Article

Protection of Energy Network Infrastructures Applying a Dynamic Topology Virtualization

Maxim Kalinin *, Dmitry Zegzhda and Evgenii Zavadskii

Cybersecurity Department, Peter the Great St. Petersburg Polytechnic University, 195251 St. Petersburg, Russia; sci@ibks.spbstu.ru (D.Z.); prj@ibks.spbstu.ru (E.Z.)
* Correspondence: max@ibks.spbstu.ru

Abstract: Rapid progress of computing and info-communication technologies (ICT) has changed the ecosystem of power production and delivery. Today, an energy network is a complex set of interrelated devices and information systems covering all areas of electric power operations and applying ICT based on open standards, such as IEC 60870, IEC 61850, and IEC 61970. According to IEC 62351, the energy networks are faced with high cybersecurity risks caused by open communications, security requirements rarely considered in the energy facilities, partial and difficult upgrades, and incompatibility of secure tools with industrial solutions. This situation results in new security challenges, e.g. denial of service attacks on the connected controllers, dispatching centers, process control systems, and terminals. IEC 62351 describes possible ways to comprehensive security in the energy networks. Most of them used in traditional networks (e.g., firewalls, intrusion detection systems) can be adapted to the energy networks. Honeypot systems as a protection measure help us to mitigate the attacks and maintain necessary security in the networks. Due to the large scale of an energy network and heterogeneity of its components, a new design, deployment, and management strategy for the honeypot systems are required. The paper suggests a new method for organizing a virtual network infrastructure of a hybrid honeypot system and a dynamic management method that adapts the network topology to the attacker’s actions according to the development graph of potential attacks. This technique allows us to dynamically build virtual networks of arbitrary scale. Because of the similarity of the virtual network to the virtualized origin and providing the level of interactivity of its nodes corresponding to real devices, this technique deploys an energy network indistinguishable from the real one for the attackers. A prototype of our honeypot system has been implemented, and experiments on it have demonstrated the more efficient use of the computing resources, the faster reaction to the attacker’s actions, and the deployment of different sizes of virtual networks for the given limits of the computing resources.

Keywords: hybrid honeypot; dynamic virtual network infrastructure; attack graph

1. Introduction

An electric power system is a complex technical infrastructure responsible for the control and monitoring of technological processes, the physical nature of which determines the high speed of their flow. Components of this infrastructure have undergone significant changes over the past decades due to the development of info-communication technologies (ICT). The introduction of new hardware and software has provided new opportunities to automate control, improve the efficiency and reliability of energy networks, and increase the speed of reaction to critical incidents. The traditional power grid has been transformed into a smart grid [1]. In conventional energy networks, cybersecurity has a low priority as control systems were developed like isolated solutions. Modern industrial control systems (ICS) and smart industrial devices such as programmable logic controllers (PLCs), operator control panels (HMIs), intelligent electronic devices (IEDs), and remote-control modules (RTUs) implement open standards IEC 60870, IEC 61850, and IEC 61970. Global integration
and interconnectivity with corporate services, coupled with high levels of openness, have increased cybersecurity risks for modern energy networks.

Correct operation of a control system directly depends on the security of the network infrastructure components on which it is built. Security flaws are regularly found in various software, which is potentially vulnerable. According to the 2021 Claroty Team 82 report [2], 637 vulnerabilities were identified in ICS devices from 76 vendors (Figure 1). The authors note that about 80% of them were discovered by third-parties unaffiliated with the manufacturing companies. The number of zero-day vulnerabilities is never known.

![Figure 1. Five leading manufacturers affected by vulnerabilities.](image)

Conventional security mechanisms (e.g., firewalls, intrusion detection and prevention systems (IDS/IPS), antivirus solutions) can be adapted to protect the energy network infrastructure. However, these protection tools do not allow tracking APT attacks, during which attackers use various undeclared software features, configuration errors, zero-day vulnerabilities, and other weaknesses of the system components that had not been foreseen during implementation. The solution to this is the introduction of honeypot systems.

A honeypot system provides the following capabilities:

- collecting information on the attackers’ tactics and techniques to improve security incident response strategy;
- identifying new attack vectors and exploitation of zero-day vulnerabilities;
- detecting potential intrusions;
- presenting a response time required to counteract the attacks and protect the system;
- hiding the real network infrastructure (network masking).

We suggest a formal description of the network infrastructure and the process of attack. On this basis, a new method for organizing a virtual network infrastructure as a honeypot and a dynamic management method that adapts a network topology to the attacker’s actions according to the development graph of potential attacks are proposed. This technique ensures the efficient use of the computing resources, which allows us to build a virtual network of different sizes. Because of the similarity of the virtual network behavior to the origin and providing the level of interactivity of its nodes corresponding to the co-work of real devices, our method allows us to deploy a network indistinguishable from a real one, from the attacker’s point of view. The prototype of our honeypot system has been implemented, and the provided experiments have shown the more efficient use of the computing resources, the faster response to the attacker’s actions, and deployment of different sizes of virtual networks for the given limits of the computing resources.

The novelty of our contribution is the proposed method of dynamic virtual network topology management based on a potential attack graph. Our method deploys a virtual network infrastructure of arbitrary scale, according to which the virtual network infrastructure changes its configuration in case of limited computing resources in the energy network dynamically adapting to the attacker’s activity.

The rest of the paper is organized as follows: Section 2 observes the related works and presents our method for dynamic virtual topology management on a honeypot system based on a potential attack graph; Section 3 discusses our implementation and experimental results; and, finally, Section 4 concludes our work.
2. Materials and Methods

2.1. The Related Works

The general classification of honeypot systems is presented in [3]. Honeypots can be divided according to the following characteristics:

- interaction: high interactive or low interactive;
- configuration: static or dynamic;
- scalability;
- resources required to deploy a honeypot system.

Low interactive honeypot systems are utilities that use various mechanisms to mimic the behavior of a real operating system or individual service. Due to its lightweight, a virtual network can be implemented in an arbitrarily scalable way by running multiple instances of the simulator. HoneyD simulator [4] is an example of this type of a honeypot. When interacting with this honeypot, the attacker’s actions are limited—he can only scan and connect to the open ports specified in the configuration file, and execute requests, the responses to which are implemented in scripts. Thus, the disadvantages of these honeypot systems are easy detection and a small amount of information collected about the attacker.

To increase the amount of gathered information about the attacker, the research works [5–13] propose honeypot systems with the possibility of self-adaptation, for which methods of game theory, Markov decision-making and machine learning are applied. High interactive honeypot systems are real operating systems or a set of real services for attackers. Highly interactive systems collect a large amount of information about the attacker: the data collection tools, and malware used, operating system or service vulnerabilities exploited, and methods of movement within the network. This type of honeypot requires significant computing resources to provide scalability, for the nodes are virtual machines or real devices. Ref. [14] proposes the solution to the issue of scalability of high interactive honeypots. IoTPOT implements the Telnet protocol and consists of many low interactive components that simulate real service by sending TCP responses and banners provided by the script, and a high interactive component to which commands unknown to simulators are redirected. Guarnizo et al. [15] and Tambe et al. [16] have proposed a method for creating a high interactive honeypot, in which scalability is ensured by using VPN tunnels, allowing the nodes to appear as several virtual nodes with different IP addresses. The disadvantage of this approach is the inability of the network to hold an attacker’s attention. By connecting to virtual nodes, which are implemented by one real node in the network, it is possible to determine that it has already been attacked before.

Molina et al. [17] have proposed a high interactive honeypot system that uses network function virtualization (NFV) and software-defined network (SDN). NFV is used to deploy network nodes with the possibility of their dynamic configuration and reconfiguration, and SDN is applied to change the configuration of the virtual network, filtering traffic and its redirection between the real network and the honeypot system. The deployment of this honeypot system requires significant computing resources to run the virtual machines.

In contrast to the typical implementation of a dynamic honeypot, ref. [18] proposes changing not the network configuration, but the position of genuine and spoofed services. To access the particular service, they use a secure communication protocol, which provides a mechanism for updating the IP address of the target node. An attacker cannot use this protocol because of a lack of cryptographic parameters and as a consequence has access to the service within a certain time interval between changes of system services positions. By the example of DoS attack, the effectiveness of this honeypot was demonstrated—smoothing load schedule on a node by distributing it among several nodes. The disadvantage of this approach is the complexity of its integration into a real network infrastructure because of the necessity to use a proprietary protocol, which requires each client to additionally listen to the port to get the service IP address updates.
The vast majority of honeypot systems for the energy networks are low interactive as presented in [19–25]. This is due to the high cost of real ICS devices and the inability to create virtual nodes running their firmware. Ref. [26] proposes a honeypot system architecture, which in addition to the virtual machine with Conpot [27] simulating the RTU devices includes virtual and real HMI panels to perform network interaction with RTU, thus betraying the realism of the network infrastructure.

Existing implementations of the high interactive honeypot for the energy networks use an architecture based on the real control devices (controllers) [28,29] or architecture with many low interactive nodes that proxy traffic to real ICS devices [30].

2.2. Motivation

The actual infrastructure of the energy network includes a large number of different devices. The honeypot systems discussed earlier do not allow deploying a network of a similar or larger scale due to the lack of tools for ICS virtualization. However, access to these tools becomes necessary to improve the security of the energy networks. The following paper proposes a new method of dynamic network topology virtualization that allows the deployment of a network of arbitrary scale, considering the limited computing resources in the energy networks.

2.3. A Hybrid Honeypot

One of the main objectives of the honeypot systems is to divert an attacker from the resources of the protected network. In order to provide this function, a semblance of the virtual and real networks is necessary. This can be achieved by using the high interactive nodes—the virtual machines. However, with the limited computing resources, it is impossible to implement a network of arbitrary scale using the nodes of this kind, as the maximal possible number of the simultaneously working machines is fixed. Thus, it is necessary to use a hybrid honeypot system.

A hybrid honeypot system is a system consisting of nodes of two types (Figure 2):

- simulated node (simulator): a software tool that implements the TCP/IP stack of a real operating system and emulates interaction with “working” services acting as the node;
- virtualized node: a virtual machine with the operating system installed on it, which allows us to exactly replicate the node of the real network.

![Figure 2. Hybrid honeypot system virtual network schema: blue—the simulator, dark blue—the virtualized node.](image)
The high interactivity of the hybrid honeypot system is ensured by switching the mode of the nodes according to the condition of their reachability for an attacker:

- simulator mode: a node at this stage of an attack is not reachable by an attacker;
- virtualizable mode: in the current state of the system, a node can be reached by an attacker.

The hybrid honeypot system responds to an attacker by changing the mode of the operation of the virtual network nodes.

Formally, a network consists of the complete graphs $K_m$ (Figure 2), as each of the $m$ nodes can contact its neighbors directly (ping) in the subnetworks. Generally, a network can be represented as a finite set $G$ of the complete graphs $K_m$, the sets of vertices of which intersect:

$$G = \{K_m^{(i)} \mid V(K_m^{(i)}) \cap V(K_m^{(j)}) = \emptyset, i \neq j\},$$

where $V$ is the set of vertices of the graph, $n$ is the number of the networks in $G$.

$$LN(G) = \{K_m^{(i)}, 1 \leq i, j \leq n\},$$

$$LN_i(G) = K_m^{(i)},$$

$$\#LN(G) = n.$$  

The vertices which belong to several complete graphs will be called a boundary node

$$V_{\text{inter}}(G) = \{v \mid v \in V(K_m^{(i)}) \& \ldots \& v \in V(K_m^{(k)}) \& K_m^{(j)} \subset G \& \ldots \& K_m^{(k)} \subset G, 1 \leq k \leq n\}.$$  

To renumber the boundary vertices for each $j$-th vertex, set a binary vector $x_j$ of length $n$, where the $i$-th element corresponds to $LN_i(G)$. The units are set at the positions corresponding to the local networks to which this vertex belongs.

During the attack, the attacker compromises the network nodes. Hence, the set of the compromised nodes is the path of the attack:

$$\text{path} = \{v \mid v \in V(G) \& \text{compromised}(v)\}.$$  

The state of the hybrid honeypot system is described by the set of the compromised nodes together with the set of the nodes that could potentially be compromised at a given time:

$$x = \bigcup_{k \in \{i \mid \text{compromised}(v), 1 \leq i \leq \#V_{\text{inter}}(G)\}} x_k,$$

$$S_j = \{LN_j(G) \mid 1 \leq j \leq n \& x[j] = 1\}.$$  

The correct work of the system is provided if at each possible state of the system the consumed resources do not exceed the physically accessible ones:

$$\forall i \text{ RC}(S_i) \leq \text{RC}_{\text{system}}, \text{RC}(S_i) = \sum_{l=0}^{\#V(S_i)} \text{RC}(v_l).$$

Thus, to evaluate the computing resources required for the hybrid honeypot system, it is necessary to specify the condition for changing the mode of the nodes from the simulated to the virtualizable mode.

**Condition 1.** The mode change from the simulated to the virtualizable node is performed in case of compromise of the adjacent node according to the virtual network infrastructure graph.

The development of the worst-case scenario in terms of the resources required, given the condition 1 introduced, is shown in Figure 3. An attacker had initially compromised the input node A. The response of the honeypot system to this action is to put the nodes
B, C, and D adjacent to it into the virtualizable mode. D was then compromised and the nodes on subnet LN₃ became reachable to the attacker. The response to this is to change their mode of operation to the virtualizable mode. Subsequent hijacking of the node H leads to the virtualization mode of the nodes in the subnet LN₅.

Figure 3. Schema of the virtual network with three compromised nodes: dark blue—the virtualizable nodes, red—the compromised virtualizable nodes, blue—the simulators.

In this path of the attack, the honeypot system has moved to state S, in which 17 nodes are virtualizable. Hence, \( RC(S) \) is equal to the sum of resources consumed by each of the virtual machines. To reduce the number of the simultaneously working machines, we introduce an additional condition 2.

**Condition 2.** When an attacker moves to another subnet, the mode of operation of the nodes in the considered subnet can be changed to the simulated node.

The attacker has sequentially compromised the nodes A, D, and H. According to the introduced condition, he can continue to propagate in the network LN₃ or LN₅. B and C can be put into the simulated mode, thus reducing the number of the virtualized machines in the system state S to 15. Note that the attacker, interacting with node H, still has direct access to the compromised nodes A, D, and at any time and can attack the nodes B and C. Therefore, the application of this condition is inadmissible.

Research [14] has pointed out that devices with known vulnerabilities tend to be attacked more frequently than nodes with unknown ones. Thus, to reduce the number of the single-moment virtual machines running, it is necessary to integrate the specially prepared vulnerable nodes into the virtual network infrastructure graph when designing it. The intruder moves within the network using the techniques given in the MITRE ATT&KE knowledge base [31]. By integrating all possible paths of the attacker’s transitions between the vulnerable nodes, a potential attacks development graph is built.

**Condition 3.** The node, which according to the potential attacks’ development graph cannot be attacked, remains simulated.

Let the potential attacks development graph includes all the boundary nodes, as well as a number of nodes from subnets: one node from the network LN₂, LN₃, LN₄ and three
nodes from the network $LN_5$. Then in the above attack path the honeypot system will perform the following actions Figure 4:
1. Node A compromised—change the mode of the nodes B, C and D to virtualizable.
2. Node D compromised—the nodes H and M are switched to virtualizable mode.
3. Node H compromised—the nodes O, P and Q became virtualized.

![Figure 4. Schema of a virtual network with three compromised nodes: dark blue—the virtualizable nodes, red—the compromised virtualizable nodes, blue—the simulators.](image)

The graph describing the state of the honeypot system $S$ under condition 3 contains only 9 nodes, which are the virtual machines. The number of nodes in the potential attacks' development graph can be chosen by taking into account the available computing resources and the time requirements for the personnel to develop a response.

Suppose an attacker explored subnets $LN_3$ and $LN_5$ and switched to compromise the node B, no longer interacting with the nodes D and H. Then to access the non-adjacent node H the attacker must start interacting with the adjacent node D. Based on this, let us formulate an additional condition.

**Condition 4.** The node that is not adjacent to the node with which the attacker interacts, in the absence of interaction within a given period of time, must be transferred to the simulated node.

Thus, if the attacker does not interact with the nodes D and H, then the nodes M, H, O, P and Q will be transferred to the simulated node. The state $S$ of the honeypot system will change to $S'$, which includes only 4 nodes that are the virtual machines. Note that when the compromised node H is switched to the simulator mode, the state of the virtual machine must be saved to recover later.

### 2.4. Potential Attacks Development Graph

The potential attacks development graph can be represented by several methods. Figure 5 shows a network diagram. Let us consider several common types of the attack graphs on this sample [32]:

1. **Full attack graph** is a graph representing every possible sequence of attacks. The nodes of the graph are states and the edges are transition conditions. As the number of states
increases, the size of this graph increases rapidly as $O(n!)$. Figure 6a shows part of the complete attack graph.

2. A prediction graph is a graph in which a new node is added provided that there are no ancestors using the same vulnerability to reach the same state. The value of nodes and edges is similar to the full graph. This graph eliminates most of the redundant structure of the full graph, as a consequence, its construction takes less time, and correctly predicts the impact of removing any of the vulnerability instances in the network. Figure 6b shows an example of a prediction graph.

3. A graph with several preconditions (MP-graph) is a graph that consists of nodes of three types: states, preconditions, and vulnerabilities. The edges indicate only the direction of transition between the nodes of the graph. In addition to the information contained in the full graph and the prediction graph, this graph also shows all the states that can be reached from the current state using cycles. Figure 6c shows an MP-graph in which the only precondition is the reachability of the network nodes.

Therefore, among the given types of attack graphs, the MP-graph is the fastest, it fully represents possible transitions between the states without structure redundancy, and it can be converted into a complete graph or prediction graph.

3. Results

3.1. Implementation of a Hybrid Honeypot Based on a Potential Attack Development Graph

The architecture of the implemented prototype of the hybrid honeypot system based on the development graph of potential attacks is shown in Figure 7.

The action collection component is responsible for detecting events produced by an attacker in a system under his control and sending them to the management server.
Figure 7. The hybrid honeypot architecture based on the potential attack development graph.

The system management component performs the simulator and the virtual machine management based on the events received from the action collection components. The simulated node management component implements starting and stopping simulators as requested by the control component.

The virtualizable node management component performs operations on the virtual machines of the implemented network.

Based on the input configuration file, which describes the development graph of potential attacks for the implemented network, the developed system deploys the virtual network and changes its structure in terms of node type changes.

3.1.1. The Actions Collection Component

The action collection component consists of two tools: the network client and the kernel event collection tool. The kernel event collection tool is an open-source project Fibratus [33], which can be used to detect the events related to processes, threads, dynamic libraries, files, registry keys, networking, and working with the system object identifiers.

The HTTP client communicates with the system management component to initialize the list of expected events and send the detected user actions, and runs the means of collecting user actions and processing the monitored events according to the received list. The following API requests are used when communicating with the server: GET request /agent/init/—initialization of the list of events, POST request /agent/event/—events sending. The header of each request specifies the identifier of the sending node.

The tools used are part of the honeypot prototype and do not provide a sufficient level of indistinguishability of a virtual network node from a real device.

3.1.2. The Simulated Node Management Component

The simulated node management component is the HTTP server implemented using the FastAPI web framework [34] and the Uvicorn asynchronous server [35]. The function of this component is to manage HoneyD simulator instances [4].

HoneyD is a daemon that creates simulated nodes on the network, which can run different operating systems and contain different services by using the nmap-os-db fingerprint database of the Nmap utility and extending it with Perl and sh scripts [36].

The simulated node management component runs each node simulator as a separate process, allowing each node to be disabled and enabled independently.
To interact with the management server, this component provides the API requests shown in Table 1.

**Table 1.** Requests of the implemented API of the simulated node management component.

<table>
<thead>
<tr>
<th>Request type</th>
<th>Path</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/new/</td>
<td>[&quot;id&quot;: node ID, &quot;path&quot;: path to the HoneyD configuration file]</td>
<td>Initialization of the structure describing the simulators and representing a dictionary, where the key is the node ID, the value is the path to the configuration file.</td>
</tr>
<tr>
<td>POST</td>
<td>/start/</td>
<td>List of node IDs</td>
<td>Run the simulators whose identifiers were passed by the system management component, and save the Popen class object in the structure described earlier.</td>
</tr>
<tr>
<td>POST</td>
<td>/stop/</td>
<td>List of node IDs</td>
<td>Stops active simulator processes advising the transmitted identifiers.</td>
</tr>
</tbody>
</table>

3.1.3. Configuration File

The configuration file contains a potential attack development graph for the virtual network being implemented in JSON format, whose structure is described by a grammar developed using the Pydantic module for the Python programming language. Figure 8 shows an example of a file describing a virtual network of four nodes.

```json
[
  {
    "id": 0,
    "ip": ["192.168.1.33/24", "192.168.2.57/24"],
    "simulator": "/home/user/Downloads/GDS/sim_assistant/configs/10.config",
    "vn": {
      "uuid": "i1dd99331-0455-4b7f-bf51-1c3748918e79",
      "initial_state": "power on"
    },
    "parent_id": null,
    "children": [1, 2],
  },
  {
    "id": 1,
    "ip": ["192.168.1.36/24", "192.168.2.57/24"],
    "simulator": "/home/user/Downloads/GDS/sim_assistant/configs/10.config",
    "vn": {
      "uuid": "6ee81d43-c9ce-41eb-b8ea-933f66fa3db5",
      "initial_state": "power off"
    },
    "parent_id": "simulator",
    "parent_id": 0,
    "children": [3]
  },
  {
    "id": 2,
    "ip": ["192.168.1.37/24", "192.168.2.57/24"],
    "simulator": "/home/user/Downloads/GDS/sim_assistant/configs/20.config",
    "vn": {
      "uuid": "c9741781-f8cf-4f74-a21d-835536bcf1bc",
      "initial_state": "power off"
    },
    "parent_id": "simulator",
    "parent_id": 0
  },
  {
    "id": 3,
    "ip": ["192.168.2.59/24"],
    "simulator": "/home/user/Downloads/GDS/sim_assistant/configs/40.config",
    "vn": {
      "uuid": "a6b1c312-bd1c-46ac-9db6-2fe1763867ac",
      "initial_state": "power off"
    },
    "parent_id": "simulator",
    "parent_id": 1
  }
]
```

**Figure 8.** Configuration file describing the development graph of potential attacks.

The node with identifier 0 has no parent (parent_id: null) and has two children (children). Since this node is the entry point of the network, only a virtual machine with the universal unique identifier 1dd99331-0455-4b7f-bf51-1c3748918e79 is defined.
The node with identifier 1 is a boundary node, so the “ip” list contains two IP addresses. The field “simulator” contains the path to the simulator configuration file. Since the initial state of the node is simulator (initial_system: simulator), the virtual machine is initially disabled. If there is no communication between the attacker and the node operating in the virtualized mode, after the timeout interval specified in the “timeout” parameter the node will be switched to the simulated node. If a node has been compromised, a snapshot of its state will be created before shutting down the virtual machine.

3.1.4. The Virtualizable Node Management Component

The virtualizable node management component performs operations on the virtual machines on the implemented network and communicates with the simulated node management component to synchronize with the current active node type. Virtual machines are managed using the Libvirt-python 6.10 module, which provides an interface for remote interaction with the hypervisor. To improve the experience of interacting with this API, the Connect, and Instance classes have been implemented.

In order to establish a connection to the hypervisor, a username and password specified when configuring the hypervisor must be provided. This component is implemented as a Worker class, which is initialized by an object of the ConnectionSetup class that contains the KVM connection parameters.

The virtualizable node management component performs the tasks placed by the system management component one by one. Each task is described by the structure shown in Figure 9.

```python
class Action(BaseModel):
    operation: PossibleOperation
    snapshot_name: Optional[str]

class Task(BaseModel):
    sim_id: Optional[int]
    sim_action: Optional[str]
    vm_uuid: str
    vm_action: Action
```

**Figure 9.** Task structure of the virtualizable node management component.

The “sim_id” and “sim_action” fields are optional, because some nodes may not have a simulator configuration, like the node with identifier 0 described in the previous subsection. The “vm_action” field contains possible virtual machine operations:

- **power on:** enabling the specified virtual machine;
- **power off:** shutting the node down with an ACPI command;
- **suspend:** suspends the virtual machine, with the current state saved on the hard disk;
- **resume:** reactivation of the node by restoring the previously saved state;
- **revert:** restores the machine to a specified snapshot, which name is specified in the “snapshot_name” field of the subobject of the Action class;
- **force off:** instantly disconnects a node.

When a task is received, a virtual machine operation is initially performed and then a /stop/ or /start/ POST request is sent to the simulated node management component to stop or start, respectively.

The method of pausing and resuming the virtualizable nodes with which the attacker has interacted is selected to create a snapshot of the state and then restore it. This method preserves all the results of the attacker’s work on a given operating system while providing a faster recovery speed than a normal startup.

3.1.5. The System Management Component

The system management component inputs a configuration file with the graph of potential attacks and the file with authentication data for connecting to the hypervisor.
When parsing the potential attack evolution graph, the structure is checked for correctness using the grammar described earlier and for consistency of the node configurations specified in it according to the following rules:

1. There should be no nodes without at least one type of node implementation.
2. When the initial node type (initial_state) is specified:
   - in the absence of a given simulator, the original node type must not be a simulator;
   - in the absence of a given virtual machine, the initial node type must not be a virtual machine;
   - if the base state of the virtual machine is “power on”, the initial node type cannot be specified as a simulator.
3. If both a virtual machine and a simulator are specified in the configuration, the original node type must be specified.
4. If no simulator is specified, the initial state of the virtual machine cannot be “power off”.

After checking the configuration file, the simulated node management component is initialized with the POST request /new/, and the simulators required at the initial stage are launched.

Several instances of the virtualizable node management component are created to manage virtual machines. Their number affects the speed of execution of the virtual machine operations, since the operation to return to the snapshot is blocked. Therefore, this parameter should be chosen according to the amount of the available resources and the development graph of potential attacks on the implemented virtual network.

Interaction of the system management component with instances of the management component of the virtualized node is performed by means of a task queue (Figure 10).

![Figure 10. Schema of interaction between the system management component and instances of the management component of the virtualized node.](image)

This component is an HTTP server that provides an API for interacting with the action collection component:

- initializing a list of expected events when the virtualizable node starts up;
- receiving data about events caused by an attacker’s actions.

When events are received, a task is generated according to the potential attack development graph, which is sent to the queue for execution by one of the instances of the virtualizable node management component.

3.2. Testing a Hybrid Honeypot Based on a Graph of Potential Attack Developments

In order to compare the parameters of a hybrid honeypot system operating under the specified conditions of the node mode change, the simulator has been developed that uses the specified server characteristics, virtual network infrastructure graph, attack path description, time of operations performed by an attacker and a hybrid honeypot system to analyze the honeypot system behavior and generate charts of resource consumption,
number of running virtual machines and presence of system reaction delays versus time. The response is understood as a set of actions that are performed as a response to an attacker’s actions.

The virtual network infrastructure graph with renumbered vertices is shown in Figure 11. The development graph of potential attacks includes the following nodes—\{0, 1, 2, 3, 7, 8, 10, 15, 16, 17, 20, 21, 25, 27\}.

![Figure 11. Virtual network graph of the hybrid honeypot system in the initial state.](image)

Tables 2–4 show the characteristics under which the simulations were performed for each of the given conditions of the hybrid honeypot system. The idle time parameter of the virtualized node (timeout) is equal to 4 seconds.

A standard attack scenario consists of the following steps: compromise the host, log on to it, download and run the necessary utilities (netcat, nmap, and others), and scan the subnet to identify the next target. In case of moving to an already compromised node, you only need to run the utilities.

**Table 2.** The specified characteristics of a virtual server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU, %</td>
<td>400</td>
</tr>
<tr>
<td>RAM, Mb</td>
<td>32,000</td>
</tr>
<tr>
<td>Disk I/O, %</td>
<td>100</td>
</tr>
</tbody>
</table>

**Table 3.** Possible attacker’s actions and execution time.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time, Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compromise</td>
<td>2</td>
</tr>
<tr>
<td>Securing</td>
<td>10</td>
</tr>
<tr>
<td>Loading the necessary tools</td>
<td>30</td>
</tr>
<tr>
<td>Launching tools</td>
<td>10</td>
</tr>
<tr>
<td>Scanning a subnet</td>
<td>5 per subnet node</td>
</tr>
</tbody>
</table>
Table 4. Actions, time and resources of the hybrid honeypot system.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time, Sec</th>
<th>CPU, %</th>
<th>RAM, Mb</th>
<th>Disk I/O, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power on</td>
<td>7</td>
<td>70</td>
<td>4000</td>
<td>15</td>
</tr>
<tr>
<td>Restore from snapshot</td>
<td>3</td>
<td>5</td>
<td>500</td>
<td>5</td>
</tr>
<tr>
<td>Working</td>
<td>—</td>
<td>30</td>
<td>2000</td>
<td>5</td>
</tr>
<tr>
<td>Snapshot creation</td>
<td>4</td>
<td>5</td>
<td>500</td>
<td>7</td>
</tr>
<tr>
<td>Shutting down</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Let the attack path is $path_1 = \{0, 1, 7, 8, 20, 3, 16, 15, 27\}$. Figure 12 shows the simulation result for the hybrid honeypot system operating according to the first condition (a) and the combination of the third and fourth conditions (b).

![Figure 12](image)

According to the results, the state $S$ denotes the hybrid honeypot system, which changes the mode of the nodes according to the first condition, before the compromise of node 8 included 12 virtual machines. When boundary node 8 is compromised, 4 more $LS_4$ subnet nodes must be started. However, due to a lack of resources, the system stopped responding to the attacker.

The hybrid honeypot system running conditions 3 and 4 reacts correctly to the attacker’s movements through the virtual network. At 420 seconds, five virtualizable nodes of subnets $LS_1, LN_2, LN_4$ are switched to the simulated mode because the attacker has switched to another subnet $LS_3$.

Figure 13 shows the simulation results for $path_2 = \{0, 1, 7, 8, 20, 3, 16, 15, 27, 1, 8, 18\}$. The re-entry of nodes 1 and 8 is due to the need to rerun utilities loaded after the compromise.

There is a delay in the graph of the system response delays when attempting to compromise node 18—it was necessary to switch the node’s mode of operation to virtualizable. This is due to the fact that this node is not included in the graph of potential attacks and the hybrid honeypot system had not enabled a virtual machine for in advance.
4. Discussion

Ensuring the security of modern energy networks is an important challenge, because a successful attack on smart grids and smart power plants can disrupt fast-flowing physical processes, destroy expensive components and disable important infrastructure.

Based on the formal specification of the network infrastructure and the attack process, a new method was developed to dynamically manage the virtual network infrastructure of a hybrid honeypot system based on a potential attack development graph, which, by providing high interactivity of nodes and similar network scale, is indistinguishable from the real ones, from the attacker’s point of view.

In our study, the existing methods for implementing the honeypot systems for the energy networks have been already observed and reviewed. The main factor constraining the development of this area is the closed nature of ICS device manufacturers. However, with the emergence of enterprise services and global integration, openness is necessary to improve the security of network infrastructure. Table 5 summarizes our work comparing our solution with the existing honeypot systems examined in Section 2.1.

Table 5. Comparison of the honeypot solutions applied at the energy networks.

<table>
<thead>
<tr>
<th>Name</th>
<th>Simulators</th>
<th>Virtual Machines</th>
<th>Real Devices</th>
<th>High Interactivity</th>
<th>Network Scalability</th>
<th>Avg. Resource Consumption, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShaPe [22]</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>IoTPOT [14]</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>100</td>
</tr>
<tr>
<td>CryPLH [23]</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>+</td>
<td>100</td>
</tr>
<tr>
<td>Electrical Substation Honeynet [25]</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>100</td>
</tr>
<tr>
<td>ICS honeypot [30]</td>
<td>+</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>Our solution</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>+</td>
<td>48</td>
</tr>
</tbody>
</table>
Ensuring high interactivity, understood as the ability to anchor an attacker to each node of the implemented network, is an important feature of a honeypot system, on which the ability to solve the problems presented in Section 1 depends. Our solution, unlike others, allows entrenchment on the attacked network nodes by granting access to an individual virtual machine rather than a simulator or a single virtual machine, satisfying this qualitative characteristic.

The energy networks may have a significant size. Therefore, the scalability with respect to computing resources requirement is also an important comparative feature. The developed method allows the more efficient usage of the resources, compared with those shown in Table 5. On average, under the attack conditions only 14 of 29 nodes of the erected virtual network will be virtual machines, which is 48% of the total network size (this is presented in Section 3.2).

The proposed honeypot system builds the section of the virtual network infrastructure visible to the attacker; at the same time, other solutions build the whole network or the nearest portion of the infrastructure. Therefore, the proposed approach allows the building of the more realistic (for the attacker) virtual model of the under-attacking network.

In the future, we are focusing on solving the drawback of the proposed method, which is the delay in case an attacker chooses a node that is not provided by the potential attack graph as a target for compromise. We also plan to develop methods to analyze the actions collected from virtual network nodes, detect compromised scenarios, and further optimize the algorithm of the node mode changing. Our method can be also applied to industrial wireless sensor networks (WSNs) in combination with security and anomaly detection techniques, given their [37–39] characteristics.

Author Contributions: Conceptualization, D.Z. and E.Z.; methodology, validation, M.K.; software, E.Z.; formal analysis, resources, D.Z.; writing—original draft preparation, E.Z.; writing—review and editing, M.K.; visualization, E.Z.; supervision, D.Z.; project administration, funding acquisition, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: The research is funded by the Ministry of Science and Higher Education of the Russian Federation as part of the World-class Research Center program: Advanced Digital Technologies (contract No. 075-15-2022-311 dated 20 April 2022).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- ICS: industrial control system
- ICT: infocommunication technologies
- PLC: devices such as programmable logic controller
- HMI: operator control panel
- IED: intelligent electronic device
- RTU: remote control module
- VPN: virtual private network
- SDN: software-defined network
- NFV: network functions virtualization

References


