CPINDEX: Cyber-Physical Vulnerability Assessment for Power-Grid Infrastructures

Ceeman Vellaithurai, Anurag Srivastava, Saman Zonouz†, Robin Berthier‡
Electrical Engineering and Computer Science, †Electrical and Computer Engineering, ‡Information Trust Institute
Washington State University, †Rutgers University, ‡University of Illinois
ceeman.vellaithurai@wsu.edu, asrivast@eecs.wsu.edu, saman.zonouz@rutgers.edu, rgb@illinois.edu

***Abstract*—To protect complex power-grid control networks, power operators need efficient security assessment techniques that take into account both cyber side and the power side of the cyber-physical critical infrastructures. In this paper, we present CPINDEX, a security-oriented stochastic risk management technique that calculates cyber-physical security indices to measure the security level of the underlying cyber-physical setting. CPINDEX installs appropriate cyber-side instrumentation probes on individual host systems to dynamically capture and profile low-level system activities such as inter-process communications among operating system assets. CPINDEX uses the generated logs along with the topological information about the power network configuration to build stochastic Bayesian network models of the whole cyber-physical infrastructure and update them dynamically based on the current state of the underlying power system. Finally, CPINDEX implements belief propagation algorithms on the created stochastic models combined with a novel graph-theoretic power system indexing algorithm to calculate the cyber-physical index, i.e., to measure the security-level of the system's current cyber-physical state. The results of our experiments with actual attacks against a real-world power control network shows that CPINDEX, within few seconds, can efficiently compute the numerical indices during the attack that indicate the progressing malicious attack correctly.**

I. INTRODUCTION

The smart grid is a modern electric power network infrastructure to improve efficiency, reliability and security through automated control, data management and advanced communication technologies. Intelligent control and automation using advanced information technologies is the driving force to the vision of a smarter grid [1]. Advancements in measurement devices such as the phasor measurement units (PMU) and smart meters have allowed greater visibility with high-resolution data in the electric power grid (EPG). Distribution automation, renewable energy integration, real time control and intelligent algorithms powered by enhanced information network helps to realize the vision of the smart grid [2], [3].

The EPG is a complex network of interdependent networks consisting of power, sensors, data acquisition, communication network, control devices, computational and management framework. These network of networks are assisted by human in the loop with the objective to maintain demand-supply balance while meeting requirement of high reliability and efficiency. Hence, the smart grid can be referred as a cyber-physical system, wherein a change in one realm may have ramifications in the other realm [4]. The EPG has been recognized as a critical infrastructure which can be a target of cyber-attacks leading to loss of power/blackout resulting in financial and life loss [5]. Traditionally contingency analysis performed for the power system assumes a natural or physical malfunction of a device as the cause of faults in the systems. North American electric reliability corporation (NERC) regulations state that the EPG needs to be able to run in an N-1 contingency state [6]. This means that the EPG should be capable of withstanding the loss of any single device such as a large generator and still be within normal operating limits. However, this does not take into account the cyber aspects of the grid. Traditional cyber analysis may not work for the smart grid since it needs to take into account the physical consequences of intrusion into cyber-asset. This requires a rethinking of the commonly used security algorithms to reflect the tight coupling between cyber and physical systems. A combined cyber-physical analysis of the power system and associated cyber system is needed to establish criteria for vulnerability assessment [7].

To model impact of cyber system on physical system, information availability, attacker profile, timeline for attack and power system dynamics need to be considered. Authors in [8] used cause-effect relationships for cyber-physical analysis and estimating impact using graph theory approach. Petri net was used by authors to model cyber-physical attack in [9]. Authors in [10] discussed about four types of interdependencies and a state mapping to map the failures in the cyber network to the failures of the power network. Optimization models are introduced to minimize the impact on the reliability indices for a smart micro-grid application. A security model is developed to evaluate viable paths that an attacker could exploit in [11]. Proposed methodologies were evaluated for AMI infrastructure.

All the above approach generally assumes availability of complete information. Fundamental changes are required to develop contingency ranking methodologies for evaluating attack patterns from a cyber-physical perspective. Contingency analysis algorithms require all available information in the power grid to be able to compute the ranking of contingency severity. However, it is to be noted here that a cyber-attacker may not have access to all the information in the power grid unless the attacker had intruded the system and passively collected information about the system. Even in such a case, it is not possible to predict or obtain all the information required to run a full scale contingency analysis algorithm. It is likely that the information available to an attacker is just the publicly available information about the EPG such as topology information. Contingency analysis metrics can be derived through the use of graph theory based on just the topology information to represent the targets, an attacker is likely to choose to attack [5].

The main role of the security assessment component in a response system is to score security of the current system state at each time instant. The proposed approach makes use of information flows among system assets, such files and processes, to enable response systems to assess system security before deciding upon actions. The only input that the security assessment algorithm needs from the administrators is a list of major mission-critical assets in the organization. Indeed, this

input is subjective and thus impossible to generate automatically via observation of the system. During the learning phase, the assessment engine captures the normal communication patterns of individual assets in the system. It does so by tracing the information flow, at a time when there is no attack, and then automatically developing a dependency graph, i.e., a Bayesian network, that represents the system characteristics when there is no attack. Later, while the system is operating, given the learned dependency graph and the mission-critical assets, the assessment engine evaluates the security score for the current state of the system, based on the triggered intrusion detection system (IDS) alerts. In particular, to evaluate the security of the system, the assessment engine employs a taint-tracking approach by running a Bayesian belief propagation algorithm, i.e., Monte Carlo Gibbs sampling [12], on the learned dependency graph to calculate the probability that the critical assets are still secure.

The objective of this paper is to present the cyber-physical vulnerability assessment of smart grid using contingency ranking based on incomplete information and cyber intrusion ranking methodology. The contributions of this are as follows: *i)* we introduce an accurate and practically scalable cyber-physical security assessment metric that could be used for security-oriented risk management in critical power grid infrastructures; and *ii)* we completed a complete working prototype of the proposed solutions and evaluated the efficiency of the proposed solution on a real-world test-bed system that we built from scratch.

II. Cyber Vulnerability Assessment

In this section, we explain how CPIndex makes use of the cyber-side security sensors to determine the current cyber security state and a stochastically learned dependency graphs to calculate the security level of the system's current state at each time instant [13]. We start with discussing how the dependency graphs are generated (Section Section II-A) and continue with description of the security index calculation algorithm (Section Section II-B).

A. Dependency Graph

The goal of the dependency graph (DG) is to free the administrator from providing low-level details about the organization. Those details are automatically captured by Seclius through a learning phase, during which the interactions between files and processes are tracked in order to probabilistically identify direct or indirect dependencies among all the system assets. For instance, in a database server, the administrator only needs to list the sensitive database files, and Seclius later marks the process `mysqld` as critical because it is in charge of reading and modifying the databases. Such a design greatly reduces the resources and time spent by administrators in deploying Seclius.

Each vertex in the DG represents an object, namely a file, a process, or a socket, and the direct dependency between two objects is established by any type of information flow between them. For instance, if data flow from object o_i to o_j, then object o_j becomes dependent on o_i; the dependency is represented by a directed edge in the DG, i.e., $o_i \rightarrow o_j$. To capture that information, Seclius intercepts syscalls and logs them during the learning phase. In particular, we are interested

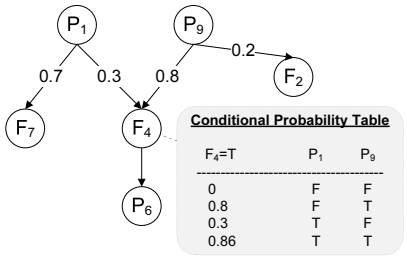


Fig. 1. Conditional Probability Table Construction

in the syscalls[1] that cause data dependencies among the OS-level objects. A dependency relationship is stored by three elements: a source object, a sink object, and their security contexts. When the learning phase is over, syscall logs are automatically parsed and analyzed line by line to generate the DG. Each dependence edge is tagged with a frequency label indicating how many times the corresponding syscalls were called during the execution.

We make use of the Bayesian network formalism to store probabilistic dependencies in the DG; a conditional probability table (CPT) is generated and associated with each vertex. This CPT encodes how the information flows through that vertex from its parents (sources of incoming edges) to its children. For example, if some of the parent vertices of a vertex become tainted directly or indirectly by attacker data, the CPT in the vertex saves the probability that the vertex (specifically, the OS-level object represented by the vertex) also gets tainted.

More specifically, each DG vertex is modeled as a binary random variable (representing a single information flow), equal to either 1 (true) or 0 (false) depending on whether the vertex has been tainted; the CPT in a vertex v stores the probability that the corresponding random variable will take the true value ($v = 1$), given the binary vector of values of the parent vertices $P(v)$. Formally, the probability value is computed using the following equation: $Pr(v|P(v)) = 1 - \prod_{p_v^i \in P(v)} \{1 - \mathbf{1}_{(p_v^i)}.Pr(p_v^i \rightarrow v)\}$ where $\mathbf{1}_{(.)}$ is the indicator function that takes on the value 1 if the condition inside the parentheses holds, and 0 otherwise. $Pr(p_v^i \rightarrow v)$ is the probability that the information will flow from the parent node p_v^i to the vertex v. This probability represents the fraction of times during which information flows from p_v^i to v. It is calculated using the frequency labels on p_v^i's outgoing edges that are captured during the learning phase. In summary, the vertex v takes on the value 1 if information flows from any of its parents that have the value 1.

Figure 1 illustrates how a CPT for a single flow (with 1-bit random variables) is produced for a sample vertex, i.e., the file F_4. The probabilities on the edges represent $Pr(. \rightarrow .)$ values. For instance, the process P_1 writes data to the files F_4 and F_7 with probabilities 0.3 and 0.7, respectively. As shown in the figure, the file F_4 cannot become tainted if none of its parents are tainted, i.e., $Pr(F_4|\bar{P_1}, \bar{P_9}) = 0$. If only the process P_1 is tainted, F_4 can become tainted only when the information flows from P_1, i.e., $Pr(F_4|P_1, \bar{P_9}) = 0.3$. If both of the parents are already tainted, then F_4 would get tainted when information flows from either of its parent vertices. In that case, the probability of F_4 being tainted would be the complement probability

[1] Specifically, we log the following syscalls: `openat`, `dup`, `dup2`, `close`, `write`, `writev`, `read`, `readv`, `ipc`, `clone`, `fork`, `vfork`, `execve`, `open`, `creat`, `mmap`, `mmap2`, `socketcall`, `recv`, `recvfrom`, `recvmsg`, `send`, `sendto`, and `sendmsg`.

of the case when information flows from none of its parents. Therefore, $Pr(F_4|P_1,P_9) = 1-(1-0.3)\times(1-0.8) = 0.86$.

Each security incident could be represented a CIA[2] criterion of a critical asset, and is modeled by CPINDEX as an information flow between the privilege domains controlled by the attacker (according to the current system state) and those that are not yet compromised. Confidentiality of an object is compromised if information flows from the object to any of the compromised domains. Integrity of an object is similarly defined, but the flow is in the reverse direction. Availability is not considered as an information flow by itself; however, an object's unavailability causes a flow originating at the object, because once an object becomes unavailable, it no longer receives or sends out data as it would if it was not compromised. For instance, if a process frequently writes to a file, once the process crashes, the file is not modified by the process, possibly causing inconsistent data integrity; this is modeled as a propagation of tainted data from the process to the file. We consider all the leaf nodes that concern the integrity criterion of critical assets as a single information flow, because they conceptually address the same flow from any of the compromised domains to the assets. However, confidentiality flows cannot be grouped, as they originate individually at separate sources.

If each information flow is represented as a bit, then to completely address n concurrent information flows, we define the random variable in each vertex as an n-bit binary vector in which each bit value indicates whether the vertex is already tainted by the bit's corresponding flow. The CPTs are generated accordingly; a vertex CPT stores the probability of the vertex's value given the value of its parents, each of which, instead of true or false, can take on any n-bit value.

B. System Security Evaluation

Given the DG generated during the learning phase, the operator turns off the syscall interception instruments and puts the system in production mode. The learned DG is then used in an online manner to evaluate the security of any system security state. The goal of this section is to explain how this online evaluation works in detail. We first assume that the IDSes report the exact system state with no uncertainty. We discuss later how Seclius deals with IDS inaccuracies.

At each time instant, to evaluate the security of the system's current state s, DG vertices are first updated according to s, which indicates the attacker's privileges and past consequences (CT's leaf nodes). For each consequence in s, the corresponding flow's origin bit in DG is tainted. For instance, if file F_4 is modified by the attacker (integrity compromise), the corresponding source bit in DG is set to 1 (evidence bit).

The security measure for a given state s is defined to be the probability that the CT's root value is still 0 ($Pr(\overline{\texttt{root}}(\texttt{CT})|s)$), which means that the organizational security has not yet been compromised. More specifically, if the CT is considered as a Boolean expression, e.g., $\texttt{CT} = (C(F10) \wedge A(P_6)) \vee I(F_2)$, Seclius calculates the corresponding marginal joint distribution, e.g., $Pr[(C(F10) \cap A(P_6)) \cup I(F_2)]$, conditioned on the current system state (tainted evidence vertices).

Seclius estimates the security of the state s by calling a belief propagation procedure, namely, the Gibbs sampler [14], on the DG to probabilistically estimate how the tainted data (evidence bits) are propagated through the system while it is in state s.

[2]CIA stands for the security criteria: confidentiality, integrity, availability.

Generally, the Gibbs sampler algorithm [14] is a Monte Carlo simulation technique that generates a sequence of samples from a joint probability distribution of two or more random variables $X_1, X_2, \ldots, X_n$. The purpose of such a sequence in Seclius is to approximate the joint distribution numerically using large number of samples. In particular, to calculate a joint distribution $Pr(X_1, X_2, \ldots, X_n|e_1, \ldots, e_m)$, where e_i represents an evidence, the Gibbs sampler runs a Markov chain on $X = (X_1, X_2, \ldots, X_n)$ by 1) initializing X to one of its possible values $x = (x_1, x_2, \ldots, x_n)$; 2) picking a uniformly random index i ($1 \leq i \leq n$); 3) sampling x_i from $Pr(X_i|x, e)$ (represented by the conditional probability tables in the generated Bayesian network), 4) updating the x vector, and 5) going back to step 2. It has been proved that the stationary distribution of the Markov chain is just the sought-after joint distribution [14]. Thus, drawing samples from the Markov chain at long enough intervals, i.e., allowing enough time for the chain to reach the stationary distribution, gives independent samples from the distribution $P(X_1, \ldots, X_n|e)$.

We make use of the Gibbs sampler algorithm in Seclius for two main reasons. First, the DG model's joint distribution is not explicitly known initially, and second, analytical calculation of it can be tedious, if possible, specially for large DG graphs [14]. The Gibbs sampler uses the DG's CPT to generate a large number of samples from the $Pr[\texttt{CT}|s]$ distribution without directly calculating the density function [14]. Similarly, the security measure is estimated individually for each system state. Therefore, if the attacker modifies any other object and/or gets more privileges, the system would switch to a new state, whose security measure would be separately evaluated.

It is worth emphasizing that Seclius does not use the DG model to estimate how the attacker contacts other objects from a compromised object, such as a tainted process, to exploit a vulnerability and/or escalate his or her privileges. Seclius uses the DG only to estimate how the tainted data would propagate through other *non-compromised* system assets, which would behave normally as they did during the learning phase. For every asset already compromised, Seclius assumes a pessimistic behavior model, i.e., the asset deterministically contacts all other assets in its privilege domain.

That approach can evaluate the security of each system state. However, the exact current security state of the system is usually not completely observable, due to IDS inaccuracies, i.e., false positive and negative rates. We define the notion of the *information state* of the system, which formally is a probability distribution over all states in the state space of the system $s \in S$. The information state of the system is estimated, based on the IDS alerts and the false positive and negative rates of the IDSes, using the following equation: $Pr(s) = \prod_{i=1}^{|s|-1}(\mathbf{1}_{s_i=1}.[1-\texttt{FP}(s_i)] + \mathbf{1}_{s_i=0}.[1-\texttt{FN}(s_i)])$, where $\mathbf{1}$ is the indicator function, and s_i is the binary state variable in state s. $\texttt{FP}(s_i)$ and $\texttt{FN}(s_i)$ denote the false positive and negative rates, respectively, that depend on the intrusion detection system by which the corresponding alerts are triggered. The false positive and negative rates can be set to be deterministic, i.e., 0, if the detectors are quite accurate. Seclius can also take qualitative values, i.e., $\texttt{FP} : s_i \rightarrow \{\texttt{low}, \texttt{medium}, \texttt{high}\}$, which are later translated into crisp values, i.e., $\{0.25, 0.5, 0.75\}$.

Once the information state of the system has been estimated, Seclius computes the expected security measure of the information state using the following equation: $\sum_{s \in S}(Pr(s).\texttt{Sec}(s))$, where the $\texttt{Sec}$ function is the security measure evaluated for the exact state s, as described above.

III. Power Contingency Analysis

It is highly unlikely that an attacker has access to all the information required to launch a coordinated attack on the power grid. Hence, the physical contingency ranking is derived based on incomplete information available to the attacker. It is assumed that the attacker has access to the topology information of the power system and hence a graph theory based topology analysis is used to rank contingencies [15]. The power system is modeled as a graph G, where the buses in the power system are treated as a set of vertices V and branch components as a set of edges E. The edges are assigned weights to represent the dissimilarity between the branch components. The weights are based on the magnitude of reactance X, since X is usually greater than resistance R of the branch.

For the purpose of ranking generator contingencies, the concept of vertex centrality is used. Vertex centrality measures assign ranking coefficients to vertices in a graph, from which the most important generators can be identified as the ones located on buses with a high value of centrality index. Among all vertex centrality indices, evidence of a close relationship between closeness centrality index and impact of generator outages has been shown in [1]. The closeness centrality for an n bus system is defined as:

$$C_c(v_i) = \frac{\sum_{j \in vi} d(i,j)}{n-1} \tag{1}$$

The above equation relies on the use of shortest path distance $d(i,j)$ between the vertices which is computed using shortest path algorithm, such as the Dijkstra algorithm. The closeness centrality index extended for an $N-X$ case is given by the closeness impact centrality index as [16]:

$$CI_c(k) = \sum_{i \in V_{cont}} \|C_c(v_i)\|, \tag{2}$$

where V_{cont} is the set of generators considered for the $N-X$ case.

For the purpose of ranking line outage contingencies, the concept of edge betweeness centrality is used. Edge Betweeness techniques applied to the n bus power system case results in a measure of branch centrality. Evidence of a close relationship between edge betweeness centrality and impact of line outages has been shown in [15]. It is defined for each branch i in the system as:

$$C_{Be}(i) = \frac{2}{n(n-1)} \sum_{j \neq k \in V} \frac{\sigma_{jk}(i)}{\sigma_{jk}} \tag{3}$$

For a $N-X$ line contingency case, the removal of X edges in a graph G will be result in a sub graph $H = GX$. The resulting edge betweeness centrality of the edges in graph H is fundamentally different from that exhibited for graph G and so direct addition cannot be performed. All combination of edges in a $N-X$ contingency case k are determined by taking $X-1$ at a time, expressed as C_X^{X-1}. The list E is defined as having unique row entries having all but one of the edges $E(k)$. Within a row c of E, the edge $e_{o,c}$ from contingency k not appearing in $E(c)$ is defined as:

$$e_{o,c} = (ek) \notin E'(c) \tag{4}$$

A new sub graph $H_c = G - E'(c)$ is defined. The edge betweeness of $e_{o,c}$ can now be derived for the new graph. The edge betweeness centrality index extended for an $N-X$ case is given by the edge betweeness impact centrality index as [17]:

The physical contingency ranking centrality index for contingencies involving both generator and line outages is given by combining the closeness impact and edge betweeness impact centrality indices as:

$$CI_{PC} = CI_c(k) + CI_{Be}(k) \tag{5}$$

IV. Cyber-Physical Security Index

We describe the cyber-physical security index that CPIndex uses to ranking various possible attacks against a cyber-physical infrastructure. In particular, the cyber indexing algorithm that was discussed in Section II calculates conditional probabilities, i.e., the probability of critical assets being affected (directly or indirectly) as the result of compromised privilege domains in the network. To further clarify, a critical asset, .e.g., a relay configuration file, is affected *directly* by an attacker if he/she has successfully accessed that asset, e.g., modified the relay configuration file maliciously. Alternatively, a critical asset, e.g., relay configuration file, is *indirectly* affected by an attacker if it gets modified automatically by the operating system (e.g., constantly running daemons) as the result of a malicious direct modification of a non-critical asset, e.g., a database that the system reads and accordingly updates the relay configuration files. Additionally, the power system security indexing algorithm (Section III) calculates the severity level of any potential incident in the power system that could be caused by the attackers.

CPIndex's objective is to come up with a measure to rank the cyber-originated intrusions that target the physical power system components through a multi-step attack scenario of compromising several host computers in the control center and eventually gaining control over the critical asset that controls the physical component and injecting malicious control commands causing a power-side security incident. CPIndex makes use of the computed conditional probabilities for the cyber incidents and the physical contingency centrality indices for the power-side incidents to calculate a unified cyber-physical security index for the whole power grid infrastructure. In particular, CPIndex is given with the cyber network's current security status, i.e., the set of compromised host systems, using the alerts from the deployed intrusion detection system sensors throughout the control center. Based on the compromised hosts knowledge, CPIndex implements the cyber-side indexing algorithm to compute the probabilities of the critical computing assets, which control the power system components, getting affected as the compromised host nodes. Consequently, each power system component, e.g., a generator, will be assigned by a probability value that indicates how likely is it that that component is affected maliciously directly or indirectly by the attacker that could potentially result in a power contingency, e.g., generator outage.

CPIndex then employs the power indexing algorithm to calculate the severity of such potential power contingency if it actually occurs in the underlying power system. The computed probability values and consequent severity degrees for each power component enables CPIndex to calculate a security risk analysis-based ranking of all of the potential cyber-physical incidents. More specifically, once an attack occurs against the cyber network of the power grid, CPIndex analyzes the cyber connections and computes the cyber index followed by calculation of the power performance index for potential power grid contingencies given the current cyber attack and affected network assets. All the enumerated contingencies

are ranked based on the calculated risk measures; consequently, the power operators need to address the contingencies proactively given the computed ranking.

Needless to mention, because the above-mentioned cyber-physical indexing algorithm is heavily dependent on the cyber network's current security status, it needs to be rerun every time the cyber security status changes as the result of the a new host getting compromised based on the deployed intrusion detection systems. Based on our experiments, the rerunning of the algorithm does not take long in practice because the algorithm can use the pre-calculated measures from the system's previous cyber security state and updates them according to the system's current security state.

V. Evaluation Results

The goal of this section is to illustrate through a realistic case study, how CPIndex can assist administrators in calculating the cyber-physical index. We start by describing the case study, which is a multi-step attack occurring in a process control network. The cyber indices are calculated accordingly after each phase of the attack, and consequently the power-side performance indices are calculated. Finally, CPIndex computes the cyber-physical indices followed by a ranked list of the corresponding contingencies.

A. Cyber-Physical Test-Bed and the Attack Scenario

To illustrate how the CPIndex framework works in a critical infrastructure, we implemented it in a power supervisory control and data acquisition (SCADA) environment, namely the Trustworthy Cyber Infrastructure for the Power Grid (TCIPG) testbed [18]. We picked the SCADA environments because cyber security in such critical infrastructures is a paramount concern [19]. Furthermore, such process control networks display rather deterministic behaviors [20] that can be learned efficiently. The TCIPG testbed is compatible with the control systems reference architecture that has been proposed by the Department of Homeland Security. Figure 2(a) shows the high-level architecture of the testbed, which consists of three network zones, namely a demilitarized zone (DMZ), a corporate network, and a control network. The traffic is segmented by firewall rules that are depicted with arrows in Figure 2(a). The control network is in charge of receiving sensory information from the power system and sending control commands, i.e., power generation set points, back. In particular, we connected the control system to a PowerLinc X10-TW523[3] [21] device, which is a remote switch used to turn a remote power system relay off or on. The corporate network is responsible for business management and customer interaction. As shown in the figure, all the data flow between the corporate and control networks is allowed only through VPN tunnels. The corporate-customer interaction is done through the two Web servers, which are running in a demilitarized zone for improved security.

The testbed has intentionally been set up to include several real vulnerabilities. In particular, the CoreHTTP Web server that resides in the DMZ is vulnerable to a buffer overflow exploitation based on CVE-2007-4060. The desktop machine in the corporate network suffers from a weak root password, i.e., `root2008`, which makes the host vulnerable to password brute-force attacks. The control system is running the eVision content management system to store data, including configuration parameters, that pertain to the underlying power system relay. According to CVE-2008-6551, multiple directory traversal vulnerabilities in the eVision 2.0 content management system allow attackers to include and execute arbitrary PHP files.

As mentioned earlier, a system state is defined as the set of past consequences of attackers' actions and privileges over the system. Zabbix and Samhain are responsible for capturing the attack consequences. However, they do not provide any information about how attackers penetrate into the system, i.e., what vulnerabilities are exploited to gain privileges over the system. To detect the vulnerability exploitations and privilege escalations, we deployed Snort, LibSafe, ClamAV, and PHP-IDS.

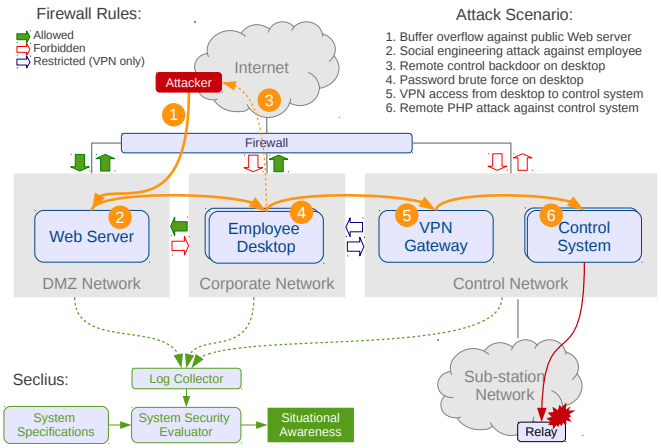
Figure 2(a) reveals the path through which we launched a multistep attack to get access to the control system. The attack scenario included the following steps. The remote attacker got access to the DMZ by exploiting the vulnerable CoreHTTP Web server to install a fake plugin request website. The goal of the malicious website was to lure one of the employees into downloading malware that we developed. This social engineering attack allowed the attacker to take control of an employee desktop inside the corporate network. We note that only such an indirect attack could work, because the firewall rules deny direct access/penetration from the DMZ zone or the Internet to the corporate network. The malware installed a back-door shell that sent periodic requests to the attacker. Once the shell access was gained on the corporate network machine, the attacker ran the *John the Ripper* password cracker to obtain the local root password. This step took about 18 seconds to complete. Then, using the root access, the attacker initiated a VPN session to the VPN gateway in order to launch a PHP code injection against a Web service hosted in the control system. This last penetration step made it possible for the remote attacker to open a shell in the control system and modify a sensitive configuration file used to control the underlying power system relay. In the testbed, the configuration file modification was physically demonstrated by the turning off of a relay-controlled light bulb.

B. Cyber-Index Computation

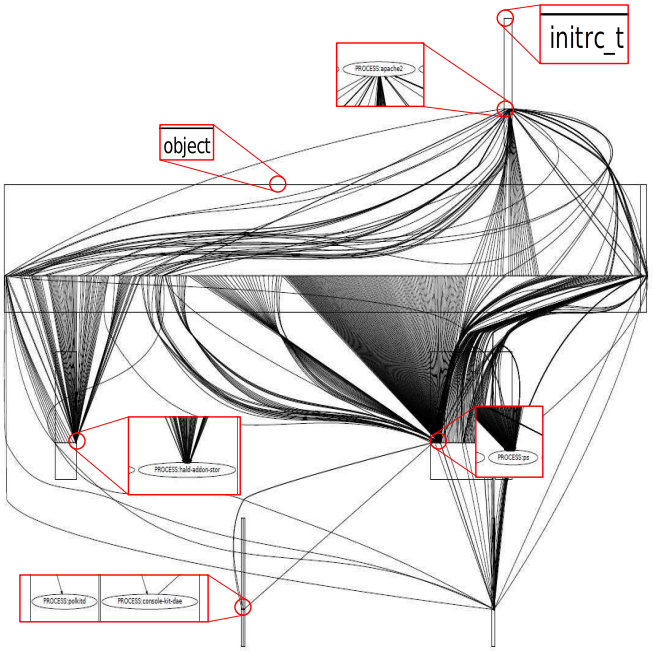
We now illustrate the benefits of CPIndex by comparing four different detection solutions applied to the same attack replayed three times against the testbed. First, we deployed the IDSes to monitor the critical assets *only*, and we did not use dependencies. Second, the intrusion detectors were configured to detect consequences for any (critical or noncritical) asset within the system, still without dependencies. Third, the detection scenario was identical to the previous one but CPIndex was also deployed to consider dependencies. The fourth and last detection scenario was identical to the previous one except that detection inaccuracies were injected through probabilistic discarding of some alerts. Using expert judgment, false positive and negative rates were set to 8 and 5 percent for Samhain, and 6 and 3 percent for Zabbix, respectively.

As a result, if a critical asset was touched by the attacker, it would be detected in all four scenarios. If a noncritical asset was touched, cases 2, 3, and 4 would detect it, but the detectors in case 2 would have had no information on how much any of the critical assets might be affected, and therefore would not have been able to distinguish among the noncritical assets being accessed. On the other hand, making use of the DG, CPIndex calculated the probability that the critical assets

[3]The PowerLinc from SmartHome provides 12V 300mA DC and two-way power-line communication to any PowerLinc-compatible unit. The device can further send/read RS232-type serial communication from/to a computer system or any other electronic device.



(a) Experimental Case Study: SCADA Testbed



(b) Dependency Graph 2(a)

Fig. 2. Experimental Case Study Network and the Corresponding Consequence Tree

TABLE I. CASE STUDY: COMPARING FOUR DIFFERENT DETECTION SCENARIOS (1, 2, 3, 4) WHERE 1: CRITICAL ASSETS ONLY, WITHOUT DEPENDENCIES; 2: ALL ASSETS, WITHOUT DEPENDENCIES; 3: CPINDEX, WITHOUT INACCURACIES; 4: CPINDEX, WITH INACCURACIES.

Detection Scenarios Comparison		Attack Steps				
		Initial	Web Server	Desktop	Gateway	Relay
Affected Nodes	CoreHTTP	(0.00, 0.00, 0.00, 0.03)	(1.00, 1.00, **1.00**, **0.94**)	(1.00, 1.00, 1.00, 0.94)	(1.00, 1.00, 1.00, 0.94)	(1.00, 1.00, 1.00, 0.94)
	Apache-DMZ	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)
	Clients-DB	(0.00, 0.00, 0.00, 0.00)	(0.00, 0.00, **0.04**, **0.03**)	(0.00, 0.00, **1.00**, **0.85**)	(0.00, 0.00, 1.00, 0.85)	(0.00, 0.00, 1.00, 0.85)
	Apache-Ctrl	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)
	Config-Files	(0.00, 0.00, 0.00, 0.00)	(0.00, 0.00, 0.00, 0.00)	(0.00, 0.00, 0.00, 0.00)	(0.00, 0.00, **0.40**, **0.37**)	(0.00, 0.00, 0.40, 0.37)
	MySQL	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)	(0.00, 0.00, 0.00, 0.03)
	DMZ	(0.00, 0.00, 0.00, 0.00)	(0.00, 0.00, 0.00, **0.02**)	(0.00, 0.00, 0.00, 0.02)	(0.00, 0.00, 0.00, 0.02)	(0.00, 0.00, 0.00, 0.02)
	Corporate	(0.00, 0.00, 0.00, 0.00)	(0.00, 0.00, **0.04**, **0.03**)	(0.00, 0.00, **1.00**, **0.85**)	(0.00, 0.00, 1.00, 0.85)	(0.00, 0.00, 1.00, 0.85)
	Control	(0.00, 0.00, 0.00, 0.07)	(0.00, 0.00, 0.00, 0.07)	(0.00, 0.00, 0.00, **0.06**)	(0.00, 0.00, **0.40**, **0.38**)	(0.00, 0.00, 0.40, 0.38)
	Organization	(0.00, 0.00, 0.00, 0.07)	(0.00, 0.00, **0.04**, **0.11**)	(0.00, 0.00, **1.00**, **0.87**)	(0.00, 0.00, 1.00, **0.92**)	(0.00, 0.00, 1.00, 0.92)
Ranked Alerts			*Bof*	*Pwd,Mlw,Bof*	*Pwd,Mlw,Bof,VPN,Net*	*Pwd,Mlw,Bof,VPN,Net*

would be affected even if none of them were directly accessed by the attacker.

Table I shows the results for all four cases right after each step of the attack. Each value in the table represents the probability of a particular security incident. For clarity, the zero values are grayed out and the meta-alerts are highlighted with bold fonts. Initially, when the attacker was not inside the system, all the DG vertices were set to false, except the socket on the Web servers in the DMZ network (i.e., the only entities that interacted with the Internet and on which potential attackers resided). As demonstrated in the first column of the table, all the security incident probabilities had initial values of zero. Once the Web server was compromised, CPINDEX updated the relevant nodes in the Bayesian network, namely all nodes in the Web server and the socket on the employee desktop, to true. The reason is that once a privilege domain was compromised, CPINDEX pessimistically marked all the nodes in that privilege domain as tainted, and triggered the corresponding meta-alerts. In our experiments, the Web server buffer overflow exploitation and the CoreHTTP process crash were detected by Snort and Zabbix, respectively.

As shown in the second column of Table I, the first attack step was detected in all the cases, as the availability of the CoreHTTP was directly affected by the attacker. It is important to note that the probability of the MySQL files being affected on the corporate network (in cases 3 and 4) also increased, although the files were not directly touched by the attacker. The increase makes sense because the attacker controlled the Web server, and thus could send data to the MySQL files (i.e., those listed as critical, integrity-wise). As shown, from the point of view of cases 1 and 2, only the Web server compromise was detected, and the database files were not affected. While assessing system security, CPINDEX took such potential adversarial flow under consideration, since it made use of the dependencies among the system components.

Then, the attacker installed the malware using a social engineering attack and got control over the corporate network machine. ClamAV was able to detect the malware as an illegitimate process and reported it. Within the newly controlled machine, the attacker replaced the `/usr/bin/mysql` executable with a downloaded executable `mysql` that, after being spawned,

accessed the clients' database[4], which was listed as a critical asset in the CT. Later, the attacker's attempt to crack the root password was detected by Zabbix due to the attack's high CPU consumption. As shown in the third column of Table I, the probability values in cases 1 and 2 were not affected as a result of the attack; however, CPINDEX pessimistically marked all the nodes in the compromised privilege domains as tainted, as processes within such domains were no longer trusted to follow their normal behavior.

Getting closer to the target, the attacker then got access to the VPN gateway. This step was detected by Snort, which had been configured to detect any connection initialized from the control network to the corporate network. Normally, the other direction was used in SCADA architectures to transfer the field sensory data into the corporate network. As there was no critical asset listed in the gateway, no node in the CT was affected in cases 1 and 2. However, as shown in the fourth column of Table I, node probabilities were updated by CPINDEX as the attacker got closer to the assets in the control system (the Bayesian network vertices corresponding to the gateway were marked as true), increasing the probability of their security criteria being indirectly affected (according to the DG). Consequently, the attacker started scanning the control network to find his or her next target, and this activity was detected by Snort.

Using a legitimate service, the attacker remotely sent relay set points to the control system without compromising it. This action turned off the physical relay, representing a potential threat for a large-scale blackout in the power grid. More specifically, the set points were first stored in a configuration database file, which was then used by a cron job to send relevant commands to the power component. The modification of the configuration database was not detected, as it was accomplished through legitimate channels. We note that to limit the false positive rate, the legitimate channels had already been white-listed. Therefore, neither case 1 nor case 2 changed any value in the CT; however, from the previous step and using the captured DG, CPINDEX had updated the CT values. Consequently, once the attack was completed, from the point of view of cases 1 and 2, the value of the root node, i.e., organizational security, was still 0. On the other hand, CPINDEX estimated the root node value as 1.0 after the attack, which better represents reality.

The last row of the table shows the list of alerts ranked from high to low after every attack step. The indices (*Bof*, *Mlw*, *Pwd*, *VPN*, *Net*) represent specific alerts that are mapped to alerts as follows. *Bof*: buffer overflow detected on Web server by Snort; *Mlw*: malware installation detected on desktop by ClamAV; *Pwd*: password bruteforce detected on desktop by Zabbix; *VPN*: VPN connection identified on gateway by Snort; *Net*: network scan detected on gateway by Snort. Alerts *Mlw* and *Pwd* flooded the screen, but here, for clarity, we map each one to a single index. After the attack's last step, CPINDEX ranked alert *Pwd* the first, since it contributed the most to the organizational security compromise, i.e., CT's root node value. On the other hand, alert *Net* was ranked the least important, since its corresponding event was just a network reconnaissance that did not affect the CT's root value and would usually be less crucial than explicit privilege escalations or malicious consequences.

[4]The access to the clients' database files was not detected even though the DazukoFS module had already been loaded and was monitoring accesses to the clients' database files; the reason was that accesses by the `mysql` process were configured to be white-listed, because the clients' databases are frequently accessed by legitimate users.

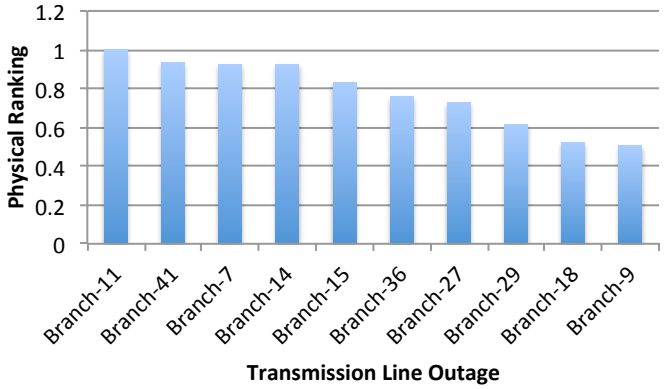


Fig. 3. N-1 Power Contingency Ranking in IEEE 30-Bus Test Case

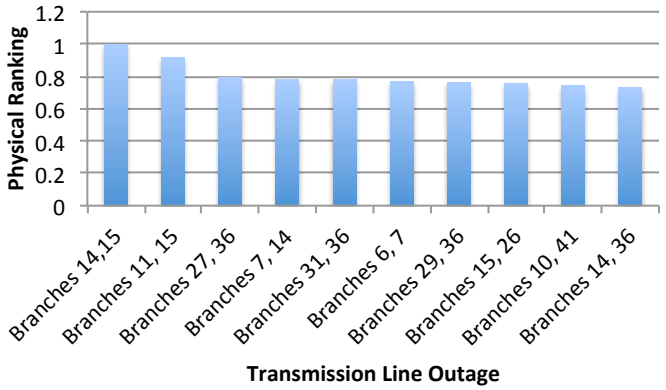


Fig. 4. Cyber-Physical Contingency Ranking in IEEE 30-Bus Test Case

C. Power System Contingency Ranking

We implemented the proposed graph-theoretic power system index calculation algorithm on IEEE 30-bus test-bed. Figure 3 shows the results for deployment of the power indexing algorithm for N-1 contingency ranking where a transmission line outage was simulated and ranked. The horizontal axis shows the individual incidents and the vertical axis represents the calculated and normalized numerical power indices. Figure 4 shows the top ten N-2 contingencies as ranked by the algorithm for line outages. It is noteworthy that we used a real-time digital simulator (RTDS), and the index values were calculated after the power system was simulated within MATPOWRR framework.

Table II shows the calculated numbers for more complicated scenarios where one generator and two transmissions lines are outaged and unusable. The last column shows the

TABLE II. N-3 POWER SYSTEM CONTINGENCY SCORES

Gen. Bus	1st Branch	2nd Branch	Score
11	14	15	1
13	14	15	0.9898
5	14	15	0.9623
11	11	15	0.9456
2	14	15	0.9441
13	11	15	0.9354
5	11	15	0.9079
8	14	15	0.8920
2	11	15	0.8896

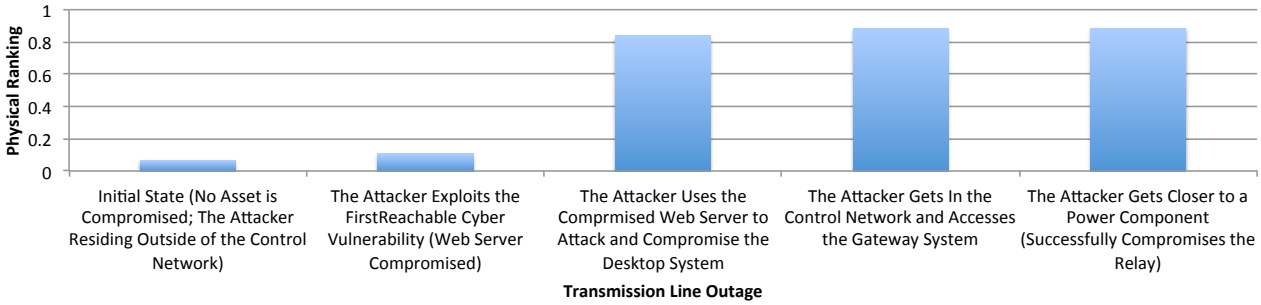


Fig. 5. Cyber-Physical Contingency Ranking

computed N-3 contingency rankings. As shown in the table, the last case (row) where the generator 2 and lines 11 and 15 are down caused the most critical impact on the power system.

D. Cyber-Physical Contingency Ranking

Given the calculated power system-side contingency ranking measures and the cyber network-side computed security-oriented risk management values on the test-bed control network, we now present how they both contribute to the overall hybrid cyber-physical indices that capture the security level of the system current state taking into account both the security status of the cyber assets as well as the power system components. Figure 5 shows the final results of the calculated cyber-physical index values during the multi-step attack that was explained earlier in this section (based on N-1 power indices). The descriptions of each step in the horizontal axis of the figure indicates the phase in the attack that it refers to. As shown, while the attacker further penetrates into the cyebr network, the calculated cyber-physical index (represented by the vertical axis) increases denoting that the attack is getting closer to cause a real-world (potential catastrophic) physical impact on the power network.

VI. Conclusions

In this paper, we presented CPIndex, an automated and proactive cyber-physical contingency analysis tool, that takes the cyber network configurations, the power system topology, as well as the current intrusion detection system alerts as its inputs and calculates a cyber physical contingency ranking that indicates how serious is the system's current status. In particular, given the cyber and power network topologies, CPIndex creates the cyber-physical model of the underlying system that takes into account the interdependencies among the cyber assets and power system components. CPIndex makes use of the generated model to proactively explore potential next immediate security incidents in the near future and calculate how "close" the system's current security state is to a catastrophic system failure. A numerical score is calculated to that end that could be used to order the cyber and physical contingencies. Our experimental results in a real-world test-bed shows that CPIndex can calculate the contingency scores very efficiently.

Acknowledgements

This work was supported in part by the Department of Energy (DoE) Award Number DE-OE0000097 (Trustworthy Cyber Infrastructure for the Power Grid).

References

[1] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: communication technologies and standards," *Industrial informatics, IEEE transactions on*, vol. 7, no. 4, pp. 529–539, 2011.

[2] H. Farhangi, "The path of the smart grid," *Power and Energy Magazine, IEEE*, vol. 8, no. 1, pp. 18–28, 2010.

[3] S. Blumsack and A. Fernandez, "Ready or not, here comes the smart grid!" *Energy*, vol. 37, no. 1, pp. 61–68, 2012.

[4] S. M. Amin, "Electricity infrastructure security: Toward reliable, resilient and secure cyber-physical power and energy systems," in *Power and Energy Society General Meeting, 2010 IEEE*. IEEE, 2010, pp. 1–5.

[5] X. Li, X. Liang, R. Lu, X. Shen, X. Lin, and H. Zhu, "Securing smart grid: cyber attacks, countermeasures, and challenges," *Communications Magazine, IEEE*, vol. 50, no. 8, pp. 38–45, 2012.

[6] NERC, 2009, reliability Standards for the Bulk Electric Systems of North America.

[7] Y. Mo, T.-J. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "Cyber–physical security of a smart grid infrastructure," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, 2012.

[8] D. Kundur, X. Feng, S. Mashayekh, S. Liu, T. Zourntos, and K. L. Butler-Purry, "Towards modelling the impact of cyber attacks on a smart grid," *International Journal of Security and Networks*, vol. 6, no. 1, pp. 2–13, 2011.

[9] T. M. Chen, J. C. Sanchez-Aarnoutse, and J. Buford, "Petri net modeling of cyber-physical attacks on smart grid," *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 741–749, 2011.

[10] B. Falahati, Y. Fu, and L. Wu, "Reliability assessment of smart grid considering direct cyber-power interdependencies," *Smart Grid, IEEE Transactions on*, vol. 3, no. 3, pp. 1515–1524, 2012.

[11] A. Hahn and M. Govindarasu, "Cyber attack exposure evaluation framework for the smart grid," *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 835–843, 2011.

[12] G. Casella and E. I. George, "Explaining the gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. 167–174, 1992.

[13] S. Zonouz, R. Berthier, H. Khurana, W. Sanders, and T. Yardley, "Seclius: An information flow-based, consequence-centric security metric," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[14] G. Casella and E. I. George, "Explaining the gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. pp. 167–174, 1992.

[15] T. Ernster and A. Srivastava, "Power system vulnerability analysis-towards validation of centrality measures," in *Transmission and Distribution Conference and Exposition (T&D), 2012 IEEE PES*. IEEE, 2012, pp. 1–6.

[16] R. Bellman, "On a routing problem," DTIC Document, Tech. Rep., 1956.

[17] Ernster, "Power system vulnerability analysis: A centrality based approach utilizing limited information," Master's thesis, School of Electrical Engineering and Computer Science, Washington State University, 2012.

[18] "Trustworthy Cyber Infrastructure for the Power Grid (TCIPG): http://www.tcipg.org," 2010.

[19] C.-W. Ten, G. Manimaran, and C.-C. Liu, "Cybersecurity for critical infrastructures: attack and defense modeling," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 40, pp. 853–865, July 2010.

[20] H. Hadeli, R. Schierholz, M. Braendle, and C. Tuduce, "Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration," in *Proceedings of the IEEE International Conference on Emerging Technologies & Factory Automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1189–1196.

[21] K. Davidson, "The x-10 tw523 two-way power line interface," *Circuit Cellar*, vol. 5, pp. 34–35, 1988.