CPAC: Securing Critical Infrastructure with Cyber-Physical Access Control

Sriharsha Etigowni
Rutgers University
se260@rutgers.edu

Dave (Jing) Tian
University of Florida
daveti@ufl.edu

Grant Hernandez
University of Florida
grant.hernandez@ufl.edu

Saman Zonouz
Rutgers University
saman.zonouz@rutgers.edu

Kevin Butler
University of Florida
butler@ufl.edu

ABSTRACT

Critical infrastructure such as the power grid has become increasingly complex. The addition of computing elements to traditional physical components increases complexity and hampers insight into how elements in the system interact with each other. The result is an infrastructure where operational mistakes, some of which cannot be distinguished from attacks, are more difficult to prevent and have greater potential impact, such as leaking sensitive information to the operator or attacker. In this paper, we present CPAC, a *cyber-physical access control* solution to manage complexity and mitigate threats in cyber-physical environments, with a focus on the electrical smart grid. CPAC uses information flow analysis based on mathematical models of the physical grid to generate policies enforced through verifiable logic. At the device side, CPAC combines symbolic execution with lightweight dynamic execution monitoring to allow non-intrusive taint analysis on programmable logic controllers in realtime. These components work together to provide a realtime view of all system elements, and allow for more robust and finer-grained protections than any previous solution to securing the grid. We implement a prototype of CPAC using Bachmann PLCs and evaluate several real-world incidents that demonstrate its scalability and effectiveness. The policy checking for a nation-wide grid is less than 150 *ms*, faster than existing solutions. We additionally show that CPAC can analyze potential component failures for arbitrary component failures, far beyond the capabilities of currently deployed systems. CPAC thus provides a solution to secure the modern smart grid from operator mistakes or insider attacks, maintain operational privacy, and support $N-x$ contingencies.

1. INTRODUCTION

Critical national infrastructure has become increasingly complex. For decades, systems such as the power grid were comprised solely of physical, mechanical components that could be reasoned about using classical physics. However, as computing has become increasingly miniaturized and ubiquitous, adding computational resources into these environments becomes not just feasible, but practical and beneficial. In the case of the power grid, adding computing elements allows for essential capabilities such as *state estimation* (i.e., understanding where the power in a grid is flowing at any given time) and *contingency analysis* (i.e., determining whether the grid is resilient to the failure of components within it). The grid exemplifies a *cyber-physical* infrastructure, with data collected from its physical components and processed by algorithms running on computers to provide for accurate and safe monitoring and control. To realize this, modern smart grids make heavy use of programmable logic controllers (PLCs) which act as dedicated embedded systems that change actuators based off of sensor values in a continuous feedback loop.

Malware-based attacks against these infrastructures, such as Stuxnet [30], Havex [51], and Dragonfly [12], have been well studied, and different solutions have been proposed [25, 45]. However, erroneous activity by human operators, whether intentionally or by mistake can have even more due consequences than existing malware attacks. The lack of protections against system misconfigurations can lead to severe consequences. In 2011, a lack of real-time situational awareness and limit protections on transmission lines resulted in a cascading series of power outages, affecting large portions of Arizona, southern California, and northern Mexico, causing 1.5 million customers in these areas to lose power for up to 12 hours [10]. Even worse, malicious activities can also seem to be operation mistakes, such as the coordinated attack on the Ukranian power grid [17]. Moreover, an operator once logged in, usually has a complete view of the whole system, even if the operator is only in charge of a sub area of the system. This unlimited access to system variables and the simple static policy controls for operators demonstrate that cyber-physical infrastructures are unprepared to maintain their safe and secure operation in the face of human mistakes, leaving alone malicious adversaries.

The key takeaway from these episodes is that *insufficient access control* coupled with an *insufficient understanding of the relationship between the control infrastructure and the underlying physical system* leads to vulnerabilities, which can be turned into attacks either by careless operators or malicious adversaries. While past approaches attempt to use information flow analysis for system modeling, have tended to ignore the physical world and miss important inter-dependencies. Moreover, traditional discretionary and mandatory access control mechanisms are often based on manually-generated policy rule sets that do not consider the underlying physics of the grid, and its complexity precludes attempts at formal analysis.

In this paper, we present CPAC, a *cyber-physical access control* framework that enables fine-grained enforcement of context-aware policies in a real-time control system environment. CPAC takes a comprehensive view of both the computing and physical elements comprising the control system, and simultaneously incorporates both continuous physical dynamics i.e mathematical models and discrete computing i.e administrator specified policies into its security monitoring and control calculations. In doing so, we can accept high-level requirements such as “Alice should not [directly or indirectly] manipulate the [power output] for the gener-

ACSAC '16, December 05-09, 2016, Los Angeles, CA, USA

DOI: http://dx.doi.org/10.1145/2991079.2991126

ator G_i" or "Bob should not know about power transformer T_j's failure," and have them enforced as low-level policies that ensure control system constraints are maintained. To generate secure policies for access requests, CPAC implements a layered ensemble of lightweight information flow analysis mechanisms. On the device side, we mark variables within PLC devices to determine data flow, and we infer information flows through the grid using physics-based, inter-component dependencies. Information is visible to operators whose access to read and modify variables is tailored to their particular roles (static polices) and depending on the information flow analysis (dynamic polices). Combining the physics model, information flow analysis on PLCs, and logic-based policy control, we are able to provide finer-grained access control and better situational awareness of the power grid than previous solutions, securing the grid from human mistakes (or insider attacks), maintaining the operation privacy, and supporting $N-x$ contingencies.

Our contributions can be summarized as follows:

- **Physics-based engine:** We demonstrate that by leveraging the underlying mathematical model within a power system, we can analyze information flow by the physics equations and restrict operations that would violate system safety.
- **Information flow analysis:** We introduce a lightweight taint-tracking mechanism into PLCs. The lightweight code instrumentation reports the dynamic control flow used in conjunction with symbolic execution of the PLC code to determine variable taints. This symbolic execution is performed offline ensure minimal performance overhead during PLC code execution.
- **Logic-based policy control:** We introduce a new context-aware policy control using Prolog, where policies are written in logic statements and the querying the permissibility of an operation in the Prolog engine. Combined with the physics engine and information flow analysis, a context-aware policy is able to guarantee the safety and privacy of an operation.
- **Scalability and performance in real-world scenarios:** We model the Polish power grid, consisting of over 2,700 buses, and model three past blackout events within this real-world system setting, demonstrating that CPAC would detect and mitigate all of these problems. CPAC's analysis and policy evaluation can be performed in under 150 *ms*, fast enough that large-scale outages can be prevented. Because CPAC maintains system context, it can manage not only $N-2$ contingency analysis (simultaneous failure of two nodes), but $N-x$ analysis, which is infeasible with existing energy management system (EMS) solutions. CPAC thus provides an effective new means of maintaining robust operation in the face of coordinated cyber attacks.

Section 2 reviews existing EMS solutions and how they fail to withstand operation mistakes or even attacks. Section 3 overviews CPAC's high-level architecture and components, describing its operation within a simple control system. Section 4 explains the physical side information flow analysis. Section 5 describes policy enforcement and Section 6 describes device-level information flow tracking in CPAC. Section 7 describes CPAC's real-world implementations and extensive experimental results. Section 8 reviews related work and Section 9 concludes.

2. ENERGY MANAGEMENT SYSTEMS

An EMS[1] is a collection of computer-aided tools used by operators of electric utility grids to monitor, control, and optimize the performance of generation and transmission systems. As shown in Figure 1, an EMS contains supervisory control and data acquisition (SCADA) functionality, comprising a suite of applications. These include:

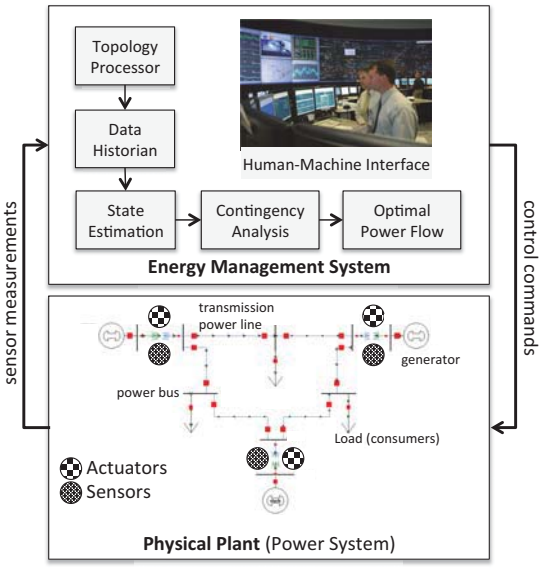


Figure 1: Existing Energy Management Systems

1. A power system topology processor [15] that continuously retains and updates electrical system topology such as branch impedance, loading, connectivity, and circuit breaker status information, with topology details used as input to the state estimation process (detailed below);
2. A data historian (database) [33] that stores sensor measurements and system asset configuration information for later grid analysis and billing;
3. A state estimation system [15] that receives plant sensor measurements and the power system's current topology, and dynamically calculates accurate state of the power system, i.e., voltage, magnitude, and phase angle on each power system bus;
4. Contingency analysis software [6] that performs what-if risk analysis of potential component failures given the power system's current state;
5. Optimal power flow control analysis [6] to calculate optimal feasible power system configuration and actuation parameters for load generation balance (i.e., the generated power should equal the end-users' electricity consumption); and
6. A human-machine interface (HMI) that includes visualization of system parameters for the operators to monitor and modify.

2.1 Existing EMS Solutions

Current EMS solutions [59, 16, 19] are designed to protect smart power grids against accidental component failures, but are limited in the protections they offer. For example, data historians enable local data storage and coarse-grained sharing of bulk system information and sensor measurements, but lack the ability to determine where data entries originate and the understanding of plant physics necessary to capture inter-data entry correlation. As a result, simple mistakes from operators can bring down the whole grid, causing millions of dollars of damage [10, 26]. Similarly, while state estimation modules provide a global view of the power system's state and parameters such as line current and bus voltages, they cannot restrict unprivileged operators from observing sensitive system information, which compromises operation privacy [7, 41]. Furthermore, while power flow solutions [34] have functionality to drive the system away from unsafe states, they do not distinguish among

[1]We discuss the configuration of existing *energy management systems* (EMS) used to control the power grid infrastructure. We also discuss their corresponding limitations and vulnerabilities (Section 2.1). Our discussion is necessarily abbreviated; a comprehensive overview of these issues is presented by Sridhar et al. [56].

operators with different privilege levels. In general, current EMS solutions solely count on correct actions from operators, who usually only need a password to log in the system and are governed by simple policies (if even these exist), and ignore the risk of operational errors or insiders attacks.

Another significant shortcoming in existing EMS solutions is the limited ability to perform contingency analysis. Within North America, power utilities must implement $N-1$ contingency analyses to comply with the North American Electric Reliability Corporation Critical Infrastructure Protection (NERC-CIP) requirements [38]. An $N-1$ analysis determines whether a power system with N components (e.g., generators) can maintain its operation despite any single component failure. However, a coordinated attack against more than one element within the grid or multiple involuntary components failures renders $N-1$ analysis ineffective as occurred in the southwest blackout incident [10]. The state space explosion associated with performing $N-2$ analysis and for larger numbers of component failures makes these analyses infeasible as shown in Section 9.

2.2 Security Threats

The threats we consider are mistakes from careless operators and intentional system manipulation from malicious adversaries, who could be operators or anyone having access to the EMS. Instead of focusing on the authentication of EMS operators, CPAC tries to authorize each operation request from legal users. Note that the whole EMS is trusted, and we assume operators do not have physical access to these machines except through the GUI/CLI terminal provided by the EMS.

Why don't existing access control mechanisms suffice? Existing host and network-based mechanisms that rely strictly on access control have proven to be insufficient in ICS environments, where cyber and physical components interact as a part of the system operation. A shortcoming of existing access control solutions, such as host-based policy enforcement (e.g., SELinux) and network firewalls, are that they ignore the underlying physics of the control systems that they protect[2]. Consequently, implementation of privilege separation and least privilege principles in highly dynamic control system environments become infeasible as access control policies for individual subjects and roles depend on the dynamically changing physical state of the plant. The state of the system may change due to actions by other subjects, e.g., a legitimate power operator on a remote substation computer increases the amount of power generation, or external malicious adversaries, e.g., malware on a remote substation computer opens a power transmission line leaving it out of service. Such incidents change the state of the underlying power system and affect access control policy rules for operators. This increases the risk that subsequent operations (either mistakes or attacks), permitted by a static policy, could compromise system dynamics thus cause damages.

Why do control system safety mechanisms fail to stop operational errors or even attacks? Traditional control system safety mechanisms have been designed to maintain safety only for physical system operations. For example, safety mechanisms in power systems include protection relays and circuit breakers to isolate transmission lines with over-capacity high current flow. Moreover, these mechanisms only consider the physical component involved in the operation rather than a complete system impact of the operation. They are designed to provide reliability and robustness in the case of accidents or harsh environments. They do not, however, take into account a careless operator's mistake, which may crash the whole system, let alone a malicious insider who analyzes the operational changes in a system as it responds to problems, and exploits this behavior to further force the system into an unsafe state [22]. While research into secure control estimation [31, 47, 42] can aid in developing more robust control algorithms, these approaches are largely theoretical and do not consider mistakes from operators or attacks from insiders.

[2]E.g., SELinux is not able to limit the CPU temperature.

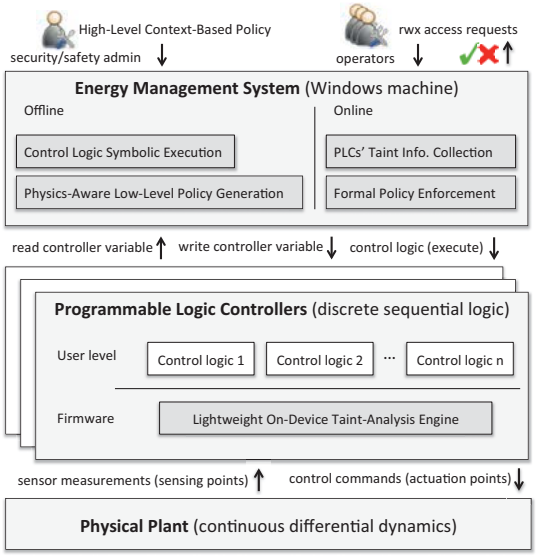


Figure 2: CPAC's High-Level Architecture

3. CPAC ARCHITECTURE OVERVIEW

We provide a high-level overview of CPAC and describe how it addresses the issues raised above. We further detail in Section 3.3 the factors resulting in the 2011 California outage discussed in the introduction [10], and how CPAC could have prevented this failure.

While the guarantees that CPAC provides could be applicable to any cyber-physical infrastructure, we focus on its use as a security protection and access control solution for the smart power grid, with multiple PLCs receiving information and sending data back to an EMS. This setup is illustrated in Figure 2. In practice, each PLC often ships with proprietary engineering software running within the EMS. This is used both offline, for control logic development and execution on the PLC, and online within the EMS, for run-time monitoring and modification of a deployed PLC's variables. The PLC is also connected to the physical plant through lines from sensors within the plant that serve as input, and outgoing wires to actuators within the plant for process control.

3.1 Information tracking

CPAC facilitates security access control in cyber-physical power grid infrastructures and consists of two major components, one residing within the EMS and the other within the PLCs. As the PLC has limited computational resources and hard real-time requirements for processing data, any security solution must minimize performance overhead. To meet these requirements, CPAC offloads most computation from the PLC to a server at the EMS, which communicates with individual PLCs to obtain fine-grained information about device execution. We use offline pre-processing techniques to minimize run-time requirements. Given a new PLC control logic, CPAC symbolically executes the code and determines the source of incoming data for every output variable over all feasible execution paths. This information is stored in a lookup table. Consequently, rather than typical heavyweight run-time taint analysis, CPAC calculates the taint information through lightweight execution path profiling to minimize run-time overhead. CPAC's PLC-based dynamic analysis engine only tracks the execution path of the running control logic (Figure 2). Dynamic tracking of the execution path merely requires run-time monitoring for branch instructions on the PLC, a significant computational reduction compared to dynamic on-device byte-level taint analysis. CPAC uses lightweight control logic instrumentation before every control logic

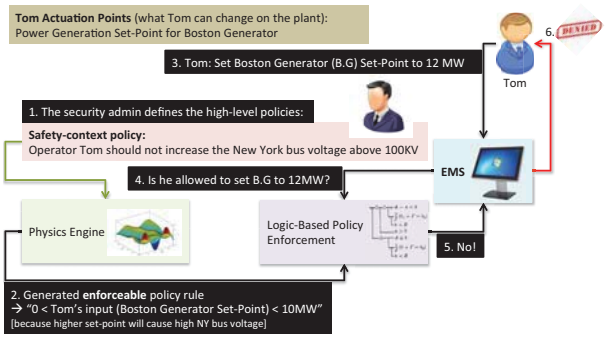


Figure 3: Physics-Aware Access Control

download on the PLC. The PLC-based agent sends collected execution path information to CPAC's EMS-side agent, which consults the symbolic execution lookup table for taint information regarding the affected sensing points.

Apart from the EMS, the device side must also be controlled. For example, Tom (Figure 3) could violate policy by downloading malicious or buggy control logic onto a PLC, or modifying its internal variables through the EMS interface. In either case, enforcing the policy requires analysis at the granularity of individual PLCs to calculate how Tom's actions would affect sensors and actuators throughout the plant. Therefore, before every control logic download to the PLC, CPAC performs an offline symbolic execution of the control logic (Figure 2) and fills out a lookup table where each entry represents an execution path of the control logic, and includes the corresponding path condition along with the symbolic values of the control logic variables at the end of the execution (scan cycle). Upon Tom's variable write or control logic download request, CPAC consults the lookup table for taint information, to determine which actuation points may be affected by Tom's request. Such analysis considers changes throughout the entire power system, relying on information generated based on an information flow analysis performed after every topology server update.

3.2 Defining policies

Consider the workflow shown in Figure 3. An administrator defines a high-level safety-context plant policy, e.g., "Tom [a power operator] should not be able to cause the bus voltage on the New York power transmission line past its capacity 100kV." The policy is defined based on the transmission line's physical limitations, and exceeding the line capacity could potentially cause a line outage, redistributing the downed line's power through its adjacent lines [54] followed by a catastrophic blackout.[3] Intentionally or otherwise, Tom sets the Boston Generator set-point to 12 MW. In doing so, the physical model calculates that the New York transmission line would exceed 110 kV. The model is based on fundamental circuit laws that are dependent on the power system's topology, dynamically updated by the EMS topology processing server. Now the policy enforcement engine evaluates the new set point request and upon determining that granting this request would cause an unsafe state, denies the request, an result that is returned to Tom.

To be practical, CPAC must automatically enforce policies without requiring the administrator to redefine them on every system topology update. CPAC eliminates the need for this involvement through differential equation-based analysis of the EMS plant model, such that the safety policy described above is automatically enforced based on the current system topology.

CPAC's architecture enables policy enforcement to satisfy privacy, safety and regulatory requirements. For instance, a privacy policy may require that some system parameters or sensor measurements about a particular power system incident not be visible to certain operators. Privacy is not only important for preventing the data leaks from certain operators but also to prevent external attacker from knowing additional information which can lead to more effective attack;as an example, the web attack against a Ukraine power plant was caused by the attackers sending commands to open circuit breakers, creating power outage [14]. A safety policy may forbid increasing a line's current beyond capacity. By considering interdependencies between policies from different contexts, CPAC evaluates the whole system to determine the allowed actions.

[3]This situation is exactly what occurred during the Aug. 2003 Northeast blackout, which caused $6 billion in damage [26].

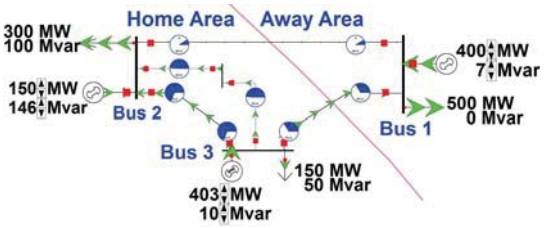
3.3 Case Study: California 2011 Blackout Emulation

As a demonstration of how the multi-layered design of CPAC allows it to maintain a secure environment, we demonstrate how CPAC could protect against a simplified emulation of the California 2011 blackout. For simplicity, we consider an EMS with an underlying four-bus power system (Figure 4a). We assume that the high-level safety and regulation-context policy rules for CPAC's enforcement are defined as follows:

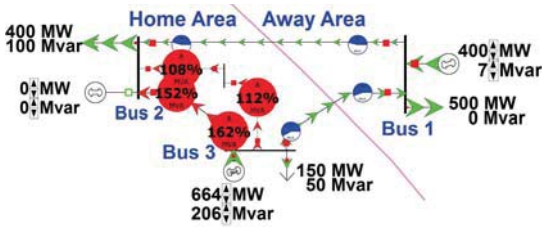
$$\text{Safety policy: } I_l \leq 0.9 \cdot C(l) \quad \forall l \in L$$
$$\text{Regulation policy: } 59Hz \leq f_b \leq 60.5Hz \quad \forall b \in B$$

which requires current I on every transmission line $l \in L$ to be below 90% of the line's physical capacity C, and the AC power frequency f on each bus $b \in B$ to be within the government's mandatory NERC-CIP margins. A security administrator defining high-level policy does not need to define low-level technical details of allowable actions for individual operators, e.g., whether an operator should be allowed to open a particular circuit breaker given the above policy, which is also dependent on the power system's topology and current state. CPAC extracts EMS-enforceable low-level policy rules automatically given the defined high-level policies and the plant topology. Were CPAC deployed, the California incident would not have occurred. Importantly, CPAC denies the operator's mistaken circuit breaker opening, which sparked the blackout. By preventing this action, we prevent a large power system frequency drop in the grid, which would violate the regulation policy. Additionally, opening the circuit breaker would cause line current overflows (Figure 4b), violating the safety policy.

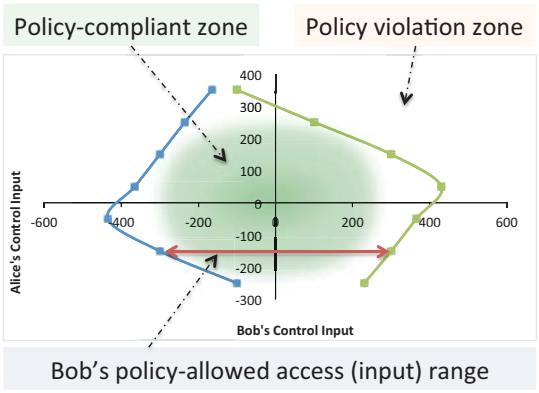
To further clarify CPAC's range-based EMS-enforceable policy generation, consider the safety policy assuming two operators, Alice and Bob, who are in charge of controlling the power generation set-points on buses 2 and 3 respectively in Figure 4a. To apply their control, the operators could either directly change the variables on EMS screens or upload controller programs on the corresponding PLCs. CPAC receives the policy regarding the line currents, and calculates the allowed generation set-point ranges for Alice and Bob's access requests using Kirchhoff's laws shown inFigure 4c, create a calculated policy-compliant region in Alice's and Bob's control input sub-space (the policy compliance zone extends to the edges of the left and right lines). The horizontal and vertical axes represent Bob's and Alice's one-dimensional action space, respectively. Note that the policy-compliant control input range for each operator depends on the system state caused by the other operator's control input value. For instance, if Alice requests a -150 write access to her control input variable on bus 2, Bob's allowed range will be limited to approximately $[-300, 300]$ illustrated by the bidirectional horizontal arrow in Figure 4c. CPAC calculates the plant's policy-compliant region every time the system's state changes since it changes for different plant states. Figure 4d shows the region for a different plant topology when the circuit breaker between buses 2 and 3 is open. In Figure 4d, Bob's allowed input value is constrained to a single value rather than a range if Alice's control input falls between $[280, 350]$, i.e., Bob's actions are constrained by Alice's inputs. It takes approximately 150 *ms* to calculate the region for large-scale plants (e.g., the Polish power grid; which we evaluate in Section 7).



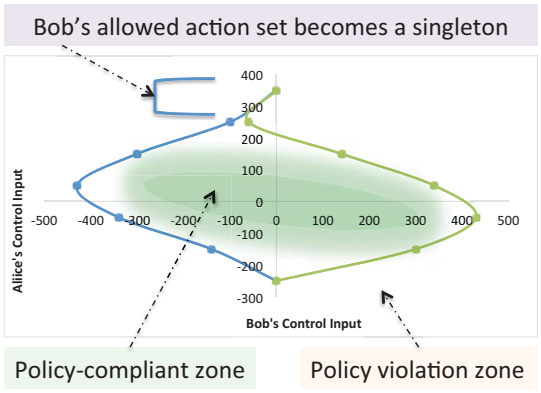
(a) Four-Bus Power System: Normal Operation



(b) California 2011 Incident Emulation



(c) Policy Region for when Relay is Closed



(d) Policy Region for when Relay is Open

Figure 4: Case Study Four-bus Power System and the Operator's Policy-Compliant Control Input Subspaces

4. PHYSICS-BASED INFORMATION FLOW ANALYSIS

CPAC leverages the underlying power system plant's mathematical model to perform physical-side information flow analysis. The power system is a nonlinear electric circuit, where system parameters are correlated according to corresponding equations that represent the physics model. Any perturbation of a particular system parameter causes updates across other parameters such that all values will comply with the equations. We define the physical-side information flow based on such inter-parameter value dependencies. For instance, changing the voltage difference on the two ends of a line with fixed resistance will cause its current update to satisfy the $V = I \cdot R$ relation. CPAC considers this to be an information flow between V and I, because measurement of the line current reveals information about the changes in voltage difference of the two ends.

An n-bus power system's dynamic behavior can be represented by parameterized differential equations [36]:

$$\dot{x} = f(x, u, \lambda) \tag{1}$$

where f is a continuously differentiable function representing the physical plant's dynamic behavior; $x \in \mathcal{R}^{2n-1}$ represents the system state vector that includes the voltage magnitude and phase angles for each bus; $u \in \mathcal{R}^m$ represents the plant's control input vector that could be manipulated by the operators, such as generator set points; λ represents a vector of discrete events that change the plant's topology, and hence its continuous differential equations. The sensor measurements are correlated with the plant state and the operator's control inputs through

$$w = h(x, u) \tag{2}$$

where w is the sensor measurement vector, and h is called the measurement function. CPAC's physical-side information flow analysis leverages the sensitivity investigation of the plant's differential equations given any stable point x_0 and calculates the margin by which each system parameter changes due to physical dependencies if a particular control input is applied to the system. CPAC marks control input actuation points as sources, and every sensing point (measured system parameter) with change margins larger than a predefined threshold ε as the corresponding information flow sinks, ignoring negligible change margins that cannot be practically recognized due to sensor noise. CPAC uses the calculated information about sink parameters to later enforce access control policies. For instance, an operator may be denied applying a particular control input value because she should not be allowed to impact a remote sink parameter beyond a limit or at all based on the safety or confidentiality/privacy context policies.

CPAC determines the allowed value ranges for individual actuation points of the plant that do not violate physics-based policy rules or sensitive parameter changes, e.g., an overloaded transmission line (safety-context policy violation) or a confidential load disclosure (privacy-context policy violation [41]). We call the control input values, beyond which the system enters the policy-violating states, the *boundary points*. The policy boundary margin M is defined as

$$M = |u_* - u_0| \tag{3}$$

where $u_* \in \mathbb{R}^m$ represents a policy boundary point (vector) and $u_0 \in \mathbb{R}^m$ is plant's input at equilibrium or stable state. CPAC uses the difference M to either allow or deny an operator's request for an actuation point change, i.e., requests that exceed the calculated range are denied. CPAC performs this analysis for individual operators separately to calculate their corresponding allowed actuation point value ranges.

CPAC implements the physical information flow analysis through dynamic behavior inspection and sensitivity analysis of the plant around Equation 1's equilibrium state:

$$\begin{aligned} f(x_0 + \Delta x, u_0 + \Delta u, \lambda_0 + \Delta\lambda) \approx f(x_0, u_0, \lambda_0) + \\ + f_x \Delta x + f_u \Delta u + f_\lambda \Delta\lambda \end{aligned} \tag{4}$$

First-order Taylor series expansion of Equation 1 around its equilibrium state is given by Equation 4 which uses the power plant's vector-valued function partial derivatives $f_x = \frac{\partial f}{\partial x}(x_0, u_0, \lambda_0)$, $f_u = \frac{\partial f}{\partial u}(x_0, u_0, \lambda_0)$, and $f_\lambda = \frac{\partial f}{\partial \lambda}(x_0, u_0, \lambda_0)$ which are nonlinear Jaco-

bian matrices given in Figure 13 of Section 9. x_0, u_0 and λ_0 are values at stable or equilibrium state. Assuming that f_x is non-singular, we can reorder Equation 4 as follows

$$\Delta x = -f_x^{-1} f_u \Delta u - f_x^{-1} f_\lambda \Delta\lambda \quad (5)$$

which formulates how the power plant's state changes every time an operator modifies an actuation point. Equation 5 shows the physical-side information flow between the actuation points and the state variables. This is useful for an operator's `write` access control, where the operator request to apply a control input and the policies are defined to prevent the system from entering unauthorized (e.g., unsafe) states. However, actuation point-to-state vector information flow analysis is not sufficient for read access requests, where the operator requests to see a particular sensor measurement, e.g., transmission line current, that is often not the same as a state variable, i.e., power bus voltage magnitude and phase angles. To support read access requests, CPAC implements actuation point-to-sensor measurement information flow analysis to determine how each sensor measurement is affected as the result of a control input application anywhere in the system. Following Equation 2's first-order Taylor expansion around its equlibrium (x_0, u_0), gives us

$$\Delta w = [w_u - w_x f_x^{-1} f_u]\Delta u \quad (6)$$

where the changes in measurements Δw that the operators could have read access request for are calculated as the result of any change in the system Δu. The Jacobian matrices w_u, w_x and f_x are in Figure 13 of Section 9. For more accuracy, second-order Taylor expansion is given in Section 9. These Taylor series expansions around the equilibrium points are used by the system to determine boundary points after perturbing around its equilibrium points and the physics equations are used to determine the information flow between different objects or parameters.

5. LOGICAL POLICY ENFORCEMENT

A key of the EMS is the HMI used by operators to facilitate checking process states, system variables, and control system settings within the physical plant devices. Most software still relies on user name and password input as the sole method of authentication and authorization. Some systems contain elements of role based access control (RBAC) [49], where certain roles are limited to certain operations through the EMS. However, RBAC requires administrators to examine all available operations provided by the EMS, assuming a static policy. Consider a trivial case where Alice cannot view the voltage or current value of a generator G_i, based on a policy that Alice should not know the working status of that generator. Alice can still learn this information by checking the temperature of G_i. These policies become more complicated when the interaction of different operations cannot be detected until run-time. Additionally, support for storing detailed provenance [21] of applications is lacking. Most EMS software provides some logging abilities to record user activity, however, these logs are mainly designed for postmortem analysis rather than policy enforcement, where incorporating provenance could allow additional fine-grained policy controls. For instance, we may want to add restrictions dynamically to operators who tried and failed certain operations over a time window.

5.1 Context-Aware Policy Control

Policy-based access control has been well studied and solutions including MAC, RBAC and capabilities have been applied into commodity computer systems [63, 53, 1]. There are also policy specification languages, such as SPL (Security Policy Language) [50] and RDL (Role Definition Language) [40]. Unfortunately, as we have mentioned above, none of these fits perfectly into the requirements and setting of cyber-physical systems (CPS), which requires, we argue:

- *Information flow control:* Unlike normal policy control systems, whose target are processes, CPS also need to control the information flow of a task/process, guaranteeing no sensitive information leakage[4].
- *Context awareness:* Not only user names, but also time epochs, locations (e.g., IP addresses) and detection of events (e.g., voltage outages) are needed to make policies more useful and practical.
- *Provenance-awareness:* All operations should be logged to allow the use of provenance data to support policies based on user historical behaviors.

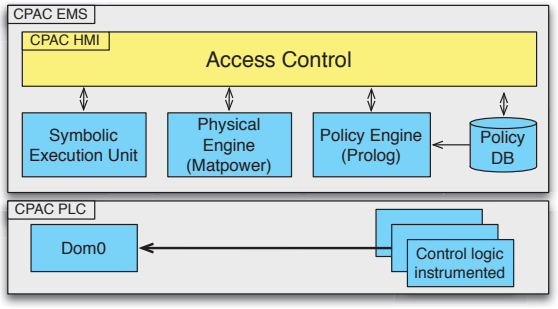


Figure 5: The CPAC EMS/PLC architecture.

To support finer-grained policy, the CPAC EMS consists of a general Modbus [2] transport layer from pvbrowser [13], a HMI access control terminal, the physics engine, the symbolic execution unit, and the policy engine to enforce the policy control and provide provenance support as shown in Figure 5. The transport layer (not shown in the figure) communicates with the PLC via the Modbus protocol (widely supported by most PLCs) over TCP, since we aim for the EMS to be independent of the PLC hardware, while the PLC is running instrumented control logic. The HMI within the EMS provides basic user authentication and accepts operation requests. Both the physical engine and symbolic execution unit provide input for the policy engine, which attempts to authorize operational requests based on policies and adds provenance meta data to these requests for future decision making.

5.2 Policy layers of CPAC

We define three further requirements for a policy control implementation: simplicity of writing policies, correct and potentially formally verifiable policy control logic, and low operational overhead. Under such considerations, we implemented our policy engine using Prolog[5], transforming a policy enforcement query into a logic reasoning process. There are four layers in the policy engine, each in charge of a different policy enforcement task. Each layer is evaluated in order until a layer results in a check failure or there are no more layers to check.

1. **Physical Layer** When CPAC EMS receives an operation permission request from an EMS, it determines whether an operation is physically possible using the analysis described in the previous section. For example, the temperature of CPU should be only readable but not writable.

2. **MAC Layer** This acts as a capability system enforcing which users can do what operations on which variables

[4]Thanks to the symbolic execution unit on the EMS side, and the nature of PLC control logic (less branches comparing to normal x86 binary), taint tracking per task is possible. However, this does not mean the adversary could not learn anything from the running program, e.g., via timing side channels.

[5]Prolog code can be compiled with the native C/C++ code to generate the binary executable, which runs much faster than its interpretive mode.

on a PLC. For instance, Alice is able to read the voltage, the current and the temperature of generator 1, but only to write/change the voltage value. This and the physical layer implement the security features most EMS software share. However, unlike traditional implementations, CPAC counts on logic rules as policies and reasoning as permission checking.

3. **Taint Analysis Layer** This layer uses taint tracking information from the PLC to find information leakage missed by the two previous layers. A trivial example may be that since Bob is not allowed to read the voltage value, he should not be allowed to read the current or temperature either. The taint analysis layer supports both the predefined static taint information (which can be derived from physical modeling), and dynamic tainting provided by the symbolic execution unit and the taint tracking enabled PLC (Section 6 gives more details).

4. **Context/Provenance-Aware Layer** This layer leverages the time, locations, events and provenance to check for permissions (e.g., operations are only allowed during the day from certain IP address for Alice; Bob is not allowed to access variables if a generator fails). As with the above layers, all policies are written in logic rules and facts, and the permissions check is a matter of querying or reasoning.

Both the physical and MAC layers generate static policies, which check for the legitimacy of operations. Passing these two layers proves the validity of an operation request from the traditional access control point of view. The next two layers then try to refute the request using the dynamic tainting information and current running context. Note that CPAC does not try to blacklist all possible illegal operations, number of which may be infinite. Instead, CPAC enables system administrators to retrospect a legitimate operation request in a rich context.

5.3 Formal description of CPAC

To grant permission for an operation, the EMS submits a query to the logic rule *cpac_granted*, which is defined with seven arguments $\{\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}, \mathcal{V}\}$, representing timestamps, locations, users, operations, PLC variable names, new values (if written) and the current value of all the PLC variables respectively, as shown below. Note that $\mathcal{V}$ could be viewed as a global variable, whose value is visible to all rules in the Prolog engine, even though it may not appear in each logic rule. It represents values of all variables available on the PLC when the query is submitted.

$$cpac\_granted(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}, \mathcal{V}) \leftarrow physical\_granted(\mathcal{N}, I, \mathcal{W}) \wedge mac\_granted(\mathcal{U}, I, \mathcal{N}) \wedge taint\_granted(\mathcal{U}, I, \mathcal{N}) \wedge context\_granted(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}).$$

physical_granted grants the permission if the I/O operation is read. Otherwise, it checks if the variable in PLC is writable and if the new value to be written is in the legal range. Note that this layer tries to check the permission from the point of PLC's constrains without considering any other policies. The system administrator is responsible for providing legal ranges for all variables based on specifications, and writing them in the format of Prolog facts (e.g., $svi(voltage, 0, 10, rw)$ shows the name of the variable, the minimum value, the maximum value, and the possible I/O operations), which can be used by the *in_range* rule (and other rules) directly.

$$physical\_granted(\mathcal{N}, I, \mathcal{W}) \leftarrow read(I) \vee (write(I) \wedge writable(\mathcal{N}) \wedge in\_range(\mathcal{W})).$$

mac_granted grants the I/O operation based on the user's capabilities. For all the variables exported by the PLC, the system administrator should assign different permissions to different users. This layer implements the general access control applied by most EMS systems. Within the layered access control structure in CPAC, *physical_granted* comes first. This means even if the operation would be allowed by *mac_granted* through the user access control policy, it may be denied based on rules defined by the physics layer. The layered approach in CPAC thus provides more modular access policies. To add a new user or modify existing policies, the system administrator only needs to create or modify the corresponding Prolog facts, such as *cap_read(bob,[current])* (giving bob the permission to read variable *current* (only)).

$$mac\_granted(\mathcal{U}, I, \mathcal{N}) \leftarrow (read(I) \wedge cap\_read(\mathcal{U}, \mathcal{N})) \vee (write(I) \wedge cap\_write(\mathcal{U}, \mathcal{N})).$$

taint_granted determines whether the target variable could be tainted by some other variables not visible to this user, and rejects the operation accordingly to avoid data leakage. CPAC supports both taint analysis by writing the static taint rules directly and the dynamic taint tracking provided by the symbolic execution unit and PLC during the run time. This layer uncovers missing policies not easily found in the traditional access control implementations. $\mathcal{Z}$ stands for all the variables visible to the EMS side (same as the $\mathcal{N}$ used in *cpac_granted*, such as the temperature and current). Both the static and dynamic rules share similar Prolog construction, *taint_X(z1, z2)*, meaning variable *z1* tainted by variable *z2*. As shown below, if variable *z2* cannot be accessed by this user, the request for accessing variable *z1* would be rejected.

$$taint\_granted(\mathcal{U}, I, \mathcal{N}) \leftarrow \forall z \in \mathcal{Z}: ((\neg taint\_static(\mathcal{N}, I, z)) \vee (taint\_static(\mathcal{N}, I, z) \wedge cap\_read(\mathcal{U}, z))) \wedge ((\neg taint\_dynamic(\mathcal{N}, I, z)) \vee (taint\_dynamic(\mathcal{N}, I, z) \wedge cap\_read(\mathcal{U}, z))).$$

context_granted leverages contextual information to help system administrators write polices fitting into their specific domains, (e.g., the power grid). To simplify the rule/policy writings, we introduce an event-driven reasoning framework and fix the default action of policies to be operation blocking. The final permission granting is then the conjunction of negations of all the *blocking* rules, which are *context_denied_X*, where *X* is an integer used to differentiate all these Prolog rules.

$$context\_granted(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}) \leftarrow (\neg context\_denied\_0(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W})) \wedge (\neg context\_denied\_1(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W})) \wedge (...).$$

All *blocking* rules are event-driven and follow the same construction. Note that all events should be predefined by the system administrator based on domain knowledge. One simple example is *event_g0_failure(V) :- g0_power=<0.*[6]. With event $\mathcal{E}$ defined, the *blocking* rule is defined as below. Given user *U*'s access request on variable *N*, if event *E* happens, and the corresponding rule *context_policy_block* contains *N* in its blocking list *B*, the context layer will deny the request.

$$context\_denied\_X(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}) \leftarrow event\_E \wedge contex\_policy\_block(\mathcal{T}, \mathcal{L}, \mathcal{E}, \mathcal{U}, \mathcal{B}) \wedge member(N, B).$$

Below we demonstrate a real code snippet within CPAC. We choose a complicated policy to demonstrate the ease of policy writing once the corresponding event is predefined by the system administrator. This rule states that for any condition, once generator 0 (`g0`) fails, the temperature value of that generator should not be visible to the operator 'dave'[7].

[6]Note that in this Prolog rule, argument *V* is not used at all, since this event call be determined solely by checking the power of the generator. Complex events can have multiple arguments and take full usage of them.

[7]In Prolog, '_' is wildcard, meaning that the value of that variable does not matter.

```
context_policy_block(_,_,g0_failure,dave,[temp0]).
context_denied_0(T,L,U,I,N,W) :-
event_g0_failure(_),
context_policy_block(_,_,g0_failure,U,B),
member(N,B).
```

To support provenance both for forensic analysis and run-time provenance-based policy enforcement (e.g., an event related with user's previous operation history), CPAC records each operation request from the EMS side, either granted or denied, both in a standalone provenance logging file and the Prolog engine as a 'fact', using the unified format:

$$provenance(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}, \mathcal{R}, \mathcal{V}).$$

Here $\mathcal{R}$ stands for the final result for this operation request (granted (g) or denied (d)) and $\mathcal{W}$ is reused to hold the return value for read operations, as well as the new value for write operations. Other variables are the same as the ones in the rule *cpac_granted*. A concrete example is shown below, where user `dave`'s request to read variable `temp0` from IP address `10.10.10.10` at time `2015071411550` was granted, with all other variable values at that time dumped in the list.

```
provenance(20150714115507, 10-10-10-10, dave, r,
temp0, 3000, g, [3000,4000,5,40,38,17,15]).
```

With more provenance added into the CPAC EMS Prolog engine, making provenance-aware polices is possible. For example, users with more than 10 denials within an hour could be blocked, as the user account may have been compromised. Also any unseen IP address used by a certain user could be blocked, which actually implements a naive intrusion detection mechanism. Since all provenance is also saved into a standalone logging file, this file can be loaded into the Prolog engine every time the EMS is restarted. With the help of the Prolog interpreter, one could submit queries, such as "who read the variable `temp0` in the past but was denied" (`provenance(_,_,X,r,temp0,_,d,_).`), and Prolog would find all users satisfying the query.

5.4 Trade-offs

Besides all the desired requirements of implementing a policy enforcement component mentioned before, one of the biggest concerns using a logic programming language to write policies is how easy it would be for system administrators to use. As shown before, the logic reasoning framework is already provided, as well as some sample constructions. Other than the definition of events, which gets complicated when the event itself is complex, CPAC EMS expects only simple inputs from users, such as the range of certain variable, and permissions for certain user. In general, we believe the advantages of using Prolog to implement the policy control outweigh the impediment of writing simple Prolog facts.

Another concern comes from the scalability issue when writing policies for real-world complex systems, e.g., a nation-wide power grid system. As we will show in Section 7, the final Prolog policy file used to simulate the Polish power grid containing more than 2700 buses is almost 1 MB, with 25K lines, among which, more than 24K lines are simple Prolog facts mentioned before and generated by a Python script. While most policies can be generated automatically given the system specification, the definitions of events used by the context-aware policies need human intervention with specific domain knowledge. As shown in Figure 12, writing in Prolog is straightforward and does not provide extra obstacles comparing to writing in other policy languages[8].

[8]In reality, such a large-scale system is usually divided into multiple sub- areas, which are maintained by different system administrators. A global policy can be defined using predefined local events from sub areas rather than dealing with thousands of variables directly.

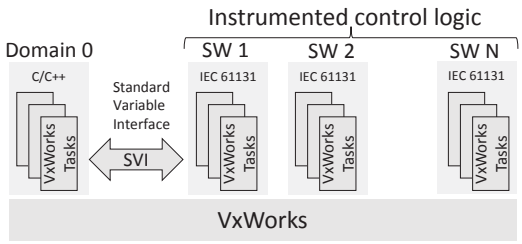


Figure 6: Domain 0 along with instrumented control logic

6. DEVICE LEVEL INFORMATION FLOW

CPAC deploys its dynamic information flow analysis through a lightweight instrumentation of the VxWorks real-time operating system, which is widely used within industrial control systems (40% of the market share [11]) and mission-critical settings, such as the Mars Curiosity rover [39]. To support such an environment, CPAC must meet two requirements: *i)* very low run-time performance overhead to prevent missed real-time deadlines for PLC-level workflows; and *ii)* very high taint analysis accuracy to prevent possibly fatal safety hazards.

Traditional information flow techniques for x86 architectures using byte-by-byte data flow tracking solutions cannot be applied due to their unacceptable run-time execution slowdown (e.g., 6X by BitBlaze [55]). Several proposals considered how to speed up dynamic taint analysis on resource-constrained devices [29] and how to extract semantic information [8]. While useful for desktop and smartphone applications, these solutions will not meet real-time deadlines (e.g., 15% overhead by [29]). As an alternative to dynamic taint analysis, static techniques remove the run-time performance problem. However, the strict accuracy requirements for control system applications limit their practicality significantly due to their well-known high false positive rates. A false positive taint analysis outcome in CPAC could potentially lead to denying an operator's legitimate access request to take care of an emergency situation. Consequently, neither dynamic nor static techniques by themselves can address both the above-mentioned control system requirements completely.

We use a hybrid approach with CPAC, leveraging specific features of the PLC execution logic and VxWorks architecture to ensure high taint analysis accuracy and minimize operational intrusiveness. In practice, PLC controller programs include far fewer branch instructions than x86 binaries. This facilitates comprehensive offline analysis of the controller before its launch time due to less path explosion. CPAC implements the PLC code symbolic execution as discussed in [5] on its EMS-side modules. CPAC uses the symbolic values to obtain the taint information for all program variables including the outputs, depending on which input values every program variable is tainted by. It creates a taint look-up table and uses it to speed up its run-time performance remarkably. Moreover, CPAC's run-time modules on the PLC do not have to implement full dynamic taint analysis, but instead just profile only the execution paths taken by the control logic. The small number of branch instructions in typical control logic that need profiling further motivates CPAC's approach. The EMS receives the execution path profiles and consults the look-up table to determine the taint information needed for the access policy enforcement.

Figure 6 shows CPAC's device-level module on a Bachmann MX231 controller running VxWorks. The main module (so-called *Domain 0*) is written in C/C++ and inserted into the PLC's kernel as a *.m* binary file. VxWorks allows several control logic modules to run simultaneously on the PLC. Before every control logic is uploaded into the PLC by the EMS, CPAC instruments the program with a lightweight inline reference monitor using the PLC instructions (IEC61131 [37]) to profile the control flow. Domain 0 dynamically collects control flow information from the running controller

programs through the VxWorks standard variable and module interfaces that enable on-device remote procedure calls [37]. It then transfers the information to EMS modules over Modbus to perform taint analysis and policy enforcement. To minimize overhead, we only collect control flow information on demand, i.e., only if there is a corresponding access request from the EMS.

7. EVALUATIONS

We evaluated CPAC on two power plants, the four-bus system (Figure 4a) and Poland's publicly available power grid with over 2,700 buses (>2,800 transmission lines) since other networks are not publicly available. We extended the open source pvbrowser v4.7.9 [13] EMS (2-core Intel 2.40GHz; 4GB memory) with our logic-based access control engine. CPAC's power system analysis module uses MatPower [66]. The PLC-based taint analysis and the symbolic execution implementations in CPAC are specifically deployed for the Wind River VxWorks operating system v5.5 running on a Bachmann MX231-Controller PLC with a GIO 212 IO modules. We then designed a set of experiments to verify whether CPAC can be useful and practical in real-world scenarios by answering the following questions empirically:

1. How accurately would CPAC prevent past real-world control system and power grid severe incidents?
2. How efficiently does CPAC perform the PLC-based taint-analysis, physical-side information flow analysis, and EMS-side logic-based policy enforcement?
3. How well does CPAC scale up for large-scale real-world control systems and power grid infrastructures?

7.1 Case Studies

To answer these questions, we validated CPAC's functionality and performance across six use cases. The first three scenarios are based on the four-bus power system; they are derived based on our practical experience and interactions with power utilities to highlight CPAC's capabilities in a typical power grid infrastructure. We assume there are two operators: Bob, a control operator on Bus 2, and Alice, a maintenance operator for the home area on the grid. Figure 4a shows the power system and its two areas separated by a line (the left area is the home area). The following scenarios list the policies for Alice and Bob in different contexts. Case A is below for intuition and the other two scenarios on four-bus power systems (case B and case C) are described in Section 9, while we directly discuss the real world scenarios which occurred in the past.

Case A: Read access control for crucial plant values. Alice, as the maintenance operator, requests to see the real-time transient power output of the generator on Bus 2. The value represents a PLC variable within a droop control logic [28] that controls the generator's power output through its governor.

Source of incident: A lack of enforcing confidentiality over sensitive control data.

Required access control policy: Only control operators are allowed to see sensitive plant control values (defined as generators' real-time frequencies).

Effects of CPAC *deployment:* CPAC denies Alice's request due to the potential for sensitive data disclosure. CPAC's PLC-based information flow analysis marks the target variable tainted by the incoming frequency measurements, which is not readable by Alice. The droop control correlates generator frequency and output power such that knowledge of one value could be used to infer the other.

The next three scenarios are based on real-world power grid incidents that had large-scale effects on millions of power grid customers, some of which were international in scope. Though N-1 contingency was enforced, due to cascading failures these events occurred. We evaluate how CPAC could have prevented these incidents by simulating their effects on the real-world model of Poland's entire power grid interconnect, consisting of over 2,700 buses. This will demonstrate CPAC's scalability to national-scale grid environments.

Case D: Southwest 2011 blackout. The Southwest (California) blackout affected 7M people in California, Arizona, and Mexico, which we describe in detail in Section 3.3. The reports from Federal energy regulatory commission (FERC) showed that "the system was not in an NERC-CIP N-1 compliant state. Utilities are required to operate the system so that the malfunction of one component can not cause instability, separation, or cascading" [10].

Source of incident: Human error from an operator violating compliance with NERC-CIP N-1 contingency regulations.

Required access control policy: No operator may issue a control command that puts the grid in a state that violates the CIP N-1 requirements.

Effects of CPAC *deployment:* Figure 8 shows the line current on bus 18 before and after the operator opened a relay on a different line. CPAC speculatively calculates the potential global impact of the operator's action and denies the action, as it would lead to an unacceptable current flow on the line that violates the NERC-CIP N-1 requirements.

Case E: Florida 2008 grid blackout. The Florida Power and Light (FPL) Company reported a widespread grid blackout occurring at the Flagami substation in west Miami as a field engineer was diagnosing a switch that had malfunctioned. Contrary to standard procedures, the engineer disabled two levels of relay protection. Because both levels of protection had been removed, the arc that resulted from the fault caused an outage that spread through the grid as power plants and transmission lines tripped off-line to protect themselves. "Standard procedures [unenforced] do not permit the simultaneous removal of both levels of protection," the utility wrote [9].

Source of incident: Human error from an operator disabling internal redundancy protections, causing a maintenance operation to disrupt the grid's real-time operation.

Required access control policy: Operators must not remove both levels of relay protection simultaneously.

Effects of CPAC *deployment:* CPAC denied the operator's second protection removal request due to the policy. Additionally, it overloads the most of the power components (Figure 7). Bus #2,392 is overloaded to 661%, which would quickly cause a line outage and damage neighboring assets due to the extremely high current.

Case F: Colombia 2008 total blackout. Colombia suffered a total blackout affecting 25 million people due to human error at the 230 KV Torca substation. An operator at the substation did not follow the correct (but unenforced) sequence of maneuvers when transferring circuits from one busbar to another within a substation before a scheduled maintenance task. The wrong maneuver overloaded the inter-bus breaker, and the breaker malfunction de-energized the whole Torca substation, igniting a cascade of events that brought down the entire Colombian electric power system.

Source of incident: Human error from an operator not correctly following substation interlocking procedures, and lack of enforcement to ensure these procedures are followed.

Required access control policy: The maintenance operator's sequence of busbar interlocking actions must not overload any inter-bus breaker.

Effects of CPAC *deployment:* CPAC denied the operator's action request as it violates the interlocking requirements. Figure 9 shows the substation-level busbar configuration where the top and bottom busbars are connected to neighboring substations, and should never be disconnected from each other. The figures shows the set of already opened breakers, the ones that the operators could open, and the breakers that are prevented from opening, resulting in an operator deny action by CPAC as they separate the two main busbars affecting the electricity grid globally.

7.2 Performance

We measured CPAC's performance for all six scenarios. Table 1 shows the Prolog engine's execution time for scenarios *a-c* averaged over 20 runs. CPAC takes under 0.3 *ms* to process the access request and render a policy decision. This quick processing time

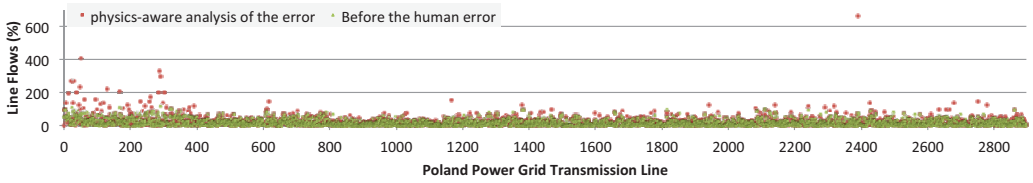


Figure 7: The system capacity overload state in case E (Section 7.1). Note that one line has been overloaded to 661% of it allowable current, a situation that CPAC would prevent from reaching.

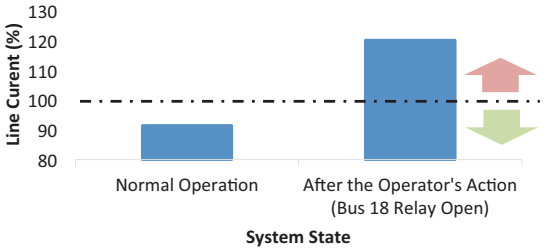


Figure 8: Southwest Blackout Prevention using CPAC. On evaluating the effects of line current on bus 18 after opening the relay. CPAC determines line would be overloaded and prevents the action.

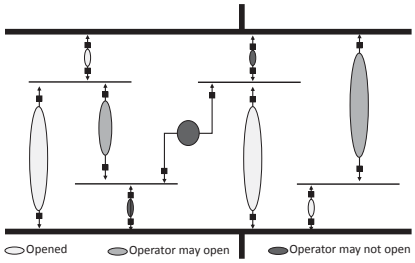


Figure 9: Columbian Blackout Prevention via CPAC. On evaluating the effects of opening the critical relays after few relays are opened. CPAC determines line would be overloaded and prevents the action.

is due in large part to our optimized implementation, where we compiled the logic into assembly using the `gplc` compiler. Table 2 shows the corresponding overhead for the domain 0 to launch taint tracking within the PLC. Most taint information collection could be done within 100 *ms*. As domain 0 is implemented as a standalone kernel module with the lowest priority, we have minimized the impact of domain 0 on other PLC tasks. On the EMS side, there are 30 power system variables in scenarios *a-c* that an operator may be able to see based on policy. As Table 3 shows, CPAC's EMS modules completes all these scenarios within 40 *ms*. Given the general EMS OS overhead and transmission delays (e.g., the 5 minute time requirements by NERC for EMS-side contingency analyses [46]), CPAC's overhead will be minimal to operators. Note that CPAC's EMS modules include the physics engine, the Modbus transport library and the Prolog policy engine.

We measured scenarios *d-f* using the topology of the entire Polish power system, comprising over 2,700 buses. Table 4 shows the general overhead of CPAC's physics engine with these real-world cases. The physics engine is able to finish the forward analysis within approximately 100 *ms*. The result was computed using MATLAB and will likely be even faster if the engine is developed in C/C++. Table 5 shows the overhead of the Prolog policy engine, reasoning about 1,000 simultaneous variables. For the three cases, the Prolog engine completed policy analysis in approximately 15 *ms*, due to our compilation of logic into native assembly. The overhead of the full analysis (without the overhead of user operations and network transmission delay) is within 150 *ms* (100 *ms* from physics engine using MATLAB, 15 *ms* from Prolog policy engine, 30 *ms* from EMS).

Scenario	Min	Avg	Max	Mdev
(a)	125.0	154.9	205.0	21.1
(b)	147.0	186.5	235.0	21.9
(c)	176.0	214.2	280.0	29.8

Table 1: Prolog Micro-Benchmark (*us*).

Scenario	Min	Avg	Max	Mdev
(a)	90.909	96.871	100.200	3.974
(b)	94.787	97.711	99.338	1.949
(c)	90.909	96.693	99.668	3.856

Table 2: Domain 0 and instrumented taint (*ms*).

Scenario	Min	Avg	Max	Mdev
(a)	30.961	31.376	33.991	0.600
(b)	30.933	31.571	32.976	0.601
(c)	29.979	30.442	32.994	0.601

Table 3: EMS Macro-Benchmark (*ms*).

7.3 Scalability: NERC-CIP $N-x$ Compliance

The state-of-the-art NERC-CIP v5 standards[9] protect the power grids against single component malfunctions. However, extensive research [23] has shown the insufficiency of single failure consideration because of increasing complexity of existing smart grids, and more importantly, the possibility of cyber attacks with (automated) subsequent component exploitations. Up to now, guaranteed $N-x$ compliance has not been scalable or feasible in practice. The main reason is that, to fully support $N-x$ contingencies, existing systems must analyze

$$\sum_{i=1}^{x}\binom{N}{i} = N + \frac{N(N-1)}{2} + \cdots + \frac{N!}{x!(N-x)!} \tag{7}$$

different contingencies that each require independent full solution of the power system. Continuing along these lines, one could show that for k simultaneous outages, $O(N^{x+1})$ power flow solutions[10] are required to process the contingency list. For practical power systems, the number of lines tends to scale linearly with the number of buses B in the system ($N \in [B, 1.5 \cdot B]$). $N-x$ compliance thus requires $O(B^{x+1})$ power flow solutions. In the Polish system, where $B = 2,746$, $N-2$ and $N-3$ compliance require $> 3.7M$ and $> 3.4B$ contingency considerations, respectively. Figure 10 shows

[9] Available at http://www.nerc.com/pa/CI/Pages/Transition-Program.aspx

[10] Intuitively, the time complexity of $\binom{N}{i}$ is $O(N^i)$, and the geometric series as the result of Equation 7 grows with the order to $O(N^{x+1})$.

Scenario	Min	Avg	Max	Mdev
(d)	102.048	102.945	104.413	0.653
(e)	100.982	101.571	102.4825	0.644
(f)	97.626	98.116	98.886	0.285

Table 4: Physics engine Macro-Benchmark (*ms*).

Scenario	Min	Avg	Max	Mdev
(d)	8.000	14.750	19.000	2.175
(e)	8.000	14.600	17.000	2.080
(f)	8.000	15.250	20.000	1.600

Table 5: Prolog Macro-Benchmark (*ms*).

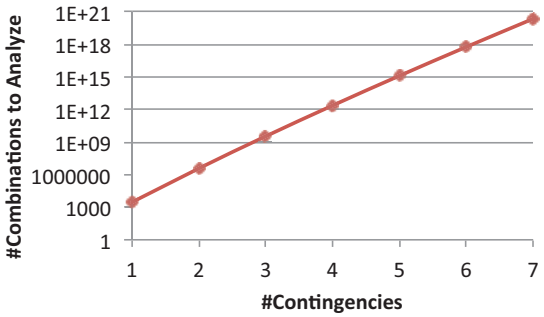


Figure 10: N-x Contingency Analysis Complexity

the results for different number of contingencies. Each contingency takes approximately 2.4 seconds to complete, and power utilities mostly run contingency analysis procedures every 5 minutes. Consequently, traditional methods do not scale up to existing strict requirements and complex grid infrastructures. Several recent efforts attempt to provide $N-2$ contingency analysis support [27, 67]; however, they are not exhaustive, and instead selectively choose and analyze particular contingencies. Consequently, the previous work may miss a contingency that may occur in practice, resulting in incorrect NERC-CIP compliance assurance. Additionally, they do not consider multiple (more than two) subsets of contingencies, i.e., they miss a combination of small contingencies that collectively contribute to a large-scale power grid blackout. *None of the traditional solutions can handle this intractable search space.* CPAC takes an alternative approach that enables $N-x$ contingency analysis even in large-scale systems. Traditional contingency analysis techniques are offline, and need to complete their analysis before any incident occurs or is about to happen. CPAC's policy enforcement framework instead takes a run-time approach analyzing any sequence of incidents before it determines whether they violate requirements. In case of a violation, CPAC denies the request and prevents the system from entering an unsafe state.

8. RELATED WORK

Control system safety. Stouffer et al. [57] present a series of NIST guideline security architectures for the industrial control systems that cover supervisory control and data acquisition systems, distributed control systems, and PLCs. Such guidelines are also used in the energy industry [60, 44]. It has, however, been argued that compliance with these standards can lead to a false sense of security [62, 48]. There have also been efforts to build novel security mechanisms for control systems. Mohan et al. [43] introduced a monitor that dynamically checks plant behavior safety. A similar approach using model based intrusion detection was proposed in [24]. Goble [35] introduce mathematical analysis techniques to quantitatively evaluate aspects of a control system such as safety and reliability, including PLC devices. However, the proposed solution focuses mainly on accidental failures and does not investigate malicious actions.

Access control. Most of the control systems, nowadays, rely on network access control [3], and host-based user authentication to protect against unauthorized plant monitoring and control activities. Additionally, PLC and HMI vendors themselves have included some rudimentary security measures into their solutions. Based on market data by Schwartz et al.[52], we studied the security measures used by PLCs accounting for 74% of market share. This included PLCs from Siemens (31%), Rockwell (22%), Mitsubishi Electric (13%), and Schneider Electric (8%). We found that all four vendors use only password authorization, typically with a single privilege level. Furthermore, password authentication measure can be disabled in all four systems. Recently, more access control capabilities have been added to HMI engineering software. For instance, certain Siemens systems, e.g., SIMATIC STEP 7 TIA Portal [18] use client-side authentication for individual IDE projects. Additionally, recent device fingerprinting mechanisms (e.g., [32]) facilitate deployment of higher level access control functionalities such as CPAC in control systems. Almost none of the existing control system access control solutions take into consideration the physical dynamics of the plant while defining or enforcing the policies. This allows the attacker to completely bypass authentication by exploiting the physical system's dynamics and inter-component interdependencies to disclose sensitive measurements and manipulate critical plant actuation points.

Information flow analysis. Many existing solutions have proposed information flow control [65, 61] and dynamic taint analyzers [20] for general-purpose computing systems, smartphones [29] and embedded devices [64]. However, they have almost never been used in real-world control systems, because of *i)* their high run-time performance overheads limiting their deployability for safety-critical real-time settings and *ii)* insufficient accuracy due to fully ignoring the physical-side information flows. Existing control system data historians [33] within energy management systems [4] provide a bulk databases-level offline information flow control between large power system areas (control centers). Such coarse-grained solutions *i)* do not support dynamic and/or fine-grained information flow control; *ii)* often result in inflexible architectures, i.e., too permissive (allow data exchange between two control centers) or restrictive (no database exchange allowed); and *iv)* completely miss the physical dependencies between various database entries within and across the control centers.

9. CONCLUSIONS

We present CPAC, a cyber-physical access control solution to protect industrial control systems against operation mistakes and insider attacks. CPAC implements lightweight on-device and mathematically sound physical-side information flow analyses to maintain a complete system view. It uses physical system model, information flow tracking, and logic-based context-aware policies to stop operations which could harm the whole system or leak sensitive information to malicious insiders. Our experimental results with CPAC's working prototype on Bachmann PLCs and EMS servers show that CPAC can terminate several past real control system incidents and perform $N-x$ contingency analysis with run-time performance overhead of only 150 *ms*.

Acknowledgements

This work is supported in part by the US National Science Foundation under grant numbers CNS-1446471, CNS-1453046, CNS-1540216, and CNS-1540217.

10. REFERENCES

[1] "NIST Role-Based Access Controls" ; available at http://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf.
[2] "Modbus" ; available at http://www.modbus.org/.

[3] "ViaSat" Critical Infrastructure Protection; available at https://www.viasat.com/services/critical-infrastructure-protection, 2015.

[4] "OSIsoft" Real-Time Intelligence; available at https://www.osisoft.com/, 2015.

[5] A trusted safety verifier for process controller code, author=McLaughlin, Stephen and Zonouz, Saman and Pohly, Devin and McDaniel, Patrick, booktitle=NDSS, year=2014.

[6] *Power system analysis, author=Grainger, John J and Stevenson, William D, volume=31, year=1994, publisher=McGraw-Hill New York*.

[7] Towards Secure Metering Data Analysis via Distributed Differential Privacy, author=Liao, Xiaojing and Formby, David and Day, Carson and others, booktitle=IEEE DSN, year=2014.

[8] Dynamic taint propagation for Java, author=Haldar, Vivek and Chandra, Deepak and Franz, Michael. In *ACSAC*, 2005.

[9] Human error cited as cause of Florida blackout; available at http://appanet.files.cms-plus.com/PDFs/March10PPW.pdf, 2008.

[10] Arizona-Southern California Outages; available at http://www.ferc.gov/legal/staff-reports/04-27-2012-ferc-nerc-report.pdf, 2012.

[11] Wind River Recognized as Global Embedded Leader: Process Visualization Browser; available at http://www.windriver.com/news/press/pr.html?ID=10681#sthash.PPtTnAIX.dpuf, 2012.

[12] Dragonfly: Cyberespionage Attacks Against Energy Suppliers https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf, July 2014.

[13] PV-Browser: Process Visualization Browser; available at http://pvbrowser.de, 2015.

[14] Analysis of the Cyber Attack on the Ukrainian Power Grid http://www.nerc.com/pa/CI/ESISAC/Documents/E-ISAC_SANS_Ukraine_DUC_18Mar2016.pdf, 2016.

[15] A. Abur and A. G. Exposito. *Power system state estimation: theory and implementation*. CRC Press, 2004.

[16] D. T. Askounis and E. Kalfaoglou. The Greek EMS-SCADA: from the contractor to the user. *Power Systems, IEEE Transactions on*, 15(4):1423–1427, 2000.

[17] M. J. Assante. Confirmation of a Coordinated Attack on the Ukrainian Power Grid. SANS Industrial Control Systems Security Blog; available at https://ics.sans.org/blog/2016/01/09/confirmation-of-a-coordinated-attack-on-the-ukrainian-power-grid, Jan. 2015.

[18] H. Berger. *Automating with SIMATIC: Controllers, Software, Programming, Data*. John Wiley & Sons, 2012.

[19] S. Bi and Y. J. Zhang. Defending mechanisms against false-data injection attacks in the power system state estimation. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pages 1162–1167. IEEE, 2011.

[20] D. Brumley, I. Jager, T. Avgerinos, and E. J. Schwartz. BAP: a binary analysis platform. In *CAV*, 2011.

[21] P. Buneman, S. Khanna, and W.-C. Tan. Data provenance: Some basic issues. In *FST TCS 2000: Foundations of software technology and theoretical computer science*, pages 87–93. Springer, 2000.

[22] A. A. Cárdenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and others. Challenges for Securing Cyber Physical Systems. In *DHS Workshop on Future Directions in Cyber-physical Systems Security*, 2009.

[23] D. Chatterjee, J. Webb, Q. Gao, M. Vaiman, M. Vaiman, and M. Povolotskiy. N-1-1 AC contingency analysis as a part of NERC compliance studies at midwest ISO. In *Transmission and Distribution Conference and Exposition, 2010 IEEE PES*, pages 1–7. IEEE, 2010.

[24] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes. Using Model-based Intrusion Detection for SCADA Networks. In *Proc. SCADA Security Scientific Symposium*, 2007.

[25] A. Clark, Q. Zhu, R. Poovendran, and T. Basar. An impact-aware defense against Stuxnet. In *American Control Conference (ACC), 2013*, pages 4140–4147. IEEE, 2013.

[26] E. C. R. Council. The economic impacts of the August 2003 blackout. *Washington, DC*, 2004.

[27] C. M. Davis and T. J. Overbye. Multiple element contingency screening. *Power Systems, IEEE Transactions on*, 26(3):1294–1301, 2011.

[28] K. De Brabandere, B. Bolsens, J. Van den Keybus, A. Woyte, J. Driesen, and R. Belmans. A voltage and frequency droop control method for parallel inverters. *IEEE Trans. Power Elec.*, 22(4):1107–1115, 2007.

[29] W. Enck, P. Gilbert, S. Han, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Trans. Comp. Sys.*, 32(2):5, 2014.

[30] N. Falliere, L. O. Murchu, and E. Chien. W32.Stuxnet Dossier. Technical report, Symantic Security Response, Oct. 2010.

[31] H. Fawzi, P. Tabuada, and S. Diggavi. Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks. *IEEE Trans. Automat. Contr.*, 59(6):1454–1467, June 2014.

[32] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems. In *NDSS*, 2016.

[33] A. Fras and T. Dang. Improving industrial application's performances with an Historian. In *IEEE Intl. Conf. on Industrl Tech.*, 2004.

[34] J. D. Glover, M. Sarma, and T. Overbye. *Power System Analysis & Design, SI Version*. Cengage Learning, 2011.

[35] W. M. Goble. *Control Systems Safety Evaluation and Reliability*. International Society of Automation, 2010.

[36] S. Greene. *Margin and sensitivity methods for security analysis of electric power systems*. PhD thesis, University of Wisconsin–Madison, 1998.

[37] K.-H. John and M. Tiegelkamp. *IEC 61131-3: programming industrial automation systems*. Springer Science & Business Media, 2010.

[38] R. Lepofsky. North American Energy Council Security Standard for Critical Infrastructure Protection (NERC CIP). In *The Manager's Guide to Web Application Security:*, pages 165–176. Springer, 2014.

[39] L. Leshin, P. Mahaffy, C. Webster, M. Cabane, P. Coll, P. Conrad, P. Archer, S. Atreya, A. Brunner, A. Buch, et al. Volatile, isotope, and organic analysis of martian fines with the Mars Curiosity rover. *Science*, 341(6153):1238937, 2013.

[40] C. Masone et al. Role Definition Language (RDL): A language to describe context-aware roles. 2002.

[41] S. McLaughlin, P. McDaniel, and W. Aiello. Protecting consumer privacy from electric load monitoring. In *ACM CCS*, 2011.

[42] Y. Mo and R. M. Murray. Multi-dimensional state estimation in adversarial environment. In *34th Chinese Control Conference (CCC)*, pages 4761–4766. IEEE, 2015.

[43] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo. S3A: Secure System Simplex Architecture for Enhanced Security of Cyber-Physical Systems. http://arxiv.org, 2012.

[44] National Energy Regulatory Comission. NERC CIP 002 1 - Critical Cyber Asset Identification, 2006.

[45] Nell Nelson,Rob VandenBrink. The Impact of Dragonfly Malware on Industrial Control Systems https://www.sans.org/reading-room/whitepapers/ICS/impact-dragonfly-malware-industrial-control-systems-36672, 2016.

[46] North American Electric Reliability Corporation, 2011. Stuxnet attackers used 4 Windows zero-day exploits, available at http://www.nerc.com/pa/Stand/Project%20200902%20Realtime%20Reliability%20Monitoring%20and/Project_2009-02_rmacsdt_white_paper_021611.pdf.

[47] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas. Robustness of Attack-Resilient State Estimators. ICCPS '14, pages 163–174, Washington, DC, USA, 2014. IEEE Computer Society.

[48] L. Pietre-Cambacédes, M. Tritschler, and G. N. Ericsson. Cybersecurity myths on power control systems: 21 misconceptions and false beliefs. *IEEE Transactions on Power Delivery*, 26(1):161–172, 2011.

[49] H. L. F. C. E. Y. Ravi S. Sandhu, Edward J. Coyne. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[50] C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes. SPL: An Access Control Language for Security Policies and Complex Constraints. In *NDSS*, volume 1, 2001.

[51] J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell. A Quantitative Evaluation of the Target Selection of Havex ICS Malware Plugin. *Industrial Control System Security (ICSS) Workshop*, 2015.
[52] M. D. Schwartz, J. Mulder, J. Trent, and W. D. Atkins. Control system devices: Architectures and supply channels overview. *Sandia Report SAND2010-5183, Sandia National Laboratories, Albuquerque, New Mexico*, 2010.
[53] J. S. Shapiro, J. M. Smith, and D. J. Farber. EROS: a fast capability system. In *ACM SOSP*, 1999.
[54] S. Singh and S. Srivastava. Improved voltage and reactive power distribution factors for outage studies. *IEEE Trans. Power Systems*, 12(3):1085–1093, 1997.
[55] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. BitBlaze: A new approach to computer security via binary analysis. In *International Conference on Information Systems Security*, pages 1–25. Springer, 2008.
[56] S. Sridhar, A. Hahn, and M. Govindarasu. Cyber-Physical System Security for the Electric Power Grid. *Proc. IEEE*, 100(1):210–224, Jan. 2012.
[57] K. Stouffer, J. Falco, and K. Scarfone. Guide to Industrial Control Systems (ICS) Security. *NIST Special Publication*, 800:82, 2008.
[58] P. Sun, X. J. Tang, H. H. Wang, W. Z. Zhong, J. Wang, and H. M. Luo. Review of AGC and Primary Frequency Regulation. 986:1263–1267, 2014.
[59] N. Toshida, M. Uesugi, Y. Nakata, M. Nomoto, and T. Uchida. Open distributed EMS/SCADA system. *Hitachi Review*, 47(5):208–213, 1998.
[60] U.S. Department of Energy Office of Electricity Delivery and Energy Reliability. A Summary of Control System Security Standards Activities in the Energy Sector, October 2005.
[61] S. VanDeBogart, P. Efstathopoulos, E. Kohler, et al. Labels and Event Processes in the Asbestos Operating System. *ACM Trans. Comput. Sys.*, 25(4), 2007.
[62] J. Weiss. Are the NERC CIPS making the grid less reliable. *Control Global*, 2009.
[63] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman. Linux security module framework. In *Ottawa Linux Symposium*, volume 8032, pages 6–16, 2002.
[64] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti. AVATAR: A framework to support dynamic security analysis of embedded systems firmwares. In *NDSS*, 2014.
[65] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières. Making information flow explicit in HiStar. In *OSDI*, 2006.
[66] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *Power Systems, IEEE Transactions on*, 26(1):12–19, 2011.
[67] S. Zonouz, C. M. Davis, K. R. Davis, R. Berthier, R. B. Bobba, and W. H. Sanders. SOCCA: A security-oriented cyber-physical contingency analysis in power infrastructures. *Smart Grid, IEEE Transactions on*, 5(1):3–13, 2014.

APPENDIX

A. SECOND-ORDER TAYLOR EXPANSION

CPAC implements the physical information flow analysis through dynamic behavior inspection of the plant around Equation 1's (Section 4) equilibrium state using the plant's Taylor approximate equivalent Equation 5 which uses the first-order partial derivatives (Jacobian matrix) of the power plant's vector-valued function $f_x = \frac{\partial f}{\partial x}(x_0, u_0, \lambda_0)$, $f_u = \frac{\partial f}{\partial u}(x_0, u_0, \lambda_0)$, and $f_\lambda = \frac{\partial f}{\partial \lambda}(x_0, u_0, \lambda_0)$. Assuming that f_x is non-singular, we can reorder Equation 4 as

$$\begin{aligned}\Delta w = (w_u - w_x f_x^{-1} f_u)\Delta u + \\ + \Delta u^T (\frac{1}{2}(f_x^{-1} f_u)^T w_{xx} f_x^{-1} f_u - w_{ux} f_x^{-1} f_u + \frac{1}{2} w_{uu})\Delta u,\end{aligned} \tag{8}$$

The sensor measurements are correlated with the plant state and the operator's control inputs through (2) at nominal operating point

The first order taylor series expansion of (2) is given in (6). For line flow analysis, w is line flow vector and u is the vector of series capacitor reactances. fx is the jacobian matrix know from Newton-Raphson and the jacobin matrices w_u, w_x and f_u are shown in Figure 13.

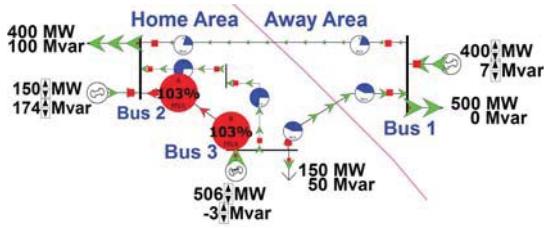
For higher policy enforcement accuracy via considering higher order dynamics of the plant, CPAC makes use of second order approximation as in (8). $w_x x, w_u x$ and $w_u u$ are second order jacobian matrices for line flow analysis are shown in Figure 13.

B. FOUR BUS POWER SYSTEM CASE STUDY

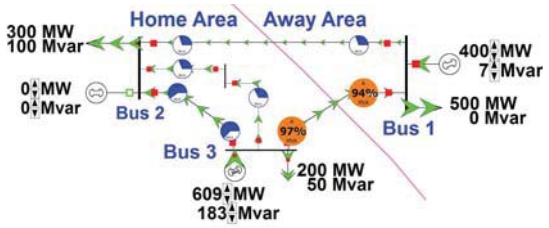
Case B: Ensuring safe power grid control. The power system is already in an unsafe state (shown in Figure 11a), where two transmission lines experience high currents. Bob, as the control operator, asks to increase the generation set-point on power bus 2. Based on the power system flow equations, this would increase line flows across the system.
Source of incident: No enforcement of control system integrity.
Required access control policy: In the case of unsafe states, control operators must not take actions that further worsens the situation (i.e., increases the overflows).
Effects of CPAC *deployment:* CPAC denies Bob's request since his action would violate the policy (Figure 12), because the action's execution drives the system further into less safe states while the system is already not safe.
Case C: Inter-area power transfer regulation. Alice requests to open the generator on Bus 2 from the rest of the grid so that she can perform follow-up maintenance tasks on the generator. In real-world practice, the inter-area power transfers should be maintained based on the scheduled values [58].
Source of incident: No regulation of actions that can affect remote power systems.
Required access control policy: Maintenance operators' action impact should be limited to the home area; their action must not affect the away area's operation.
Effects of CPAC *deployment:* CPAC denies Alice's action request once it completes its physics-based analysis. Figure 11b shows the state that the system would enter following Alice's action. The line on inter-area tie-line (the line that connects home and away areas together) is indirectly affected if Alice's action occurs, and hence her action is denied.

```
curr_low(0).
curr_high(100).
event_curr_inrange_alice(U,I,N,W) :-
U==Alice,I==w,N==currReq,
curr_low(L),curr_high(H),W>=L,W=<H.
```

Figure 12: Prolog Policy Rule for Case B.



(a) Bob's Action Request in Case B.



(b) Alice's Action Request in Case C.

Figure 11: The four-bus system presented in Figure 11a after operator modification requests.

Jacobian matrix $w_u = [w_{u_c}]$

$$\frac{\partial w_l}{\partial Z_{cl'}} = \begin{cases} \gamma_l(V_i^2 - V_i V_j \cos\theta_{ij}) - \beta_l V_i V_j sin\theta_{ij} & l = l' \\ 0 & l \neq l' \end{cases}$$

Jacobian matrix $F_u = \begin{bmatrix} P_{z_c} \\ Q_{z_c} \end{bmatrix}$

$$\frac{\partial P_i}{\partial Z_{Cl}} = \begin{cases} \gamma_l(V_i^2 - V_i V_j \cos\theta_{ij}) - \beta_l V_i V_j sin\theta_{ij} & i \neq SLK, l \in S^{L(i)} \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial Q_i}{\partial Z_{Cl}} = \begin{cases} \beta_l(-V_i^2 + V_i V_j \cos\theta_{ij}) - \gamma_l V_i V_j sin\theta_{ij} & i \in S^{PQ}, l \in S^{L(i)} \\ 0 & \text{otherwise} \end{cases}$$

Jacobian matrix $w_{ux} = [w_{Z_c\theta} \quad w_{Z_c V}]$

$$\frac{\partial w_{Z_c}}{\partial \theta_i} = \begin{cases} \gamma V_i V_j sin\theta_{ij} - \beta_l V_i V_j \cos\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial w_{Z_c}}{\partial \theta_j} = \begin{cases} -\gamma_l V_i V_j \sin\theta_{ij} + \beta_l V_i V_j \cos\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial w_{Z_c}}{\partial V_i} = \begin{cases} 2\gamma_l V_i - \gamma_l V_j \cos\theta_{ij} - \beta_l V_j \sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial w_{Z_c}}{\partial V_j} = \begin{cases} -\gamma_l V_i \cos\theta_{ij} - \beta_l V_i \sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

Jacobian matrix $w_{xx} = [w_{\theta\theta} \quad w_{vv}]$

$$\frac{\partial^2 w_l}{\partial \theta_i^2} = \begin{cases} g_l V_i V_j \cos\theta_{ij} + b_l V_i V_j \sin\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial^2 w_l}{\partial \theta_j^2} = \begin{cases} g_l V_i V_j \cos\theta_{ij} + b_l V_i V_j \sin\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial^2 w_l}{\partial V_i^2} = \begin{cases} 2g_l & \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial^2 w_l}{\partial V_j^2} = \begin{cases} 0 & i \in S^{PQ}, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

Jacobian matrix $w_{uu} = [w'_{z_c}]$

$$\frac{\partial w_l}{\partial Z_{Cl'}} = \begin{cases} \gamma'_l(V_i^2 - V_i V_j \cos\theta_{ij}) - \beta'_l V_i V_j sin\theta_{ij} & l = l' \\ 0 & l \neq l' \end{cases}$$

Jacobian matrix $w_x = [w_\theta \quad w_v]$

$$\frac{\partial w_l}{\partial \theta_i} = \begin{cases} g_l V_i V_j \sin\theta_{ij} - b_l V_i V_j \cos\theta_{ij} & i \neq SLK, i, j \in l \\ o & \text{otherwise} \end{cases}$$

$$\frac{\partial w_l}{\partial \theta_j} = \begin{cases} -g_l V_i V_j \sin\theta_{ij} + b_l V_i V_j \cos\theta_{ij} & i \neq SLK, i, j \in l \\ o & \text{otherwise} \end{cases}$$

$$\frac{\partial w_l}{\partial V_i} = \begin{cases} 2g_l V_i - g_l V_j \cos\theta_{ij} - b_l V_j \sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ o & \text{otherwise} \end{cases}$$

$$\frac{\partial w_l}{\partial V_j} = \begin{cases} -g_l V_i \cos\theta_{ij} - b_l V_i \sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ o & \text{otherwise} \end{cases}$$

Notations:

n_l	number of transmission lines
l, l'	indices for transmission lines
z_l	series impedance of line l
y_l	series admittance of line l
Z_c	series capacitive resistances
P	active power injections at all nodes except slack node
Q	reactive power injections at PQ-nodes
w	active power line flows
$S^{L(i)}$	set of lines connected to bus i
S^{PV}	set of PV-nodes
S^{PQ}	set of PQ-nodes
SLK	slack node

Definitions:

z_l	$r_l + j(x_l - x_{cl})$
y_l	$g_l + jb_l$
Z_c	${x_{cl}}^{nl}$
θ_{ij}	$\theta_i - \theta_j$
γ_l	$\frac{\partial g_l}{\partial x_{cl}} = \frac{2r_l(x_l - x_{cl})}{(r_l^2 + (x_l - x_{cl})^2)^2}$
β_l	$\frac{\partial b_l}{\partial x_{cl}} = \frac{-(x_l - x_{cl})^2 + r_l^2}{(r_l^2 + (x_l - x_{cl})^2)^2}$
γ'_l	$\frac{-2r_l(r_l^2 + (x_l - x_{cl})^2) + 6r_l(x_l - x_{cl})^2}{(r_l^2 + (x_l - x_{cl})^2)^3}$
β'_l	$\frac{2(x_l - x_{cl})(r_l^2 + (x_l - x_{cl})^2)^2 + 4(r_l^4 - (x_l - x_{cl})^4)(x_l - x_{cl})}{(r_l^2 + (x_l - x_{cl})^2)^4}$

Figure 13: Physical-Side Sensitivity-Based Information Flow Analysis