of smaller processes. For example, power consumption of a Chemical plant may be considered a critical process of its own. In our method of image annotation, we classify a screenshot based on the contents on the screenshot even though it may be a part of another CI sector because the adversary is interested in learning about the current system she was able to hack.

A.3 Image metadata for reconnaissance

This subsection focuses on the information obtained from the images. This is done to exhaustively characterize our dataset. We keep track of the meta-information collected while constructing the dataset. We collected the data downloaded as metadata using google_images_download python library. The 'description' section of the images revealed precise plant information as well. For example, the following is a description of one image: *Stainless Steel HMI Solution for Food and Beverages Industry.* Although this particular image was not used in this work because of resolution issues but similar descriptions, available for images used for machine learning, helped in image annotation.

We also searched for these information on Shodan to see if these images verifiably correspond to IoTs on the internet. For example, if a city is revealed from the metadata with some common CI vendor names, we searched for the combination of city and vendor with ICS ports like 502, 2404 which correspond to two common protocols used in ICS, Modbus and IEC 104. Using metadata from images we could learn more about CIs deployed in various places in the world.

B ICS FINGERPRINTING

B.1 Process classification using HMI screenshots

In Section 4.4.2, we have discussed how ICSREF could be used to fingerprint a particular function block which could be a PLC control loop. In this section we consider the attack scenario where continuous screenshots of HMI could be taken to build a time-series of all the variables that are being monitored in the HMI. Using this time-series, we demonstrate a possible way of automatically fingerprinting the infected loop by considering the time-series as a *signature for that process variable*. Please note, there cannot be a machine learning model that can classify a process variable in this way across all sectors in all plants. Therefore, this methodology of reconnaissance depends on the plant attacked and is not generalizable according to our definition in Section 3. For example, a process variable pressure may have a time series with average 2800 kPa and 1000 kPa for different kinds of ICS plants and we cannot universally make a time-series representative of pressure in this way. Thus, this method of process variable identification is dependent on the attacked plant. In this methodology, the reconnaissance tool identifies common process variable names from the HMI using OCR and builds time-series for them.

As mentioned in the setup, each of the control (PID) loops may be considered to be controlled by a PLC which in turn, may be monitored by the HMI. Here, we collected data from TE model from the inputs of the PID loops emulating collection of data from inputs of the PLC. Each of these inputs are the process variables of the plant considering the feedback value coming to the PLC (Fig. 5). We built time-series for those process variables to train a Naive Bayes Classifier. Our aim was to finger print all the independent loops of the MATLAB model. We found 16 out of 18 loops were directly interacting to the measured variables. We collected data for 60 hours of operation amounting to 900 data points for every PID loop. Further, we split the data into time-series consisting of 30 data-points and calculated the average values of the time-series for the 16 different measured variables. We used 80% of the dataset for training and the rest as test set to evaluate the efficiency of the classifier. Our aim was to identify each of the PID loops with high accuracy using the corresponding time-series. The test accuracy for all the 16 PID loops was 91.67%. Thus, from this machine-learning based reconnaissance, we were able to fingerprint a particular loop, thus, a process variable and further assess the vulnerability of a particular loop towards successful fingerprinting and attack.

B.2 Sector classification using PLC binaries

We perform experiments to show the efficacy of using PLC FB/FBs and strings for **sector classification**. We use the same dataset of binaries. Using FB/FBs, we achieved accuracy of 33% which attested to the fact that a process F/FB cannot be used to fingerprint ICS sector. After data cleansing and string translation, we trained machine learning models with these stings to classify binaries into the sectors shown in Table 6. Our results show that that this method is ineffective for classifying the binary with an accuracy of 33.33%. Since the PLC string-based classification model performed poorly for ICS sector classification, We performed another experiment

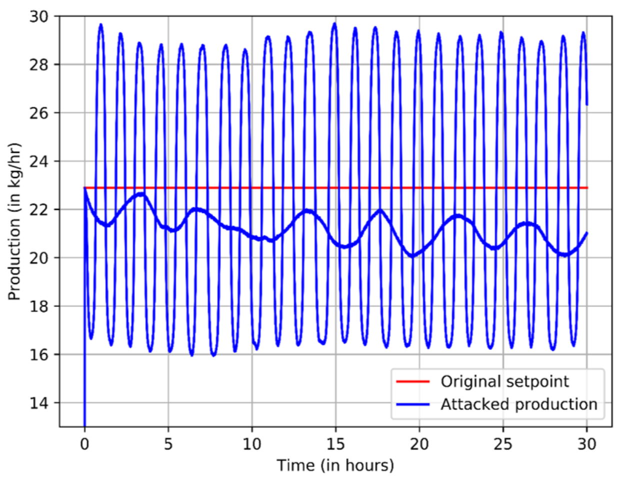

Figure 8: Oscillatory perturbation attack on production.

leveraging transfer-learning to classify the PLC binaries obtained from the field without adding any binary specific information to the machine-learning model. The machine learning model predicted 84.21% of the binaries to belong to the energy sector. On manual analysis we found many binaries contain processes in the energy sector, namely boiler control and temperature control and the accuracy was found to be slightly better. This was a preliminary experiment but this concept may be used for more robust reconnaissance.

B.3 ICS device fingerprinting

ICS attack literature has also focussed on fingerprinting vendors or specific devices to enable attack vectors be tailored according to the device [14]. In this work, we target generalized attack vectors which can be aided with device level fingerprinting for successful infection.

C OTHER PROCESS-AWARE ATTACKS

C.1 Oscillatory perturbation attack

We performed oscillatory perturbations on plant production (Fig. 8). Production was chosen as the process variable for this attack because we wanted to experiment with attacks that are not directly leverage the modelled dynamics. The objective of these stealthier attacks are malicious physical dynamics which are not part of plant performance metrics and thus, are undetectable by any threshold. For our first attack, we choose the alarm thresholds as follows: upper limit 25 kg/hr and lower limit 20 kg/hr. The payload generated an attack that oscillates the production between 23 kg/hr and 20 kg/hr. The second attack is more aggressive and the alarm thresholds in production are chosen to be between 15 kg/hr and 30 kg/hr. This resulted in oscillations of production between 16 kg/hr and 30 kg/hr. The oscillatory perturbations can be further increased with further change in K_P, but then the system would be driven into unstable region of operation (i.e. the exponential increase of oscillations).

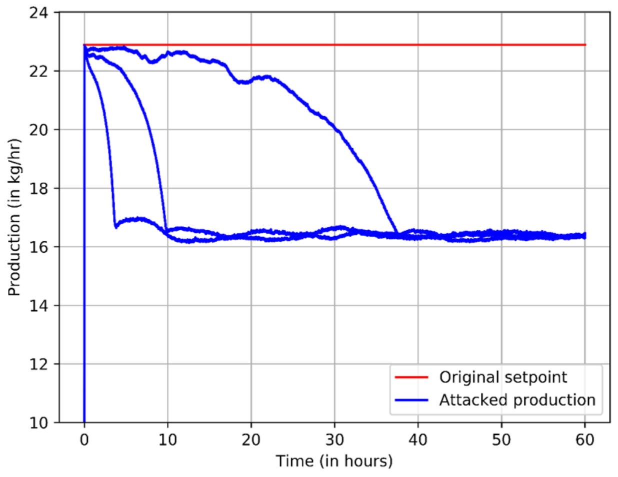

Figure 9: Stable perturbation attacks with physically configurable trigger time.

C.2 Physically configurable attack trigger time

From Fig. 7, we see that the attacked pressure values settle at different time for different payloads. Here we choose another process variable to specifically perform this attack. We performed the stable perturbation attack with physically configurable trigger time on production. We chose production to perform this attack because decrease in production causes direct losses to ICS. The trigger times for such explosive attacks are chosen between 0 and 40 hours. As can be seen from Fig. 9, the payload was able to find various trigger times at 4, 10 and 36 hours after infection. Thus, if an adversary chooses to cause explosive attacks to ICS plants, this completely physical attack may be used to cause damage after he escapes.