Journal of Computer Virology and Hacking Techniques (2019) 15:109–125
https://doi.org/10.1007/s11416-018-0325-y

ORIGINAL PAPER

Malware propagation in smart grid networks: metrics, simulation and comparison of three malware types

Peter Eder-Neuhauser[1] · Tanja Zseby[1] · Joachim Fabini[1]

Received: 12 August 2017 / Accepted: 11 August 2018 / Published online: 28 August 2018

Abstract
Smart grids utilize communication technologies that make them vulnerable to cyber attacks. The power grid is a critical infrastructure that constitutes a tempting target for sophisticated and well-equipped attackers. In this paper we simulate three malware types capable of attacking smart grid networks in the ns3 simulation environment. First, an aggressive malware type, named the *pandemic* malware, follows a topological-scan strategy to find and infect all devices on the network in the shortest time possible, via a brute force approach. Next, the more intelligent *endemic* malware sacrifices speed for stealthiness and operates with a less conspicuous hit-list and permutation-scan strategy. Finally, a highly stealthy malware type called the *contagion* malware does not scan the network or initiate any connections but rather appends on legitimate communication flows. We define several metrics to express the infection speed, scanning efficiency, stealthiness, and complexity of malware and use those metrics to compare the three malware types. Our simulations provide details on the scanning and propagation behavior of different malware classes. Furthermore, this work allows the assessment of the detectability of different malware types.

Keywords Malware attacks · Smart grids · Communication networks · Anomaly detection · Network security

1 Introduction

Cyber attacks targeting critical infrastructures evolved to a major threat and featured several occurrences in the news media [8,11]. Consequently, utility companies have started upgrading their control networks with enhanced security measures because the power grid is increasingly fitted with remote controllable devices that require protection.

Smart grid communication is often based on technologies known from the Internet and, therefore, well understood, reasonably priced, and capable of providing affordable and reliable control mechanisms to remote areas. However, they also introduce vulnerabilities and an increased attack surface. Although power grid control equipment typically has physical tampering protection and utility companies protect their assets against cyber threats, new vulnerabilities will emerge in the future.

Cyber attacks often involve attackers directly invading their target with the help of functional malware, as in the Ukraine attack, cf. [8,11]. Development of malware technology with increasing modularity opens novel opportunities and risks, in particular when such malware acts autonomously in large scale smart grid implementations. Therefore, this paper defines metrics that allows to quantify various performance aspects of malware as prerequisite for adequate countermeasures and defensive solutions against malware. These methods can complement and support basic protection measures of smart grid networks that, like most industrial networks, should be isolated from the Internet and protected by dedicated border gateways, firewalls, and intrusion detection systems (IDS) that the utility company operates. We simulate the initial stages of malware propagation in such networks with the goal of commencing cyber attacks against critical infrastructures.

The remainder of this paper is structured as follows. Section 2 defines the system model representing the operational

✉ Tanja Zseby
tanja.zseby@tuwien.ac.at

Peter Eder-Neuhauser
peter.eder-neuhauser@posteo.eu

Joachim Fabini
joachim.fabini@tuwien.ac.at

1 Communication Networks Group, Institute of Telecommunications, TU Wien, Gusshausstrasse 25/E389, Vienna 1040, Austria

environment in which our malicious software (malware) models are applied. We define the system borders, communication technologies, network topology, node types, and network traffic. Section 3 defines the attack model and its capabilities, including details on all malware types. Section 4 introduces metrics by which we categorize the malware types. Section 5 discusses the simulation results and our findings, whereas Section 6 proposes countermeasures. Section 7 concludes the paper and details on the findings.

The contributions of this paper include:

- A set of generic metrics to describe, quantify and compare the scanning, propagation capabilities and efficiency of malware from a communication network perspective.
- *Three prototypical malware categories* featuring:
  - Different *scanning behavior* that allows us to extract details for effective detection mechanisms.
  - Different *propagation behavior* that allows us to extract details for effective detection and protection mechanisms.
- A *mesh-network-based smart grid simulation environment* with a backhaul network infrastructure that is attacked using different malware categories. The results allow inference of different malware behavior and anomalies that help us define defensive solutions.
- A *specific anomaly detection* method based on flow anomaly detection that is able to detect covert malicious packets inside legitimate flows.
- Several *countermeasures* that support the defense against these malware types.

2 System model and operational environment

We use the ns3 simulation environment [28] to model three malware classes and simulate attack propagation in urban mesh-based smart grid networks.

2.1 Topology

The proposed network topology includes four wireless mesh-based sub-networks that are interconnected via a wired backhaul network infrastructure as shown in Figure 1. Each mesh network contains a number of field nodes, each representing one city building and, therefore, one remote controllable smart grid unit. This building unit includes sensors and other local devices, such as switchable loads, smart meters [34] and distributed generators. This results in 2352 apartments simulated over 196 buildings, representing a small city district. We introduced an extensive investigation on smart grid network topologies in [9] which shows that such mesh-based

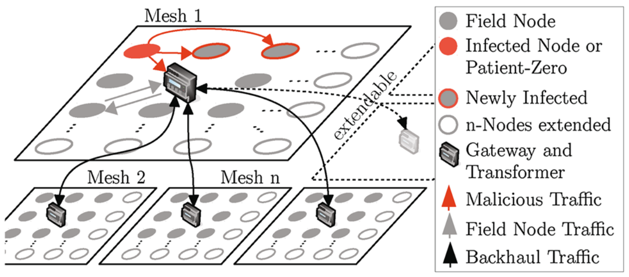


Fig. 1 Sub-network structure of the simulation environment

network topologies benefit from increased resilience against node failure compared to star-topologies. Furthermore, we discussed that smart grid networks must scale to huge numbers of nodes, likely to be implemented by the same utility company in a monoculture of devices. Therefore, we simulate a subset of the large scale smart grid.

Figure 1 illustrates several field nodes and one center-gateway node inside each one of four mesh sub-networks. The mesh sub-networks are connected by the backhaul network infrastructure via the gateway nodes. The figure illustrates each mesh sub-network in a square layout of a 49 node grid. We assume that a square layout is reasonable for a city district. The mesh networks are modeled using the Open Link State Routing (OLSR) protocol as implemented in ns3 version 3.26 [28].

The backhaul network infrastructure is modeled as optical fiber links (point-to-point links in ns3) between each gateway/transformer. We limit our network size to 49 field nodes because several sources [2,30,35] have discovered scaling limitations with OLSR-based mesh networks larger than 70 nodes unless hybrid infrastructures are used, hence the backhaul links. We assume that these limitations are taken into account when establishing urban mesh structures. We assume a distance between the field nodes of 100 m because this distance fits well within city districts between buildings. Therefore, gateways are at least 700 m apart, because they are located in the center of each mesh network.

2.2 Communication

The proposed communication model consists of field nodes (e.g., smart meters) and gateways (e.g., data concentrators). Field nodes are allowed to initiate communication with the gateway and send data to it. Additionally, field nodes are not supposed to initiate IP-layer connections with other field nodes directly. However, packets are typically processed by intermediate field nodes when routing these packets through the underlying OLSR network. This is why faulty configurations, impersonation of gateways by malicious field nodes or vulnerabilities in the OLSR implementation may enable field nodes to address and contact each other. Gateways can communicate with each other and can establish connections with field nodes in their managed subnet to, e.g., pull power grid data, or send firmware updates. We do not simulate update traffic because it does not occur in regular intervals. Moreover, some existing malware types (introduced in Section 3) are capable of impersonating update servers, in our case the gateways, to contact new victims and infect them. This behavior is illustrated in the malicious traffic in Figure 1. Since field nodes do expect commands from the gateway, but not from neighboring field nodes, they are susceptible to lateral attack vectors. The network model assumes exclusively unicast communication, excluding any broadcast or multicast communication patterns.

In order to transmit data from field nodes to the gateway each field node establishes a Transmission Control Protocol (TCP) connection to the gateway. Data is sent repeatedly in 60 s intervals. If the gateway is not in radio range of the field node, data is routed through the mesh network (OLSR routing) to reach the gateway. After successful establishment of the connection between field node and gateway, the field node begins sending without any further request. The native data traffic between field nodes and gateways is modeled at 100 kbyte every 60 s [1].

Gateways from different mesh networks are connected via the point to point backhaul network infrastructure. The gateway in the first mesh sub-network initiates communication to the other gateways. They exchange aggregated data and negotiate switching optimizations. This backhaul traffic is modeled at 100 kbyte every second [1].

We assume that the backhaul network infrastructure is not immune to the attack, meaning that gateways can be infected by the malware, too. This assumption is challenging but nonetheless realistic. Utility companies have major incentives to deploy devices that have been manufactured by the same vendor. These reasons include interoperability on one hand. On the other hand, utility companies can decrease their maintenance cost when deploying monocultures in terms of devices, decreasing effort for firmware update, device replacement, personnel training, etc. Vendors, however, have substantial incentives to reuse hardware and software to lower their development and manufacturing costs for devices. As recent publications [23,25] have shown, monocultures in terms of common vulnerabilities may derive from as little as using a common processor family. Putting all of these arguments together there is a realistic likelihood that a malware can exploit vulnerabilities both in field nodes and in gateways and full infection (lateral movement to other sub-networks) is possible. We assume a dedicated network with no other services than smart grid control traffic, hence no external traffic or office utilization.

Gateways are assumed to be located in the master-transformer station in the center of each mesh network. Gateways are responsible for data collection, remote switching, and local energy optimizations. In reality there exist several local slave transformers that are located inside the buildings. We consider them inside the field nodes and omit them from the simulation. The reason for omitting them is that the master transformer control unit, the gateway, is the only switching unit capable of disabling large areas, overruling slave transformer settings. The slave transformers are remotely controlled by it. Although these transformers cannot be ignored when simulating the electricity grid, they can be omitted in communication networks, because an infected gateway is capable of switching off entire sections, rendering the slave transformers irrelevant. Therefore, we assume that the gateway unit is capable of disabling the electricity supply for all subjacent nodes in the sub-network.

Table 1 Collection of all settings and assumptions in the system model

	Field nodes	Gateways
Number of nodes	49 per subnet	1 per subnet
Total number of nodes	192	4
Sending cycle [s]	60	1
Distance [m]	100	$>$ 700
Connection to	Local gateways	Neighbor gateways
Legitimate data [kbyte]	100	100

We use the standard OLSR model available in ns3 [28] and increase the interval of periodic OLSR control message exchanges [29] because we use a rigidly static network with no topological changes. Since these changes have no effect on the network structure they can be omitted from overall network traffic. Furthermore, Trullols et al. [38] state that OLSR generally does not converge within one control message iteration. Therefore, we allow the initial control traffic to settle until all nodes have all location information about the network before starting the actual malware simulation.

Table 1 summarizes all settings and assumptions made in the system model.

3 Attack model

In our attack model the attacker tries to infect as many nodes as possible with an automated self-propagating malware. The reason for using self-propagating malware is that, although direct infection of few major control nodes may be feasible, manual infection does not scale as well in terms of large scale smart grids. We simulate a subset of a city wide scale smart grid network, the infection goal being launching different attack types against either a utility company or selected field nodes, depending on what types of nodes can be infected. Attack examples include for instance, disconnecting parts of the electricity grid (if this functionality is implemented in the field nodes), modifying sensor readings to influence, e.g., invoices, statistics, and grid control decisions, and using compromised devices for other malicious activities, e.g., establishing a botnet, as was introduced in [10]. In the case of few infected field nodes the attacker cannot do much damage, except disconnect some field nodes or modify their readings. However, when the first gateway is infected, the attacker gains control over an entire sub-network, enabling a shutdown. Furthermore, the attacker can infiltrate aggregated data collected at the gateway, or propagate to all subjacent field nodes. Using the gateway, the malware can gain access to the backhaul network propagating across other parts of the network.

After infecting the last gateway the attacker can control all power switching equipment in the entire network, including their aggregated data that concern all sub-networks regardless if all field nodes are infected. However, if all field nodes are infected, additional attack vectors emerge that include, e.g., selective deactivation of targets, selective espionage on targets, building a botnet across a great number of nodes, and unpermitted distributed computing on a large number of infected nodes.

We define the attack models' starting point at one compromised field node as patient zero, i.e., the first infected node, in the lowest level of the hierarchy. Patient zero represents, e.g., an infected smart meter, which can initially be compromised via an unpatched vulnerability, e.g., zero day exploit, or gaining physical access to the device. From there, the malware spreads via different propagation vectors, discussed in Sections. 3.1 through 3.3, and introduced in [10]. Additionally, we assume that an exploitable zero day vulnerability exists in the entire population of hosts, as can occur in monocultures of devices, cf. [23,25], or alternatively a group of vulnerabilities that have in sum the same effect on different types of devices. The vulnerabilities allow remote code execution and administrator rights upon infection throughout the entire population of network devices. Since utility companies have strong financial incentives to implement device monocultures in terms of single-vendor deployments as detailed earlier in Section 2.2, the same or similar vulnerabilities across the network hierarchy can be expected in such a setup.

The communication model in Section 2.2 states that field nodes can initiate communication to the gateway and vice versa. Infected field nodes can therefore disguise themselves as gateways toward other field nodes, as the existing infection vector of flame [20] demonstrates, to infect victims laterally. Generally the malware tries to infect all available nodes, the goal being full infection. Since all malware types introduced in this model use different scanning techniques, we cannot make general assumptions except that losses on the communications link can lead to missed scanning opportunities. Figure 2 shows an illustration of all possible states on the communication links and nodes. Figure 2a shows malicious scanning of a non existent node. This only occurs when malware scans an address space of a sub-network regardless if real hosts exist. Figure 2b shows scanning of an existing node with packet loss, which may occur when the network link is saturated. Both cases result in a timeout. Figure 2c shows successful scanning, infection, and confirmation of the newly infected node. Figure 2d shows legitimate data transfer between a field node and a gateway.

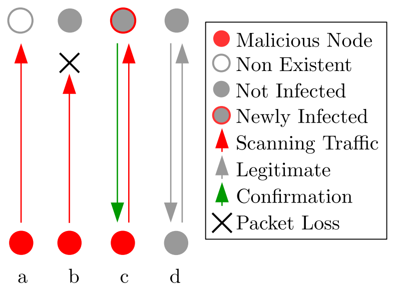


Fig. 2 Communications link and packet losses

The three investigated malware types range from simple brute force types to advanced stealthy types, as introduced in [10]. Aligning their naming with the notation used by the Centers for Disease Control and Prevention (CDC) [5] we term the three malware types pandemic malware, endemic malware, and contagion malware.

This difference in intelligence is reflected by the malware's communication patterns, as well as by the size of the payload that it must transfer over the network. This assumption is in line with the extensive review of malware evolution that has been published in [4] and bases on the observation that less complicated malware embodies fewer features requiring fewer lines of code, and needs lower development effort, which is why it can be developed by less equipped attackers. In addition, a comparison of malware that implements few features [6,27,31,33] vs. complex malware that implements advanced stealthiness and modular extensions, cf. [12,17,22,26,36] illustrates the correlation between feature set and payload size. We conducted an extensive investigation on malware capabilities in [10] that confirms these assumptions, albeit not all advanced malware have a large payload. We do, however, see a trend toward increased modularity, which brings additional capabilities to advanced malware often at the cost of a larger payload. Details can be found in Sections. 3.1 through 3.3, and in [10]. We only simulate the network side of malware propagation and scanning. Therefore, we omit those parts of network traffic used for remote control, i.e., Command & Control (C&C), to remove direct human interaction from the propagation process, making our simulated malware types less dependent. We assume that our malware types are self-propagating, thus, represent the initial propagation stages until all nodes are infected. However, remote control could be added at a later stage to start the attack phase or to extend the malware's functionality with novel or updated features.

Furthermore, we do not simulate host mechanisms, e.g., infection or exploit. However, we do consider them and other host based factors in a random delay between 0 and 0.5 s during host infection representing differences in operating systems, exploit qualities, and other random factors.

3.1 Pandemic malware

This aggressive malware type summarizes features of, e.g., Code Red 1 and 2 [6,27,31,33], Nimda [6,33], Slammer [24,31], and Conficker [7,13,32], with the goal of rapid infection of all nodes on the network. We use the naming convention of the CDC [5] to characterize this aggressive, fast, and wide reaching malware type. Pandemic malware, as introduced in [10], scans aggressively without hiding from detection mechanisms, propagates quickly to new victims, and uses a simple monomorphic self-carried payload without any detection-evasion or obfuscation mechanisms. Furthermore, malicious data transfer and scanning is done without regard to native traffic leading to saturated communication links. Therefore, timeouts, retransmissions, and postponed legitimate traffic produce many anomalies in the process. This malware type opens unsolicited TCP connections (disguised as a fake gateway) to other field nodes that should be detected easily if the network is monitored. Unsolicited TCP connections between field nodes should not occur and do not match the behavior of legitimate applications. Our pandemic malware model is capable of restricting its scanning strategy to the sub-network by collecting easily available network information from the host. Therefore, it does not scan the entire Internet Protocol version 4 (IPv4) address space, but instead checks the subnet mask to extract information about its sub-networks size, a topological scan strategy. Pandemic malware is meant to be simple, which is why it is inept at advanced espionage and commonly does not feature neither modular extensions nor large payload.

The most relevant features for the pandemic malware model can be summarized as follows:

- *Aggressive topological scanning:* Every infected node scans the entire sub-network cf. [24,31] with 100 scans per second.
- *Automatically initiates unsolicited TCP connections:* Infected nodes connect to potential new victims and transfer a self-carried payload, regardless of native network traffic.
- *Small payload size:* A simple monomorphic payload of 500 bytes represents that this malware type has no advanced on-board features and no obfuscation capability.

3.2 Endemic malware

Endemic malware describes the second malware model, behaving more stealthily than pandemic malware, as was introduced in [10]. The persistent presence that this malware exhibits in a group of hosts recommends drawing a behavioral analogy to diseases and using the CDC naming convention of endemic malware [5]. Its behaviour is loosely modeled after the behavior of, e.g., Regin [17,36], Duqu [3,19,22,37], and Flame [3,20,22]. Worth mentioning is that Stuxnet [12,22,26] shares many features with the endemic category but does *not* use hit lists, which is why it has been omitted in the exemplary malware list.

All endemic malware variants have in common increased intelligence for a number of obfuscation, espionage, and attack features in modular extensions. Summarizing the high-level description, endemic malware poses a greater challenge for defenders, although detectable in strictly controlled networks.

Their scanning strategy uses an optimized permutation-hit-list as described in [10,24,31,33]. Already scanned nodes are taken off the hit-list, not scanned again, and the list is passed on to child-malware. The child-malware begins a new random scan within a reduced search space. This strategy minimizes multiple scans of the same nodes and decreases suspicious and detectable network traffic. Additionally, this malware type can extract more information from an infected host, i.e., sub-network size and a list of existing hosts via OLSR information. We assume that endemic malware is subject to high development skills and carries advanced on-board features in its payload, including encryption among other obfuscation features, espionage features, and different attack features. We assume that it is capable of using a polymorphic payload for propagation that is more difficult to detect by signature-based defense mechanisms. However, we do not simulate a second-channel payload that represents increased modularity, instead use a self-carried payload that transfers the entire malware in one package.

The most relevant features for the endemic malware model can be summarized as follows:

- *Permutation-hit-list scanning:* Every infected node scans part of the sub-network randomly according to the hit-list with 1 scan per second cf. [24,31,33]. The hit-list is optimized by local information extracted from infected hosts.
- *Automatically initiates unsolicited TCP connections:* Infected nodes connect to discovered victims and transfer a self-carried payload.
- *Large payload size:* A complex large payload (5000 bytes) represents more capable on-board features that are transferred in a polymorphic payload. These features include hiding the malware from detection mechanisms.

3.3 Contagion malware

The third malware class we simulate is called contagion malware, as introduced by Staniford et al. [33], and discussed in detail in [10].

Contagion malware follows a passive scanning strategy and refrains from any active scanning. Legitimate TCP connections between non-infected remote hosts (applications) and infected hosts serve as both, an indication of the existence of a new victim (as a replacement of active scanning) and as communication channel for infecting the new victim, cf. [10,24,31,33]. Right before the remote application closes its TCP connection, the contagion malware sends its payload on the established communication channel and infects the victim.

So contagion malware reuses existing legitimate TCP flows of infected hosts and application-layer vulnerabilities in the target to embed and transfer its payload. Main drawback of this method is the malware's dependence on vulnerable applications to exist on host systems and target systems that actively establish TCP flows. Contagion malware must wait for host systems to communicate with each other before it can infect neighboring nodes, which can result in huge propagation delays but offers the benefit of largely reduced anomaly output.

Contagion malware propagation is highly challenging to trace by network-detection mechanisms or intrusion detection systems as this propagation method is not visible as separate TCP flows and hides within legitimate traffic. These methods include highly obfuscated network channels, e.g., appended onto legitimate traffic or, although not simulated, passive propagation via removable drives. There are few real world examples available. Therefore, we include malware types that utilize highly evolved obfuscation features and are notoriously difficult to detect. Examples include Gauss [3,21,22], Equation [18,22], and AdWind [15,16,22]. We assume that the malware uses a metamorphic payload, i.e., highly obfuscated, encrypted, and periodically reiterated on the host system making host-based detection even more challenging.

The most relevant features for the contagion malware model can be summarized as follows:

– *Passive scanning:* This method leaves no traceable scanning anomalies in the network.
– *No unsolicited TCP connections:* This malware type exploits application-layer vulnerabilities of the target to transfer its payload using legitimate TCP connections between host and target before connection close.
– *Large payload size:* A complex large payload (5000 bytes) represents more capable on-board features that are transferred in a metamorphic payload. These features include hiding the malware from detection mechanisms and obfuscating it amongst native network traffic.

Table 2 Collection of all factors discussed in the attack model

	Pandemic	Endemic	Contagion
Scanning	Topological	Hit-list	Passive
Behavior	Scan	Scan	Scan
Scan-rate [1/s]	100	1	N.A.
Propagation	Self-carried	Self-carried	Embedded
Payload [Byte]	500	5000	5000
Payload morphism	Mono-morphic	Poly-morphic	Meta-morphic

Table 2 summarizes the characteristics and parameters of our malware variants.

4 Metrics

This section describes the metrics by which we evaluate and compare the three malware models introduced in Section 3.

4.1 Node infection ratio

We define the *infection ratio* (R_{inf}), i.e., the percentage of all nodes that are infected during the simulation, as follows:

Notation:
s = Number of sub-networks
n_{inf}(i) = Number of infected nodes in sub-network (i)
n_{host}(i) = Existing number of nodes in sub-network (i)

$$R_{inf} = \frac{\sum_{i=1}^{s} n_{inf}(i)}{\sum_{i=1}^{s} n_{host}(i)} \quad (1)$$

Consequently, the *ratio of clean nodes* (R_{clean}), i.e., the percentage of nodes that have not been infected, is defined as follows:

$$R_{clean} = \frac{\sum_{i=1}^{s} n_{host}(i) - n_{inf}(i)}{\sum_{i=1}^{s} n_{host}(i)} \quad (2)$$

Generally, field nodes have one network interface. However, the gateway nodes being the connecting nodes between different networks, can have several network interfaces. Therefore, n_{host} and n_{inf} take those extra interfaces into account making it usable in other metrics that consider connections between IP addresses via those interfaces.

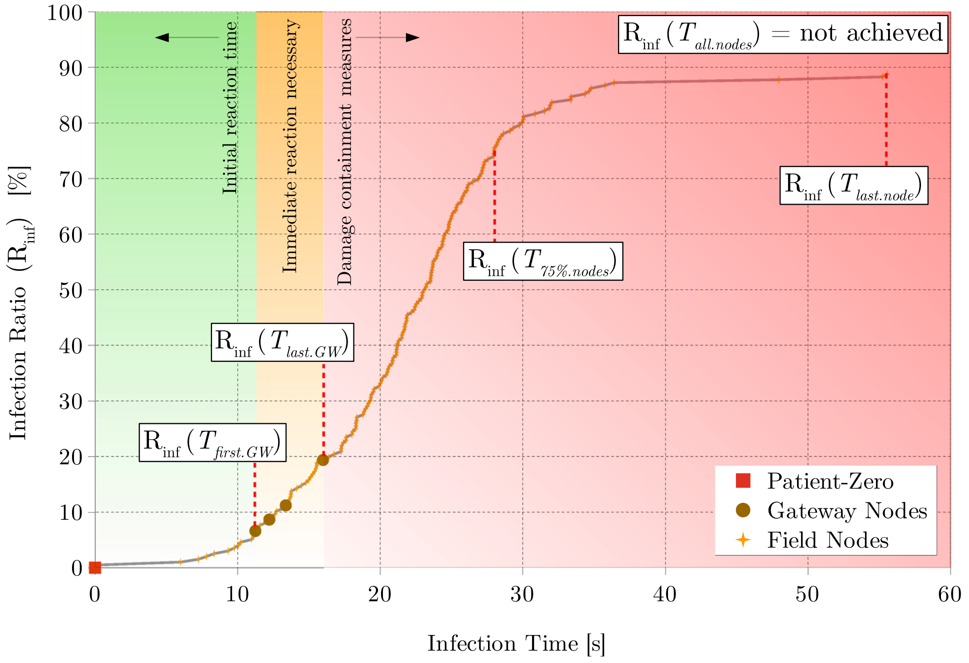


Fig. 3 Infection graph with significant infection times, an arbitrary example to illustrate the metrics

4.2 Infection times

We define five points in our simulation that characterize the instant in time when significant events occur at specific positions of the network topology. We start the infection time with patient zero, which is infected manually at the lowest level. The next significant point in time is when the first gateway is infected ($T_{first.GW}$). This is when an attacker gains the capability to either spy on the aggregated data of the gateway or disable communications and electricity supply to the entire sub-network and subjacent field nodes, regardless if all the field nodes of this particular subnetwork are infected. The second point denotes infection of the last gateway ($T_{last.GW}$), this is also the last instance at which trustworthy sensor readings and reports can be expected from the nodes on the network. At this point in time the attacker can spy on all aggregated data or disable communications and electricity supply to all gateways including their subjacent field nodes, regardless if all field nodes are infected. Such an attack—even if not fully automated—can destabilize the power grid, as witnessed in the Ukraine incident [8,11]. We then define the point when 75% of all nodes are infected ($T_{75\%.nodes}$). At 75% infection ratio the attacker already has a significant amount of nodes under control. Also the chance that a specific target of interest is among the infected nodes is high. This opens additional attack vectors such as selective deactivation of field nodes, selective espionage, building a botnet, or unpermitted distributed computing on a large number of infected nodes.

Next, we introduce the point that marks the time when the last node has been infected ($T_{last.node}$), i.e. no more nodes can be infected and the malware does not propagate any further. Full infection can not be achieved in all cases, potential reasons include packet loss or that some nodes are not scanned by the malware algorithm, e.g., if errors exist in the hit-list. This is why we define $T_{last.node}$ as the (always measurable) point in time when the last node is effectively infected and ($T_{all.nodes}$) as the (potentially hypothetical) instant when *a*ll nodes are infected. Whenever the malware fails to infect all nodes, $T_{all.nodes}$ will be set to infinite. With respect to the assigned value, $T_{last.node}$ that marks the end of the simulation run will be always less or equal to $T_{all.nodes}$, equality being recorded if and only if the malware succeeded in infecting all existing nodes.

Summarizing our simulation is characterized by the following time instances:

$T_{first.GW}$ = Time until first gateway is infected
$T_{last.GW}$ = Time until last gateway is infected
$T_{75\%.nodes}$ = Time until 75% of all nodes are infected
$T_{last.node}$ = Time until the last field node is infected.
$T_{last.node} \leq T_{all.nodes}$
$T_{all.nodes}$ = Time until all nodes on the network are infected. May be ∞

In Figure 3 we show an arbitrary example infection graph with the infection ratio R_{inf} as function of the infection time, illustrating the time instances. Colored areas in the graph illustrate that countermeasures must react within a short time window, e.g., until the first gateway is infected, in order to prevent the malware to propagate into other sub-networks.

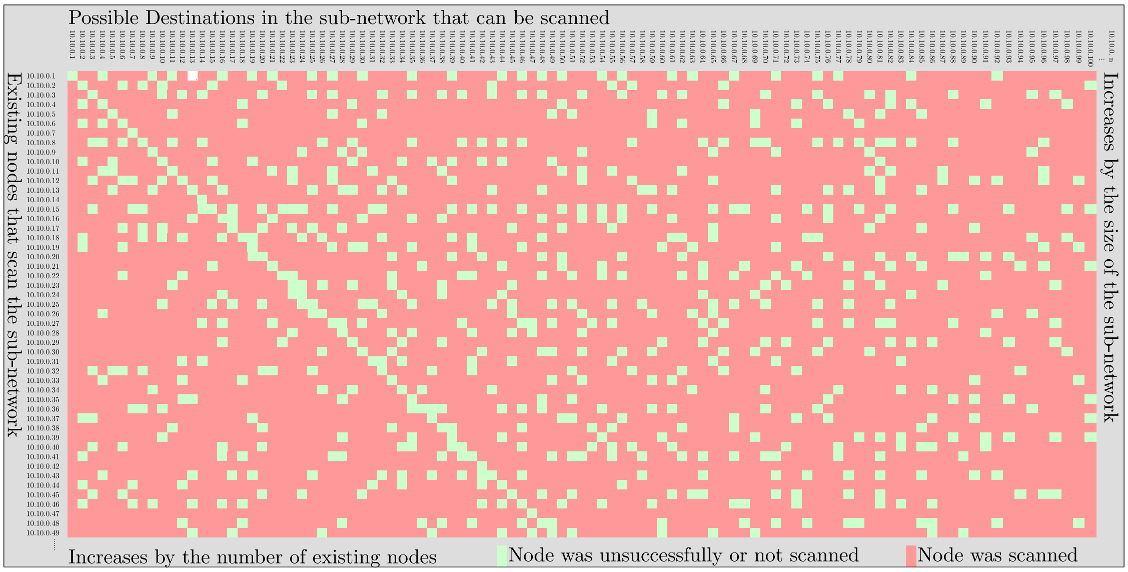


Fig. 4 Scan ratio of pandemic malware

4.3 Scanning stealthiness

We characterize the scanning stealthiness of a malware by its scanning behavior, i.e., how much "noise" a malware generates when scanning the network. We define two indicators to show the amount of scanning traffic needed for the infection process. First, the *scan ratio* (R_{scn}) represents the ratio of (successfully and unsuccessfully) scanned addresses to all theoretically possible scans.

Notation:
n_{host}(i) = Existing number of nodes in sub-network (i)
n_{addr}(i) = Number of theoretically available addresses per sub-network (i)
n_{scn}(i) = Number of all scans for sub-network (i)

$$R_{scn} = \frac{\sum_{i=1}^{s} n_{scn}(i)}{\sum_{i=1}^{s} n_{host}(i) * (n_{addr}(i) - 1)} \quad (3)$$

The denominator in formula 3 defines the scanning space, i.e. the number of theoretically possible scans, when each host scans other hosts at most once but not itself:

$$\sum_{i=1}^{s} n_{host}(i) * (n_{addr}(i) - 1) \quad (4)$$

Therefore, each address may be scanned multiple times by different nodes if a malware is not sophisticated to coordinate the scanning behavior. Different scanning strategies discussed in Sections. 3.1 through 3.3 dictate whether a source node scans the entire address space, just a reduced address space, or does not scan at all.

Figure 4 shows an example scanning space for one subnet. Red dots mean a scan has been performed from a specific host (y-axis) to an address of the address space (x-axis). Green dots mean the host did not scan the particular address. Worth noting is the diagonal line of green dots that identifies nodes never scanning themselves. For our model we also assume that one node scans one address only once and we do not have any retransmissions (re-scanning by the same node if a scan packet gets lost). So one hosts scans each address of the address space at most once.

Using the scanning space defined in Formula 4 the *scan ratio* in formula 3 is defined as the number of scans that were actually performed by the malware divided by the theoretically possible scans (scanning space). A higher value represents more scanning "noise", i.e., more traffic that may be detected and therefore a lower scanning stealthiness. The ratio of *unscanned addresses* (R_{uscn}) from the scanning space is illustrated by the green dots in Figure 4 and defined as follows:

$$R_{uscn} = 1 - R_{scn} \quad (5)$$

We represent scanning "noise" with the indicators R_{scn} and R_{uscn} rather than illustrations. Therefore, Figure 4 is just shown as an example and will not be repeated for all malware simulations.

4.4 Scanning efficiency

Furthermore, we define the *efficiency of scanning* (E_{scn}) that represents how efficient a malware type is discovering new targets among available addresses. A higher value represents a less “noisy” scanning strategy. E_{scn} defines the infection ratio of all infected nodes in the network to the number of scans. It is defined as follows:

$$E_{scn} = \frac{\left(\sum_{i=1}^{s} n_{inf}(i)\right) - 1}{\sum_{i=1}^{s} n_{scn}(i)} \tag{6}$$

where

$$\left(\sum_{i=1}^{s} n_{inf}(i)\right) - 1 \tag{7}$$

represents all nodes that are infected across all sub-networks *excluding patient-zero* which is infected manually. Because patient zero is never scanned from the network side we substract it from the overall number of infected hosts. Therefore, the maximum efficiency is 100% when each scan results in one infection.

E_{scn} expresses a very strict definition of efficiency. A 100% efficiency can only be achieved in two cases:

- One source node per sub-network scans all existing nodes and every scan is a success. No packet loss occurs and none of the other hosts participate in scanning. In this case all the scanning effort needs to be taken over by one node, decreasing the scanning speed. In addition, the scanning source is easily detected if the network is observed.
- The scanning is highly coordinated. One possibility is to use sequential scanning where each node only scans and infects one node, then stops, and the following node continues. Sequential scanning is slow and fails if a scanning packet gets lost. An alternative is to use some control traffic (C&C) to coordinate the scanning. But such control traffic requires additional effort and also reduces the stealthiness.

In the optimal case the number of scans is equal to the number of existing nodes excluding patient zero, i.e., $E_{scn} = 100\%$. However, if more than one node scans the sub-network or nodes scan unassigned addresses, the efficiency drops considerably. In reality, attackers aim to optimize the scanning strategy depending on the sophistication level of their malware. Decreasing scanning output to reach 100% efficiency would require perfect coordination, thus shifting this effort to control traffic. Since we assume self-propagating malware in this work, we do not simulate malware with sophisticated control structures.

4.5 Propagation stealthiness

In addition to scanning, the malware also need to propagate itself, i.e. sending the actual payload to the new victims. This is another activity that can be detected in the network and reduces stealthiness. We define the propagation behavior of malware that utilizes *unsolicited traffic* (U_{tr}) as another measure for its stealthiness. U_{tr} describes what percentage of traffic (in bytes) in a certain time frame have suspicious origin, thus are not invoked by legitimate smart grid applications. It is defined as follows:

Notation:
T_{active} = Time of malware activity, used to normalize infectious traffic with overall traffic such that: $B_{mal} = 0$ for all t > T_{active}
$B_{mal}(T_{active}, i)$ = Bytes associated with unsolicited traffic in sub-network i during interval (0, T_{active})
$B_{total}(T_{active}, i)$ = Bytes of total traffic in sub-network i during interval (0, T_{active})

$$U_{tr} = \frac{\sum_{i=1}^{s} B_{mal}(T_{active}, i)}{\sum_{i=1}^{s} B_{total}(T_{active}, i)} \tag{8}$$

The time of malware activity (T_{active}) needs to be defined, because the activity time differs for the different malware types. In order to get comparable results we always compare the unsolicited traffic during the activity time interval with the total traffic in the activity time interval. Additionally, we define another metric representing the stealthiness of malware based on the expected origin and target of TCP flows, thus ignoring all other protocols. We name it *unsolicited TCP flow* (U_{TCP}). It is defined by the number of unsolicited TCP flows in comparison to all TCP flows within the time interval (0, T_{active}). Although the metric does not convey information about the amount of data transported, it does represent the amount of connections established.

Notation:
$F_{mal}(T_{active}, i)$ = Number of TCP flows with unsolicited traffic in sub-network i during interval (0, T_{active})
$F_{total}(T_{active}, i)$ = Number of TCP flows in effective overall traffic in sub-network i during interval (0, T_{active})

$$U_{TCP} = \frac{\sum_{i=1}^{s} F_{mal}(T_{active}, i)}{\sum_{i=1}^{s} F_{total}(T_{active}, i)} \tag{9}$$

Both propagation detection metrics assume that the network operator has good monitoring and knowledge about the expected network traffic patterns.

4.6 Summary of all notations

Table 3 summarizes all notations introduced in the metrics of this section.

Table 3 Summary of all Notations in order of appearance

Notation	Explanation
s	Number of sub-networks
n_{inf}(i)	Number of infected nodes in sub-network (i)
n_{host}(i)	Existing number of nodes in sub-network (i)
$T_{first.GW}$	Time until first gateway is infected
$T_{last.GW}$	Time until last gateway is infected
$T_{75\%.nodes}$	Time until 75% of all nodes are infected
$T_{last.node}$	Time until the last field node is infected. $T_{last.node} \leq T_{all.nodes}$
$T_{all.nodes}$	Time until all nodes on the network are infected. May be ∞
n_{host}(i)	Existing number of nodes per sub-network (i)
n_{addr}(i)	Number of theoretically available address per sub-network (i)
n_{scn}(i)	Number of all scans per sub-networks (i), cf. Figure 4
T_{active}	Time of malware activity, to normalize infectious traffic to overall traffic
$B_{mal}(T_{active}, i)$	Bytes associated with unsolicited (malicious) traffic in subnet i during interval (0, T_{active})
$B_{total}(T_{active}, i)$	Bytes of effective overall traffic (legitimate and malicious) in subnet i during (0, T_{active})
$F_{mal}(T_{active}, i)$	Number of TCP flows associated with unsolicited traffic in subnet i during interval (0, T_{active})
$F_{total}(T_{active}, i)$	Number of TCP flows in effective overall traffic in subnet i during interval (0, T_{active})

5 Results

In this section we discuss the simulation results for all malware types. The subsections include infection graphs and pcap-generated illustrations of the network traffic. In the interest of conserving space we only show the pcap patterns of the first sub-network representative for the entire network. Therefore, points in the pcap illustrations, i.e., from one network, do not correspond to all points in the infection graphs, i.e., from all networks.

5.1 Pandemic malware

This subsection discusses the simulation results for pandemic malware. Figure 5 illustrates an infection graph including patient-zero, the gateway nodes, the most significant points in time, and the field nodes. Furthermore, we mark nodes from the first sub-network that are infected by patient-zero different from the remaining field nodes. This allows an estimate of the time it takes to fully infect one sub-network. From the infection graph we can see that the malware capabilities, as discussed in Section 3.1, result in very fast infection of most field nodes, incurring a long-tail for the infection of the last few nodes. Although full infection takes longer than 60 s, 75% of all nodes are infected within the first 8 s. Additionally, all gateways are infected within less than 4 s, giving the attacker full control over the network. However, this malware type produces a large number of anomalies that could be easily detected, making it an unlikely candidate for stealthy targeted attacks.

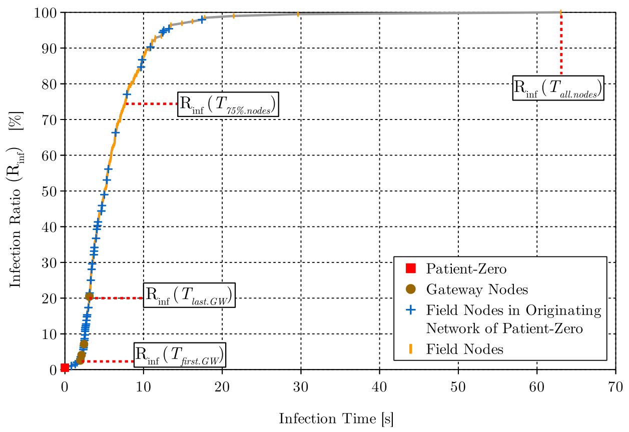


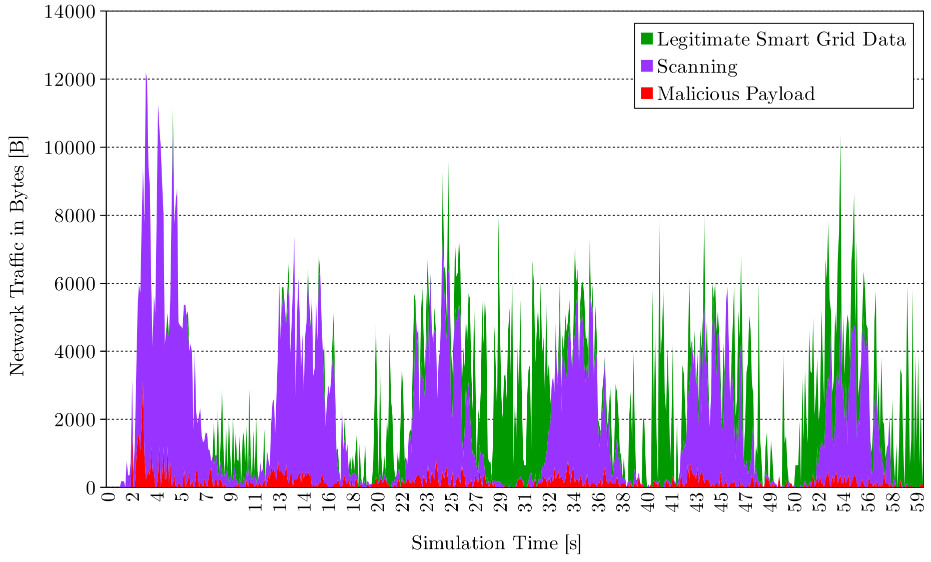
Fig. 5 Pandemic malware infection graph

Figure 6 illustrates a pcap recording that shows the high amount of scanning traffic compared to legitimate traffic. The payload transfer (propagation) of this malware produces little traffic because the payload has a small size. However, the malwares' only goal is fast infection and this is achieved with outstanding performance.

Results of the pandemic malware simulation can be summarized as follows:

- *Aggressive scanning:* High scanning ratio (R_{scn} = 94.42%) leads to detectable anomalies, thus few nodes are not scanned (R_{uscn} = 5.58%). The scanning efficiency (E_{scn} = 1.66%) is very low because every infected node scans the entire sub-network. Figure 6 shows that scanning traffic is dominant over any other network traffic, consuming the bandwidth and leading to postponed legitimate connections and highly visible anomalies.
- *Unsolicited connections:* Pandemic malware opens outbound TCP connections regardless of native network traffic, i.e., the percentage of malicious bytes inside overall traffic, U_{tr} = 2.54%. However, the percentage of maliciously established TCP connections compared to overall TCP is U_{TCP} = 60.46% for the timeframe $0 - T_{active}$. Unsolicited connections that do not match patterns expected from legitimate applications are an indication that illegitimate services are using the data link.
- R_{inf} = 100%; All nodes in this network have been infected.

Fig. 6 Pandemic malware pcap sample of the home-network



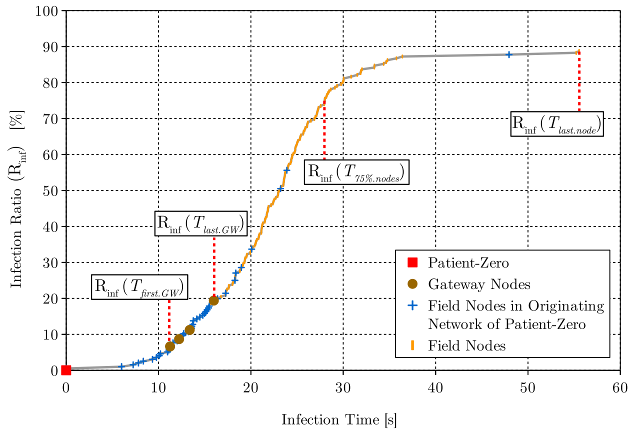


Fig. 7 Endemic malware infection graph

- $R_{clean} = 0\%$; No nodes have slipped through the infection process.
- *Infection Times:*
  - $T_{first.GW} = 1.97$ s; Entire first sub-network can be controlled.
  - $T_{last.GW} = 3.12$ s; All sub-networks can be controlled.
  - $T_{75\%.nodes} = 7.69$ sec; 75% of all field nodes can be controlled selectively.
  - $T_{last.node} = T_{all.nodes}$
  - $T_{all.nodes} = 63.03$ s; 100% of all field nodes can be controlled selectively.

5.2 Endemic malware

This subsection discusses the simulation results for endemic malware. Figure 7 presents the infection graph, which illustrates that endemic malware is generally slower than pandemic malware. However, it is still fast enough to attack all gateways within the first 20 s and 75% of all nodes within 30 s, making it a highly capable threat for network operators. Full infection is not achieved in this case. The reason is that packet loss in the scanning cycle still occurs as indicated in Figures 2 and 4. However, endemic malware follows a much less aggressive scanning strategy by passing a hit-list of scanned nodes to its children. Therefore, it is possible that nodes are overlooked and not scanned. Furthermore, the increased on-board intelligence allows the malware to reduce the search space significantly by only scanning nodes the infected victim has knowledge of. However, the hit-list is not reduced optimally and quick enough in the early stages of infection, leading to several rescans of existing nodes. However, $E_{scn} = 10.91\%$ is significantly higher than with pandemic malware.

Figure 8 illustrates a pcap recording that shows little scanning traffic compared to the payload traffic. The increased payload size leads to higher traffic producing enough anomalies for detection if the network is properly monitored. These anomalies can be detected as unsolicited connections. Furthermore, the scanning traffic, although inconspicuous, is not invisible and could also be detected. It does however obfuscate much better within native traffic compared to the aggressive scanning strategy of pandemic malware.

Results of the endemic malware simulation can be summarized as follows:

- *Permutation-hit-list scanning:* Endemic malware operates on an inconspicuous scanning strategy. This low scanning output $R_{scn} = 16.51\%$ leads to fewer detectable anomalies. $R_{uscn} = 83.49\%$. Additionally the hit-list is optimized, but still, rescanning occurs leading to a scan

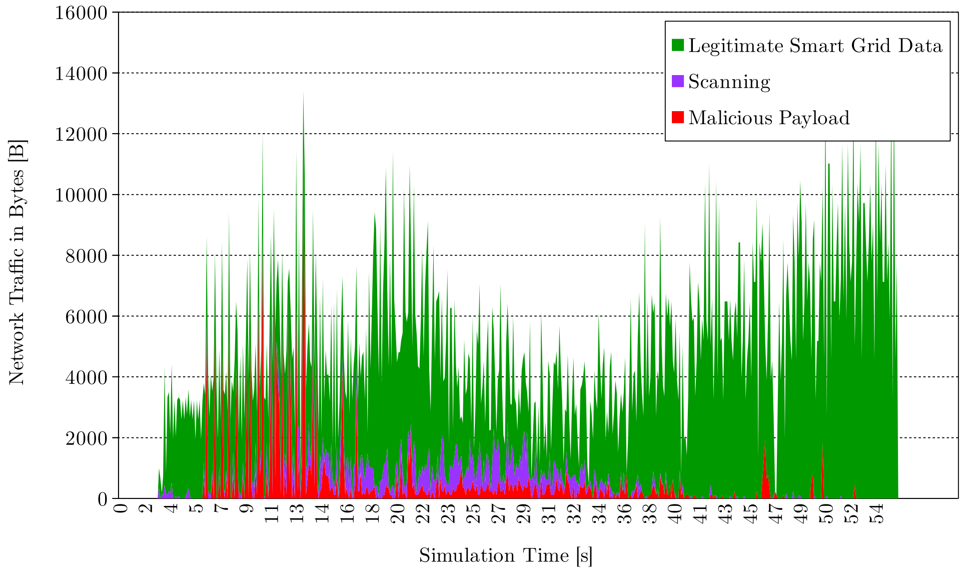


Fig. 8 Endemic malware pcap sample of the home-network

efficiency of $E_{scn} = 10.91\%$ which is much higher compared to pandemic malware. Figure 8 shows that scanning traffic is very low compared to payload and legitimate traffic.

- *Unsolicited connections:* The malware opens outbound TCP connections to transfer the payload. The payload being large (5000 bytes) compared to pandemic malware (500 bytes) produces higher peaks that could be detected, i.e., $U_{tr} = 6.49\%$. The percentage of malicious TCP connections compared to overall TCP is $U_{TCP} = 64.70\%$ for the timeframe 0 - T_{active}. These unsolicited connections do not match expected patterns from legitimate smart grid applications, thus are a sign that illegitimate services are using the data link.
- $R_{inf} = 89.11\%$ of all nodes have been infected.
- $R_{clean} = 10.89\%$ of all nodes have been missed by the infection process.
- *Infection Times:*
  - $T_{first.GW} = 11.25$ s; Entire first sub-network can be controlled.
  - $T_{last.GW} = 15.99$ s; All sub-networks can be controlled.
  - $T_{75\%.nodes} = 28.00$ s; 75% of all field nodes can be controlled selectively.
  - $T_{last.node} = 55.54$ s; 89% of all field nodes can be controlled selectively.
  - $T_{all.nodes} = \infty$

5.3 Contagion malware

This subsection discusses the simulation results for the contagion malware. Figure 9 illustrates the infection graph that shows significantly slower propagation through the network. Since this malware type does not actively scan the network, E_{scn} is not evaluated at 100% but instead "not applicable" (n.a.). However, the scanning output (R_{scn}) is still counted as 0%, thus $R_{uscn} = 100\%$. Furthermore, it is strange to find long delays between the infection of, e.g., patient-zero and the first target node. The reason for such delays can be explained by the propagation strategy that constitutes hiding the payload in legitimate native connections. Since legitimate connections are only established from field nodes to and between gateway nodes, the first victim is always the home gateway. The malware waits until the field node establishes the connection to the gateway and then piggybacks payload onto the existing TCP connection after legitimate data transfer has finished.

From there, the malware spreads quickly to other gateways and field nodes. Furthermore, the propagation strategy requires that the malicious payload is appended at the end of legitimate connections, giving priority to legitimate data, completely removing any anomaly output that is related to their postponement. Data is not injected via the network side but on the application layer, thus the TCP packet sequences do not show any anomalies. This propagation strategy favors native traffic, dropping malicious payloads that produce delays on the link. Therefore, some infections do not finish the infection cycle and the link is closed to avoid anomaly output. Parent-malware generally do not attempt to infect the same node again in favor of reduced anomaly output. Figure 9 shows that the gateway nodes are infected successively within a very short time, some even overlapping in the illustration, leaving little reaction time for automated defenses.

Figure 10 shows the malicious payload next to legitimate data. However, the payload is actually embedded in the legit-

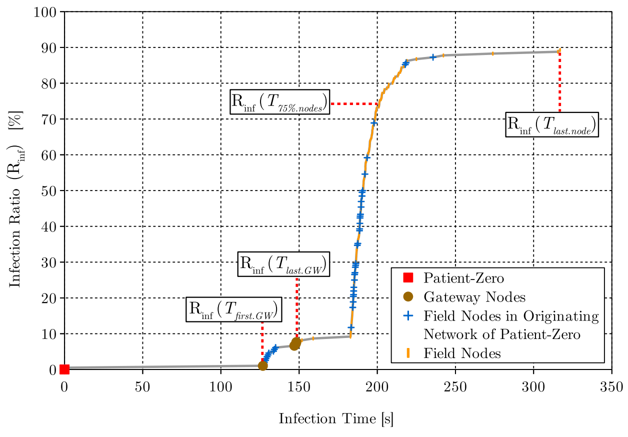


Fig. 9 Contagion malware infection graph

imate traffic. Furthermore, there is no scanning output and no dedicated TCP flow available for detection purposes.

Results of the contagion malware simulation can be summarized as follows:

- *Passive scanning:* This malware type does not scan the network, thus the scanning output $R_{scn} = 0\%$, not scanned nodes $R_{uscn} = 100\%$, and E_{scn} = not applicable. No anomalies can be detected from scanning.
- *No unsolicited connections:* This malware type appends on legitimate connections and no unsolicited connections can be detected. Therefore, U_{tr} and U_{TCP} are both 0%.
- $R_{inf} = 89.60\%$ of all nodes have been infected.
- $R_{clean} = 10.40\%$ of all nodes have not been infected.
- *Infection Times:*
  - $T_{first.GW} = 126.83$ s; Entire first sub-network can be controlled.
  - $T_{last.GW} = 148.52$ s; All sub-networks can be controlled.
  - $T_{75\%.nodes} = 201.50$ s; 75% of all nodes can be controlled selectively.
  - $T_{last.node} = 316.97$ s; 89% of all field nodes can be controlled selectively.
  - $T_{all.nodes} = \infty$
- *Waiting periods:* The infection graph, cf. Figure 9, shows long waiting periods between infections. They originate from the propagation strategy that appends the payload at the end of legitimate TCP connections to decrease anomaly output. Therefore, all legitimate data must be transferred first by the smart grid application to avoid postponed legitimate traffic, then the malware hijacks the TCP flow from the local smart grid application and appends its payload before the connection is closed. This may lead to delays caused by the regular reporting cycle of field nodes, cf. Table 1.

6 Countermeasures

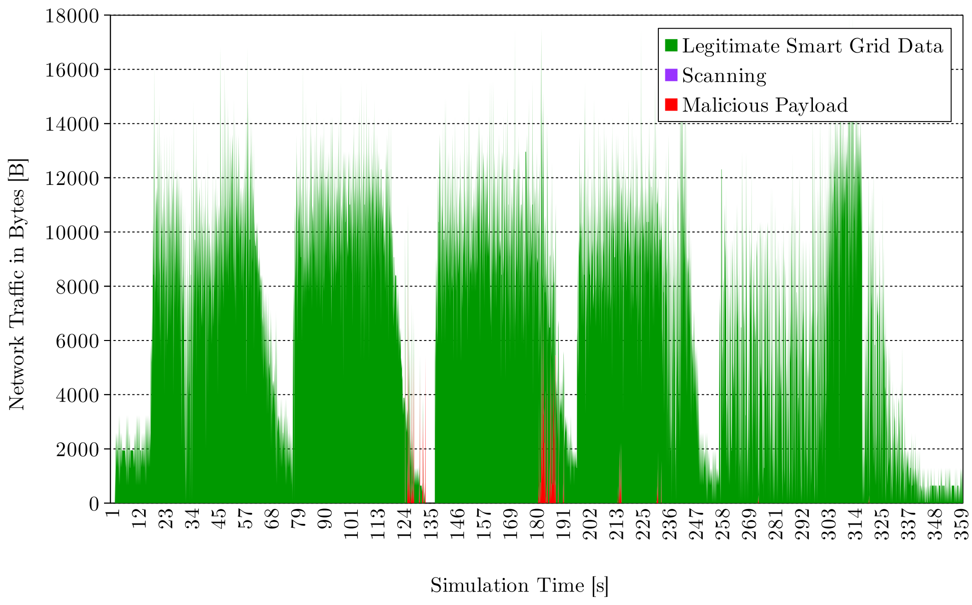
This section discusses several effective countermeasures according to the capabilities of each malware type.

6.1 Pandemic malware countermeasures

Pandemic malware should be easily defeated by basic security measures such as:

- Restricted Virtual Local Area Networks (VLAN).
- Perimeter firewalls between network segments.

Fig. 10 Contagion malware pcap sample of the home-network



– Data origin authentication using asymmetric cryptography (digital signatures).

Although the BSI protection profile [14] does not allow mesh networks, it can provide useful insight into proper authentication and segmentation mechanisms. Furthermore, slowing and blocking unsolicited connections should also suffice to effectively halt malware propagation. While alarm messages to the gateways or increased awareness on the expected communication patterns should provide an early warning, such measures should be highly effective against pandemic malware.

6.2 Endemic malware countermeasures

Endemic malware should be discovered by properly configured detection methods. Since infected nodes open unsolicited TCP connections feasible methods include:

– All defensive measures mentioned for pandemic malware.
– Intrusion detection systems that can detect unsolicited traffic and scanning behavior.

Because less scanning anomalies are available, detection must be tuned for the specific propagation patterns. Furthermore, the active services available on such critical networks should be tightly controlled in terms of expected traffic patterns. Otherwise malicious traffic may easily be missed.

6.3 Contagion malware countermeasures

Contagion malware does not produce any anomalies found in the other models that could be detected. Instead it hides in legitimate connections. Those connections could be slowed or blocked losing legitimate data in the process. Effective countermeasures include:

– All defensive measures mentioned above.
– Firewalls or intrusion detection systems checking every connection for its legitimate expected behavior.

We urge utility companies to implement extensive defense measures in all perimeter points of their segmented networks. Furthermore, services should be kept up-to-date and all networks should only provide those services required for its operation. Networks of different function should not be connected unless absolutely necessary.

6.4 Anomaly detection specific to contagion malware

We were unable to detect any network anomalies in the contagion malware simulations with the methods used for pandemic and endemic malware. Therefore, we introduce a specific detection method for contagion malware, namely *Anomalous Flow Detection* (A_{flow}) that detects patterns diverging from the expected behavior in terms of network traffic. *Anomalous Flow Detection* depends on the analysis of both, flow direction and payload size within existing TCP flows to detect the payload that contagion malware injects at the end of an active legitimate communication flow. It is worth noting that *Anomalous Flow Detection* is substantially more demanding in terms of resources than detection mechanisms for pandemic and endemic malware. Whereas the two latter malware types open unsolicited connections that can be detected with low effort, *Anomalous Flow Detection* depends on additional packet-level captures and analyses of potentially end-to-end encrypted traffic. However, the contagion malware uses metamorphic (encrypted) payload that can not be detected using pattern matching or other deep packet inspection methods, anyhow. This is why identical algorithms are applicable for the detection of anomalies in both, encrypted and non-encrypted traffic.

Figure 11 illustrates one exemplary TCP flow from a field node to the gateway, which, for the most part, does not appear suspicious. However, at the end a large payload is injected, which is visible as an anomaly. The malicious payload is colored in red. 15.7% of all connections in this simulation exhibit such a specific anomaly. The detected peaks of malicious packets at the end of legitimate data flows push data to the victim node, in our case in reverse direction of the flow. Figures 11, and 12 elaborate on the malicious traffic

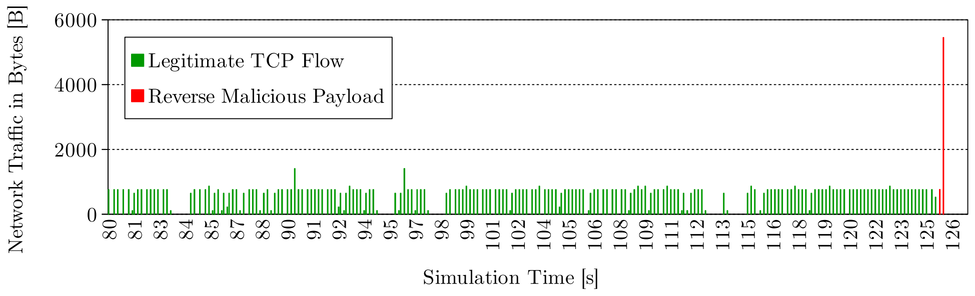


Fig. 11 Single illustration of covert malicious payload anomaly

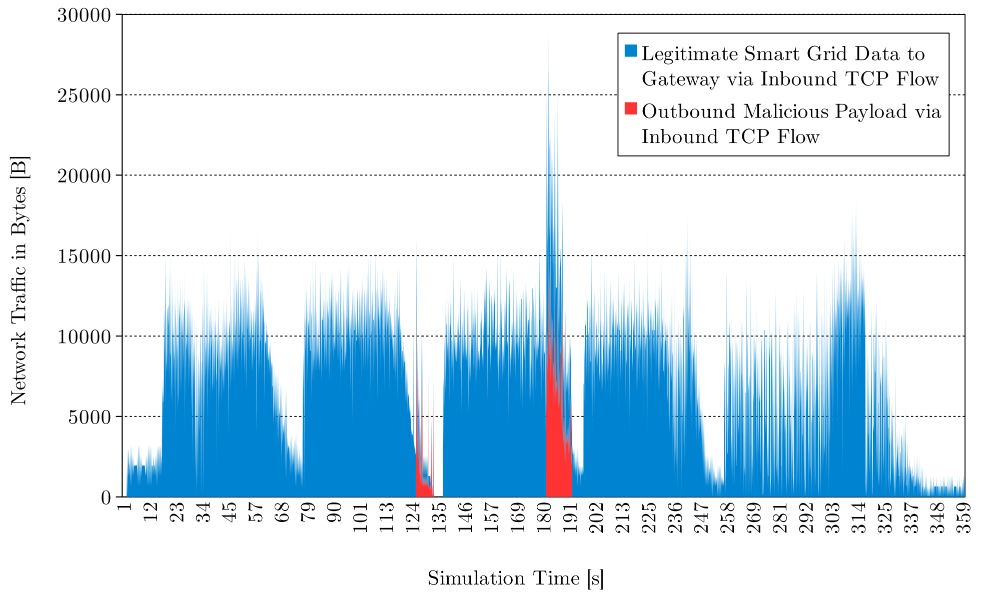


Fig. 12 Illustration of anomaly detection for contagion malware

vs. legitimate traffic, although these packets are obfuscated inside legitimate TCP flows.

Notation:
$F_{covert}(T_{active}, i)$ = Number of flows in sub-network i containing malicious data during time interval [0, T_{active}]

$$A_{flow} = \frac{\sum_{i=1}^{s} F_{covert}(T_{active}, i)}{\sum_{i=1}^{s} F_{total}(T_{active}, i)} \tag{10}$$

Figure 12 illustrates all inbound and outbound gateway traffic including the TCP overhead, legitimate inbound traffic to the gateway, and malicious outbound traffic.

6.5 Results summary

This section summarizes the simulation results of all malware types in Table 4. We highlight that pandemic malware and endemic malware could be detected with straight-forward detection methods that identify scanning traffic or unsolicited connections. However, contagion malware could not be detected with these methods. Our proposed detection method investigates the legitimacy of packets inside TCP flows. Since contagion malware does not inject packets from the network side but rather from the application layer, the packet sequence does not give any indication of malicious content. However, large packets are sent in reverse direction through the same TCP flow. This can be detected and thus provides an effective measure to discover contagion malware at the cost of additional computational effort.

Table 4 Results summary

Metric	Pandemic	Endemic	Contagion
R_{inf} [%]	100.00	89.11	89.60
R_{clean} [%]	0.00	10.89	10.40
$T_{first.GW}$ [s]	1.97	11.25	126.83
$T_{last.GW}$ [s]	3.12	15.99	148.52
$T_{75\%.nodes}$ [s]	7.69	28.00	201.50
$T_{last.node}$ [s]	$T_{all.nodes}$	55.54	316.97
$T_{all.nodes}$ [s]	63.03	∞	∞
R_{scn} [%]	94.42	16.51	0.00
R_{uscn} [%]	5.58	83.49	100.00
E_{scn} [%]	1.66	10.91	N.A.
U_{tr} [%]	2.54	6.49	0
U_{TCP} [%]	60.46	64.70	0
A_{flow} [%]	N.A.	N.A.	15.70

7 Conclusion

We introduced three malware types that attack a mesh-based smart grid environment simulated in ns3. These malware types include aggressive fast spreading malware that is easily detectable, more stealthy malware that sacrifices speed for stealthiness, and highly obfuscated malware that is not detectable by conventional means. We elaborate on the results and the differences between the malware types.

First, *pandemic* malware is very fast in infecting all nodes in the network. With its aggressive scanning strategy pandemic malware is able to scan 94% of the available addresses and enables infection times of under 8 s for 75% of all hosts. However, the scanning efficiency is measured as low as 1.66%, producing significant "noise" usable by network detection mechanisms. This behavior delays legitimate traffic which is also noticeable on the gateway. Although the small payload is less easily detected, the propagation method does require new TCP connections that appear as unsolicited flows $U_{TCP} = 60.46\%$ and transfer a low byte count $U_{tr} = 2.4\%$. Properly configured detection mechanisms should be able to discover such behavior. This malware type should be easily defeated by basic security measures, cf. Section 6.

Endemic malware uses a permutation-hit-list in its scanning strategy that requires more intelligence on the malwares' side. Its scanning strategy is capable of minimizing the search space, gathering information from local hosts, and splitting the hit-list between parent and child-malware. This leads to a much less conspicuous scanning behavior, reduced scanning $R_{scn} = 16.51\%$ and an increased scanning efficiency of $E_{scn} = 10.91\%$. However, it is still detectable to well configured defense mechanisms. Additionally, the propagation mechanism of endemic malware is more easily detected ($U_{TCP} = 64.70\%$ and $U_{tr} = 6.49\%$). A large payload, representing higher intelligence and more sophisticated onboard features, needs to be transmitted and is visible as unsolicited connections among the overall network traffic. Properly configured detection mechanisms should be able to discover these connection attempts. Endemic malware is fast enough to infect 75% of all nodes on a network in under 30 s.

Contagion malware generally poses a difficult challenge and requires strong detection mechanisms around perimeter points. This malware type does not scan the network actively but rather hijacks existing TCP connections on the application layer of infected hosts. The payload is injected into legitimate data flows and cannot be detected by anomalies such as network scanning or unsolicited connections. Additionally, the malware first targets the home-gateway of patient-zero being the only legitimate communication partner. Once the gateway is infected, propagation to other field nodes and through the backhaul network is possible.

This malware type takes a long time infecting the network ($T_{75\%.nodes} = 209.79$ s) yet moves highly stealthily. This allows the attacker to prepare selective attacks against field nodes and gateways. The only feasible detection method is to correlate TCP flows and check for reverse packets of a suspicious size that mismatches expected traffic patterns. Such anomalies could be detected within $A_{flow} = 15.7\%$ of all legitimate connections. Additionally, those connections could be slowed or blocked losing legitimate data in the process.

Summarizing, the set of generic metrics that we proposed characterize the propagation, verbosity and efficiency of malware types based on their network communication patterns. Results of the mesh-network-based smart grid simulation attacked by three malware types confirm the applicability of the proposed metrics and provide detail on scanning and propagation behavior of three malware types. The simulation results emphasize that particular attention must be paid to well-implemented security features on the gateways that act as perimeter security points between networks. Strong segmentation, a progressive update policy and fast reaction time are some of the main components that support utilities in defending against large-scale malware attacks.

Acknowledgements The authors acknowledge the TU Wien University Library for financial support through its Open Access Funding Programme. This work was supported by the research project SIMULTAN funded by the Austrian Research Funding Association (FFG).

References

1. Arbiter Systems, O.: Model 1133a Power Sentinel—GPS-Synchronized Power Quality/Revenue Standard. Product Manual. Arbiter Systems Inc, Paso Robles (2016)
2. Audeh, M.: Metropolitan-scale Wi-Fi mesh networks. Computer **37**(12), 119–121 (2004)
3. Bencsáth, B., Pék, G., Buttyán, L., Félegyházi, M.: The cousins of stuxnet: duqu, flame, and gauss. Future Internet **4**(4), 971–1003 (2012)
4. Calleja, A., Tapiador, J.E., Caballero, J.: A look into 30 years of malware development from a software metrics perspective. In: Research in Attacks, Intrusions, and Defenses—19th International Symposium, RAID 2016, Paris, France, September 19–21, 2016, Proceedings. pp. 325–345 (2016). https://doi.org/10.1007/978-3-319-45719-2_15
5. Center for Disease Control and Prevention: Principles of Epidemiology (2016). http://www.cdc.gov/ophss/csels/dsepd/SS1978/Lesson1/Section11.html
6. Chavan, M.K., Madane, P.V.: Modelling and detection of camouflaging worms—a survey. Int. J. Emerg. Technol. Adv. Eng. **2**(10), 564–569 (2012)
7. Dainotti, A., King, A., Claffy, K., Papale, F., Pescape, A.: Analysis of a "/0" stealth scan from a botnet. IEEE/ACM Trans. Netw. **23**(2), 341–354 (2015)
8. Dragos Inc.: CRASHOVERRIDE Analyzing the Threat to Electric Grid Operations. Tech. rep. (2017)
9. Eder-Neuhauser, P., Zseby, T., Fabini, J.: Resilience and security: a qualitative survey of urban smart grid architectures. IEEE Access **4**, 839–848 (2016)
10. Eder-Neuhauser, P., Zseby, T., Fabini, J., Vormayr, G.: Cyber attack models for smart grid environments. Sustain. Energy Grids Netw. **12**, 10–29 (2017)
11. F-Secure Labs: Blackenergy & Quedagh—The Convergence of Crimeware and APT Attacks. White Paper No. 1030745, F-Secure Labs, Helsinki (2014)
12. Falliere, N., Murchu, L.O., Chien, E.: W32. Stuxnet Dossier. White Paper, Security Response, Symantec Vol. 5, p. 6 (2011)
13. Faulhaber, J., Felstead, D., Henry, H., Jones, J.: Microsoft Security Intelligence Report—Zeroing in on Malware Propagation Methods (2011)
14. Federal Office for Information Security (BSI): Protection Profile for the Security Module of a Smart Metering System (Security

Module PP). Common Criteria Protection Profile BSI-CC-PP-0077-2013, Germany (2013)

15. Idiom: Binaryforest—AlienSpy Java Rat Overview (2016). http://blog.idiom.ca/2015/03/alienspy-java-rat-overview.html
16. Kamluk, V., Gostev, A.: Adwind—A Cross Plattform RAT. White Paper V. 3.0 #Adwind, Kaspersky Labs (2016)
17. Kaspersky Labs: The Regin Platform: Nation-State Ownage of GSM Networks. White Paper, Moscow (2014)
18. Kaspersky Labs: Equation Group: Questions and Answers. White Paper Report No. 1.5 #EquationAPT, Moscow (2015)
19. Kaspersky Labs: The Mystery of Duqu 2.0 a Sophisticated Cyberespionage Actor returns. White Paper, Moscow (2015)
20. Kaspersky Labs: The Flame: Questions and Answers (2016). https://securelist.com/blog/incidents/34344/the-flame-questions-and-answers-51/
21. Kaspersky Labs: Gauss: Nation-State Cyber-Surveillance meets Banking Trojan (2016). https://securelist.com/blog/incidents/33854/gauss-nation-state-cyber-surveillance-meets-banking-trojan-54/
22. Kaspersky Labs: Targeted Cyberattacks Logbook (2016). https://apt.securelist.com/#secondPage
23. Kocher, P., Genkin, D., Gruss, D., Haas, W., Hamburg, M.: Spectre Attacks: Exploiting Speculative Execution (2018). arXiv preprint arXiv:1801.01203
24. Li, P., Salour, M., Su, X.: A survey of internet worm detection and containment. Commun. Surv. Tutor. **10**(1), 20–35 (2008)
25. Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W.: Meltdown (2018). arXiv preprint arXiv:1801.01207
26. Matrosov, A., Rodionov, E., Harley, D.: Stuxnet under the Microscope. Final Report No. 1.31, eset (2011)
27. Moore, D., Shannon, C., Brown, J.: Code-red: a case study on the spread and victims of an internet worm. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment. pp. 273–284. ACM SIGCOMM/USENIX Internet Measurement Workshop, Marseille, France (2002)
28. NS-3 Consortium: ns-3 (2016). https://www.nsnam.org/
29. olsr.org: Open Link State Routing Protocol—Man Page (2004). http://www.olsr.org/docs/olsrd.conf.5.html
30. Palma, D., Curado, M.: Inside-out OLSR scalability analysis. In: Ad-Hoc, Mobile and Wireless Networks. pp. 354–359. Springer, Berlin, Heidelberg (2009). https://link.springer.com/chapter/10.1007/978-3-642-04383-3_27, https://doi.org/10.1007/978-3-642-04383-3_27
31. Riley, G., Sharif, M., Lee, W.: Simulating internet worms. In: The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings. pp. 268–274. IEEE (2004)
32. Shin, S., Gu, G., Reddy, N., Lee, C.P.: A large-scale empirical study of conficker. IEEE Trans. Inf. Forensics Secur. **7**(2), 676–690 (2012)
33. Staniford, S., Paxson, V., Weaver, N.: How to own the internet in your spare time. In: USENIX Security Symposium. pp. 149–167. San Francisco (2002)
34. Statistik Österreich: Gebäude- und Wohnungszählung 2001. Statistik Austria, Wien, oCLC: 58527040 (2004)
35. Suganthi, P., Tamilarasi, A.: Performance of OLSR routing protocol under different route refresh intervals in ad hoc networks. Int. J. Comput. Sci. Eng. **3**(1), 133–137 (2011), http://www.enggjournals.com/ijcse/doc/IJCSE11-03-01-095.pdf
36. Symantec: Regin: Top-Tier Espionage Tool Enables Stealthy Surveillance. White Paper, Symantec (2014)
37. Symantec Security Response: W32.Sality | Symantec (2016). https://www.symantec.com/security_response/writeup.jsp?docid=2006-011714-3948-99
38. Trullols-Cruces, O., Fiore, M., Barcelo-Ordinas, J.: Understanding, Modeling and Taming Mobile Malware Epidemics in a Large-Scale Vehicular Network. In: 14th International Symposium and Workshops on A World of Wireless. Mobile and Multimedia Networks (WoWMoM), pp. 1–9. IEEE, Madrid (2013)