Ad Hoc Networks 11 (2013) 2661–2674

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

SVELTE: Real-time intrusion detection in the Internet of Things

Shahid Raza [a,*], Linus Wallgren [a], Thiemo Voigt [a,b]

[a] SICS Swedish ICT, Stockholm, Sweden
[b] Department of Information Technology, Uppsala University, Sweden

ARTICLE INFO

Article history:
Available online 17 May 2013

Keywords:
Intrusion detection
Internet of Things
6LoWPAN
RPL
IPv6
Security
Sensor networks

ABSTRACT

In the Internet of Things (IoT), resource-constrained things are connected to the unreliable and untrusted Internet via IPv6 and 6LoWPAN networks. Even when they are secured with encryption and authentication, these things are exposed both to wireless attacks from inside the 6LoWPAN network and from the Internet. Since these attacks may succeed, Intrusion Detection Systems (IDS) are necessary. Currently, there are no IDSs that meet the requirements of the IPv6-connected IoT since the available approaches are either customized for Wireless Sensor Networks (WSN) or for the conventional Internet.

In this paper we design, implement, and evaluate a novel intrusion detection system for the IoT that we call SVELTE. In our implementation and evaluation we primarily target routing attacks such as spoofed or altered information, sinkhole, and selective-forwarding. However, our approach can be extended to detect other attacks. We implement SVELTE in the Contiki OS and thoroughly evaluate it. Our evaluation shows that in the simulated scenarios, SVELTE detects all malicious nodes that launch our implemented sinkhole and/or selective forwarding attacks. However, the true positive rate is not 100%, i.e., we have some false alarms during the detection of malicious nodes. Also, SVELTE's overhead is small enough to deploy it on constrained nodes with limited energy and memory capacity.

1. Introduction

With IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) [1,2] it is possible to connect resource constrained devices, such as sensor nodes, with the global Internet using the standardized compressed IPv6 protocol. These networks of resource constrained devices, also called 6LoWPAN networks, and the conventional Internet form the Internet of Things or strictly speaking the IP-connected Internet of Things (IoT). A 6LoWPAN Border Router (6BR) is an edge node that connects 6LoWPAN networks with the Internet. Due to the resource constrained nature of the devices or *things*, 6LoWPAN networks mostly use IEEE 802.15.4 as link and physical layer protocol.

Unlike typical wireless sensor networks (WSN), 6LoWPAN networks or IP-connected WSN are directly connected to the untrusted Internet and an attacker can get access to the resource-constrained *things* from anywhere on the Internet. This global access makes the *things* vulnerable to intrusions from the Internet in addition to the wireless attacks originating inside 6LoWPAN networks. Potential applications of the IoT are smart metering, home or building automation, smart cities, logistics monitoring and management, etc. These applications and services are usually charged and the revenue is based on data or services used. Hence, the confidentiality and integrity of the data and timely availability of services is very important.

Researchers have already investigated message security for the IoT using lightweight DTLS [3], IPsec [4], and IEEE 802.15.4 link-layer security [5]. Even with message security that enables encryption and authentication, networks are vulnerable to a number of attacks aimed to disrupt the network. Hence, an Intrusion Detection System (IDS)

* Corresponding author.
E-mail addresses: shahid@sics.se (S. Raza), linus@sics.se (L. Wallgren), thiemo@sics.se (T. Voigt).

1570-8705/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2013.04.014

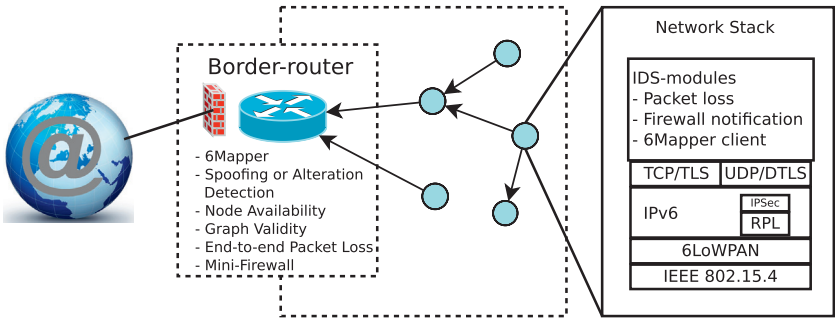


Fig. 1. An IoT setup where IDS modules are placed in 6BR and also in individual nodes.

is necessary to detect intruders that are trying to disrupt the network.

The available IDSs for WSNs could be used in the IoT. However, most of these approaches are built on the assumptions that (i) there is no central management point and controller, (ii) there exists no message security, and (iii) nodes cannot be identified globally. The IoT has a novel architecture where the 6BR is assumed to be always accessible, end-to-end message security is a requirement [5], and sensor nodes are globally identified by an IP address. Besides these opportunistic features, an IDS for the IoT is still challenging since the *things* (i) are globally accessible, (ii) are resource constrained, (iii) are connected through lossy links, and (iv) use recent IoT protocols such as CoAP [6], RPL [7], or 6LoWPAN [2]. Therefore, it is worth investigating and providing an IDS for the IoT exploiting these opportunities and threats.

To this end, we design, implement, and evaluate a novel Intrusion Detection system for the IoT that we call SVELTE.[1] To the best of our knowledge this is the first attempt to develop an IDS specifically designed for the IoT. Network layer and routing attacks are the most common attacks in low power wireless networks [8], and in this paper we primarily target these attacks. SVELTE is also inherently protected against sybil and clone ID attacks; we discuss these attacks in Section 3.2.5. We evaluate SVELTE against sinkhole and selective-forwarding attacks. Our approach is, however, extensible and can be used to detect other attacks as we discuss in Section 7.

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [7] is a novel standardized routing protocol primarily designed to meet the specific routing requirements of the IoT. SVELTE uses RPL as a routing protocol. It has two main components: the 6LoWPAN Mapper (6Mapper), and intrusion detection modules. The 6Mapper reconstructs RPL's current routing state, i.e., its directed acyclic graph, at the 6BR and extends it with additional intrusion detection parameters.

One of the important decisions in intrusion detection is the placement of the IDS in the network. We use a hybrid approach, see Section 3, and place the processing intensive SVELTE modules in the 6BR and the corresponding lightweight modules in the constrained nodes. Fig. 1 presents an overview of our IDS that we explain in more detail in Section 3. One of our main design goals is that the IDS should be lightweight and comply with the processing capabilities of the constrained nodes.

In addition to the 6Mapper and the intrusion detection techniques, we also propose and implement a distributed mini-firewall to protect 6LoWPAN networks against global attackers from Internet. We implement SVELTE in the Contiki operating system [9].

The main contributions of this paper are:

- We present SVELTE, a novel IDS with an integrated mini-firewall for the IP-connected IoT that uses RPL as a routing protocol in 6LoWPAN networks.
- We implement SVELTE and thoroughly evaluate it for 6LoWPAN networks that consist of resource-constrained *things* and have lossy communication links.

The next section of this paper gives an overview of the technologies used in SVELTE. Section 3 describes SVELTE that includes 6Mapper, the actual intrusion detection techniques, and the firewall. In Section 4 we detail SVELTE's implementation for the Contiki OS. Section 5 presents our detailed performance evaluation of SVELTE. We highlight the current IDSs and their applicability in the IoT in Section 6. Section 7 discusses the possible extensions in SVELTE, and finally we conclude the paper in Section 8.

2. Background

In this section we briefly discuss the technologies involved in SVELTE for the IoT.

2.1. The Internet of Things

The Internet of Things (IoT) or strictly speaking IP-connected IoT is a hybrid network of tiny devices, typically WSNs, and the conventional Internet. Unlike typical WSN where devices are mostly resource constrained and unlike in the Internet where devices are mostly powerful, the nodes or *things* in the IoT are heterogeneous devices. An IoT device can be a light bulb, a microwave, an electricity meter, an automobile part, a smartphone, a PC or a laptop, a powerful server machine or a cloud, or potentially anything. Hence the number of potential devices that can be connected to the IoT are in hundreds of billion. IPv6's huge address space has been designed to address this issue.

[1] SVELTE literary means *elegantly slim*.

To connect resource constrained nodes such as WSN with the Internet using IPv6, a compressed version of the IPv6 called 6LoWPAN has been standardized [1,2]. The 6LoWPAN protocol enables the routing of IPv6 packets in the IP-connected WSN (also called 6LoWPAN network) in a compressed and/or fragmented form. Compression is needed since 6LoWPAN's link and physical layer protocol, IEEE 802.15.4, has a Maximum Transmission Unit (MTU) of 127 bytes. 6LoWPAN supports multi-hop enabling nodes to forward packets on behalf of other nodes that are not directly connected to the 6LoWPAN border router (6BR). The 6BR is an end device that connects 6LoWPAN networks with the Internet.

A 6LoWPAN network is a multi-hop wireless network where communication links are usually lossy and devices are resource-constrained and often battery powered. Therefore, the connection-less User Datagram Protocol (UDP) is mostly used in 6LoWPAN networks. Further, connection oriented web protocols such as HTTP are not feasible and a new protocol, the Constrained Application Protocol (CoAP), is being standardized for the IoT. Further, a new routing protocol, IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [7], is standardized. SVELTE is primarily designed for RPL-based 6LoWPAN networks.

2.2. RPL

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [7] is a standardized routing protocol for the IP-connected IoT. RPL is a versatile protocol that enables many-to-one, one-to-many, and one-to-one communication. It creates a Destination-Oriented Directed Acyclic Graph (DODAG) and supports different modes of operation: uni-directional traffic to a DODAG root (typically the 6BR), and bi-directional traffic between constrained nodes and a DODAG root. A typical RPL DODAG is shown in Fig. 2 where each node has a node ID (an IPv6 address), one or more parents (except for the DODAG root), and a list of neighbors. Nodes also have a rank that determines their individual position with respect to the DODAG root and relative to other nodes. Ranks strictly increase from the DODAG root to nodes and strictly decrease in the direction towards the DODAG root.

Every node in the RPL network must be able to determine whether packets are to be forwarded to its parents, i.e., upwards, or to its children. The most simple way for a node to accomplish this is to know all its descendants and use the route to its parent as default route for all other packets. In-network routing tables are required to separate packets heading upwards and the packets heading downwards in the network. This is the mechanism in the RPL implementation in the Contiki operating system [9], that we use in this paper to evaluate SVELTE.

2.3. Security in the IoT

Real world IoT deployments require security. The communication between devices in the IoT can be protected on an end-to-end or on a hop-by-hop basis. IPsec [4] in transport mode provides end-to-end security between two hosts in the IoT. In IPv6 networks and hence in 6LoWPAN,

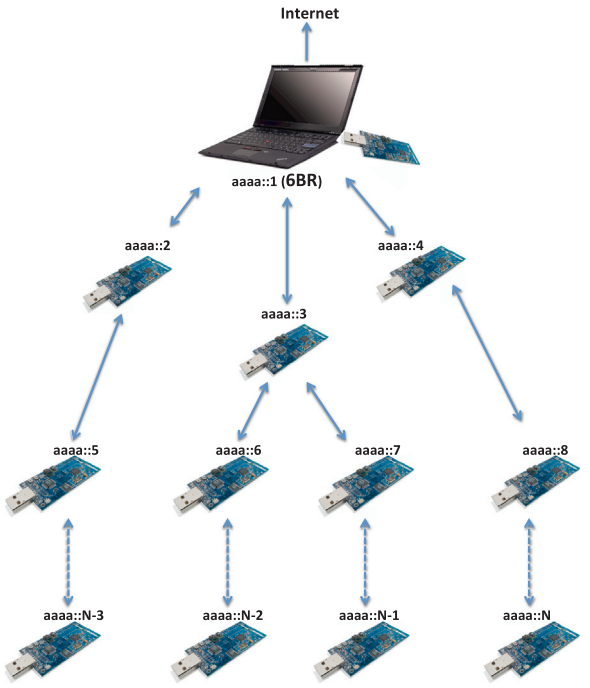


Fig. 2. A sample RPL DODAG that has N nodes with IPv6 addresses from aaaa::1 to aaaa::N.

IPsec is mandatory to implement, meaning that each IPv6-enabled device must have IPsec capabilities. IPsec's ESP protocol [10] ensures application data confidentiality and optionally data integrity and authentication, and AH [11] protocol ensures the integrity of whole IPv6 datagram that includes application data and IPv6 headers.

If the Constrained Application Protocol (CoAP) [6] is used in the IoT as an application protocol then end-to-end security between two applications can be provided with the Datagram Transport Layer Security (DTLS). Also, IEEE 802.15.14 link-layer security can be used for per hop security.

Besides having message security, the IoT is vulnerable to a number of attacks [12] aimed to disrupt the network; hence intrusion detection mechanisms are important in real world IoT deployments, e.g., in building automation, industrial automation, smart metering and smart grids.

2.4. IDS

An Intrusion Detection System (IDS) is a tool or mechanism to detect attacks against a system or a network by analyzing the activity in the network or in the system itself. Once an attack is detected an IDS may log information about it and/or report an alarm. Broadly speaking, the detection mechanisms in an IDS are either signature based or anomaly based.

Signature based detections match the current behavior of the network against predefined attack patterns. Signatures are pre-configured and stored on the device and each signature matches a certain attack. In general signature based techniques are simpler to use. They need, however, a signature of each attack and must also store it. This

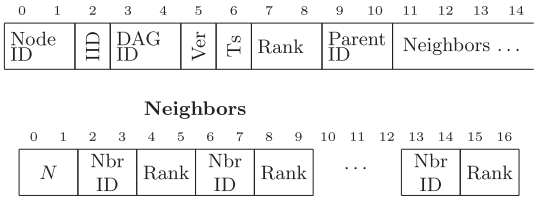


Fig. 3. Packet format of the mapping response.

requires specific knowledge of each attack and storage costs grow with the number of attacks. This approach is more static and cannot detect new attacks unless their signature is manually added into the IDS.

Anomaly based detection tries to detect anomalies in the system by determining the ordinary behavior and using it as baseline. Any deviations from that baseline is considered an anomaly. On one hand, anomaly based systems have the ability to detect almost any attack and adapt to new environments, but on the other hand these techniques have rather high false positive rates (to raise an alarm when there is no attack) as deviations from the baseline might be ordinary. Also, they have comparatively high false negative rates (no alarm when there is an attack) as attacks might only show a small deviation that is considered within the norm.

Keeping in view the novel requirements of the IoT, in this paper we use a hybrid of signature and anomaly based detections. We try to balance between the storage cost of the signature based detection and the computing cost of the anomaly based techniques. In SVELTE the detection techniques mostly target routing attacks such as sink-hole, selective forwarding, and spoofed or altered routing information [12]; however, SVELTE is extensible and can be used to detect other attacks as we discuss in Section 7.

In *sinkhole attacks* [12] an attacker advertises a beneficial routing path and thus makes many nodes route traffic through it. In RPL, an attacker can launch a sinkhole attack by advertising a better rank thus making nodes down in the DODAG select it as parent. In *selective forwarding attacks* [12], an attacker forwards only selected packets. For example, an attacker could forward only routing messages and drop all other packets to disrupt part of the network.

Once an attack is detected, the goal is to mitigate its effect and remove the attacker from the network. The simplest approach to remove an attacker is to ignore it. This requires the identification of the attacking node. Neither MAC nor IP addresses are trustworthy as they can be easily spoofed. One possible way to ignore a node is to use either a blacklist or a whitelist. A blacklist would include all malicious nodes and a whitelist would include all valid nodes. Maintaining a whitelist is easier in the presence of many attackers. In either way it is necessary that an attacker cannot obtain another valid identity, with sybil or clone ID attacks [12], as otherwise the attacker could restart the attack without effort. In SVELTE we use a whitelist.

3. SVELTE: An IDS for the IoT

Recall that a 6LoWPAN network is a lossy and wireless network of resource constrained nodes which uses IPv6 as networking protocol and often RPL as a routing protocol. One of the design goals of any protocol for the IoT is its ability to be deployed and run on constrained nodes in 6LoWPAN networks. Based on the novel requirements of the IoT, we propose SVELTE: a lightweight yet effective intrusion detection system for the IoT. We also compliment SVELTE with a distributed mini-firewall in order to filter malicious traffic before it reaches the resource constrained nodes.

We design SVELTE for a 6LoWPAN network that uses message security technologies, such as IPsec [4] and DTLS [3] to provide end-to-end message security. In the rest of this section we present our intrusion detection system.

Placement of SVELTE: The placement of an IDS is an important decision that reflects the design of an IDS and the detection approaches. Keeping in view the resource constrained nature of the devices and the IoT setup shown in Fig. 1, we use a hybrid, centralized and distributed, approach and place IDS modules both in the 6BR and in constrained nodes.

SVELTE has three main centralized modules that we place in the 6BR. The first module, called 6LoWPAN Mapper (6Mapper), gathers information about the RPL network and reconstructs the network in the 6BR, as we describe in Section 3.1. The second module is the intrusion detection component that analyzes the mapped data and detects intrusion; Section 3.2 discusses this. The third module, a distributed mini-firewall, is designed to offload nodes by filtering unwanted traffic before it enters the resource constrained network; Section 3.3 details this. The centralized modules have two corresponding lightweight modules in each constrained node. The first module provides mapping information to the 6BR so it can perform intrusion detection. The second module works with the centralized firewall. Each constrained node also has a third module to handle end-to-end packet loss; this is discussed in Section 3.2.4.

3.1. 6LoWPAN Mapper

A vital component of SVELTE is the 6LoWPAN Mapper (6Mapper) that reconstructs the RPL DODAG in the 6BR and complements it with each node's neighbor and parent information. To reconstruct the DODAG, the 6Mapper sends mapping requests to nodes in the 6LoWPAN network at regular intervals. The request packet contains the information necessary to identify an RPL DODAG. It includes the RPL Instance ID (IID), the DODAG ID, and the DODAG Version Number [7]. It also includes a timestamp (*Ts*) to know the recency of the mapping information received. The total size of a mapping request packet is 5 bytes.

Each node responds to the mapping request by perpending a Node ID to the request packet and by appending node rank, parent ID, and all neighbor IDs and ranks. An illustration of the mapping response packet format is shown in Fig. 3. The basic response packet is 13 bytes long and requires an additional four bytes for each neighbor.

6Mapper with Authentic and Reliable Communication: It is likely that IPsec Authentication Header (AH) [4] or IEEE 802.15.4 link-layer security are enabled in the IoT to protect the integrity of the IP headers. In this case there is

no need to include the node ID in the response packet, as that would be the source address in the IP header. When the 6Mapper host, i.e., 6BR, has the same IPv6 address as the DODAG root it is also unnecessary to include the DODAGID that corresponds to the destination IP in the IP header. In the request packet the source and destination fields in the IP header have the opposite meaning, i.e., the IP source corresponds to the DODAGID and the destination corresponds to the node ID.

If mapping-packets are transferred reliably, for example, by using CoAP that employs acknowledgements, there is no need to send a timestamp with the mapping data as we can be sure that the packets arrive within the timeout specified for the underlying protocol. When the communication in the 6LoWPAN is authentic and reliable, the size of the 6Mapper request and response packets is reduced to 1 byte and 8 bytes, respectively.

Unidirectional RPL 6Mapper: Some RPL implementations only support traffic destined to the DODAG root, typically the 6BR. To provide network mapping for these 6LoWPAN networks it is possible to alter the 6Mapper and let it wait for the periodic mapping response packets from each node without sending the explicit request packet. This solution has the additional advantage that it reduces traffic in the network which reduces power consumption. However, slightly more logic has to be added in each node which increases the memory consumption.

Valid inconsistencies in 6Mapper: In our 6Mapper there is a possibility that mapping responses are inconsistent with each other, which can lead to false positives if not handled properly. This can happen if the information a node sends to the network mapper has become outdated or when an attacker deliberately changes the information. Below we show how valid routing graph inconsistencies occur. Consider a RPL DODAG where Node P is the parent of node C, the function $R_a(Node)$ represents the actual rank of $Node$ and $R_m(Node)$ represents the rank known to the 6Mapper.

- Node P sends its rank to the 6Mapper, $R_a(P) = 1024$ and $R_m(P) = 1024$.
- Node P recalculates its rank and advertises it, $R_a(P) = 512$ and $R_m(P) = 1024$.
- Node C receives the updated rank from P.
- Node C recalculates its rank. $R_a(C) = 768$.
- Node C sends its rank to the 6Mapper, $R_a(C) = 768$ and $R_m(C) = 768$.

As can be seen the state of the network is:

$R_a(P) = 512$.
$R_a(C) = 768$.
$R_m(P) = 1024$.
$R_m(C) = 768$.

This state is perfectly valid as node P has a better rank than node C, $R_a(P) < R_a(C)$. However, the 6Mapper assumes that the child, node C, has a better rank than its parent, which is inconsistent as $R_m(P) > R_m(C)$.

This is a problem which needs to be taken into consideration when designing methods for analyzing the mapped data. Leveraging the amount of sensors in a 6LoWPAN we improve the accuracy when faced with both natural and artificial inconsistencies; Section 3.2.1 discusses methods to overcome such inconsistencies.

Mapping requirements: For our 6Mapper to be fully effective the packets used to map the network need to be indistinguishable from other packets. If an adversary can distinguish the traffic used by the 6Mapper from other traffic it is possible for an adversary to perform selective forwarding and only forward traffic necessary for the mapper, while dropping other traffic.

The first step to prevent this is to encrypt the data, to avoid that the packet content is revealed to an eavesdropping adversary. As mentioned earlier we assume that the message contents are protected with upper-layer security protocols such as IPsec or DTLS. Secondly, headers should not reveal any information that enables an eavesdropper to determine that the packet is used by the 6Mapper. Therefore it can be problematic if the source of the 6Mapper is the same for all nodes, as the IP header must be readable for all nodes. The adversary could use the IP header and the knowledge about the 6Mapper's host address to identify network mapping traffic. A simple solution to prevent this is to assign as many IPv6 addresses to the 6Mapper as there are nodes in the network. This is possible for RPL as IPv6 has a potentially unlimited address space of 2^{128} addresses. Thus, when an adversary compromises a node it will only know the node's mapping address and no other mapping addresses. Hence, it is not able to distinguish between ordinary traffic and mapping traffic for other nodes.

However, if the attacked node has more resources it may use more advanced traffic patterns and node behavior analysis techniques, and it might still be possible for an adversary to distinguish between ordinary and mapper-related traffic.

3.2. Intrusion detection in SVELTE

We design and implement three detection techniques which use the 6Mapper. The detection techniques primarily detect spoofed or altered information, sinkhole, and selective forwarding attacks. However, our approach is extensible and more attacks can be detected; we discuss some of the possible extensions in Section 7.

3.2.1. Network graph inconsistency detection

In the IoT individual nodes may be compromised by an attacker and later used to launch multiple attacks. For example, in RPL-based 6LoWPAN networks the attacker can use compromised nodes to send wrong information about their rank or one of their neighbor's rank to the 6Mapper. It is also possible to get an incorrect or inconsistent view of the network because of the lossy links in the IoT. It is therefore important to detect the inconsistencies, distinguish between valid and invalid consistencies, and correct the invalid information. The complete algorithm to detect and correct the routing graph inconsistencies is described in Algorithm 1.

Algorithm 1. Detect and Correct the RPL DODAG Inconsistencies

```
Require: N – A list of nodes
  for Node in N do
    for Neighbor in Node.neighbors do
      Diff = |Node.neighborRank
  (Neighbor) − Neighbor.rank|
      Avg = (Node.neighborRank
  (Neighbor) + Neighbor.rank)/2
      {If the absolute difference is greater than 20% of
  the ranks average}
      if Diff > Avg ∗ 0.2 then
        Node.fault = Node.fault + 1
        Neighbor.fault = Neighbor.fault + 1
      end if
    end for
  end for
  for Node in N do
    if Node.fault > FaultThreshold then
      Node.rank = Rank reported for Node by any
    neighbor
      for Neighbor in Node.neighbors do
        Node.neighborRank (Neighbor) = Neighbor.rank
      end for
    end if
  end for
```

In order to *detect* incorrect information and to make sure that information is consistent across the network, each edge in the network is checked. The 6Mapper provides node ID and rank of each node, of its parents, and of its neighbors. We iterate over each edge in the network, checking that both nodes agree with each other about their rank and detect the inconsistencies. It is possible that a false alarm is raised because the detected incorrect information is a result of valid mapping inconsistencies described in Section 3.1.

In order to *distinguish* between valid and invalid inconsistencies, or to avoid false positives, we rely on (i) the number of reported faulty ranks and (ii) the difference between the two reported ranks. We use a simple threshold, referred to as *FaultThreshold* in Algorithm 1, and classify a node as faulty if the number of disagreements this node has with other nodes are larger than the threshold. Most of the disagreements between two nodes are small and a result of varying link quality and ultimate RPL adjustments. To accommodate valid inconsistencies, we only consider disagreements where the difference of the two nodes ranks is greater than 20% of the ranks average; this value is based on our empirical evaluation of SVELTE.

We *correct* the faulty information when both of the above conditions are met, i.e., once we have large inconsistencies towards a node. The faulty information corresponding to a node is corrected by changing the rank known to 6Mapper by substituting it with the information reported by one of its neighbors. The neighbor information is updated with the information reported directly by its neighbors.

Once it is detected that a routing inconsistency is a result of a deliberate attack, SVELTE either removes the faulty node or corrects the inconsistency. SVELTE keeps track of inconsistencies and if it is the first time a node is detected as malicious it is not immediately removed as it may be a false alarm or result of a passive attack; in this case the faulty information is corrected as described above. However, if the same node is detected as faulty again it is removed by deleting its entry from the whitelist maintained in the 6Mapper.

3.2.2. Checking node availability

It is important to detect if a node or set of nodes are available and operating properly. When a particular node is compromised it may launch multiple attacks to disrupt the network. For example, it may launch a selective forwarding attack and intelligently drop messages. If an RPL network uses CoAP to send application data the attacker could forward RPL traffic but drop CoAP traffic. This would result in a seemingly working network even though no useful traffic gets through.

Depending on the RPL implementation and the configuration, we can use the RPL routing table in the RPL DODAG root as a basis for available nodes in the network. As we require a whitelist of valid nodes in the network for access control we could also use that list as a basis for detection.

When we compare the whitelisted nodes with the nodes in our RPL DODAG all differences are offline nodes or unauthorized nodes. Let W be a set of all whitelisted nodes and let R be the nodes known to RPL in the RPL DODAG root, the offline nodes, O are thus:

$$W \setminus R = O$$

where O is the relative complement ($\setminus$) between two sets W and R meaning that O contains all elements of W that are not in R.

It is however not possible to determine if nodes excluded from O are being filtered or are simply offline. That is, if an attacker performs a selective forwarding attack and filters everything but RPL messages it would with the previous method appear as if the nodes are still online, even though all application data is being filtered. By extending the above method with the information available through 6Mapper it is also possible to detect selective forwarding attacks. Let M represent nodes known to 6Mapper and F be the filtered nodes we get the following relationship:

Algorithm 2. Detect Filtered Nodes

```
Require: W – Set of whitelisted nodes
Require: M – Set of nodes known to the 6Mapper
  F = [] {F will contain the filtered nodes}
  for Node in W do
    if Node in M and M[Node].lastUpdate
  () > RecencyThreshold then
      F.add (Node)
    end if
end for
return F
```

$$W \setminus M = F$$

As the 6Mapper for each node keeps track of the last time it received a packet from a node we can detect filtered nodes by

simply checking if we have not recently received any packets from them. In order to mitigate the effects of packet loss or other similar events common in lossy networks we introduce a threshold on the time since our last packet. We define the threshold as a number of mapping-requests allowed to be unanswered. With this threshold it is possible to alter the sensitivity of the filtered node detection to be easily adaptable to specific deployments. Algorithm 2 describes this behavior and finds all filtered nodes F in a network.

3.2.3. Routing graph validity

By artificially altering the routing graph, an attacker can reshape the topology of the network and can control the traffic flow to his advantage. For example, an attacker performs a sinkhole attack by advertising a very good rank to its neighbors. The problem becomes more severe if the sinkhole attack is coupled with other attacks. A sinkhole attack can, for example, enable the attacker to intercept and potentially alter more traffic than otherwise. If combined with a selective forwarding attack a much larger part of the network can be controlled. It is therefore important to detect such attacks.

With SVELTE, it is possible to detect most sinkhole attacks by analyzing the network topology. If the routing graph is inconsistent it is likely an attack is in place. In RPL, the rank in the network should be decreasing towards the root, i.e., in any child-parent relation the parent should always have a lower rank than the child. All cases where a child has a better rank than its parent is an indication of routing graph incoherency, as specified in [7].

When an incoherency is found the child in the relation is at fault, as a node should never have a lower rank than its parent. With such a simple approach false positives are likely to arise, i.e., we detect inconsistencies while in fact all nodes are working properly.

In order to minimize the effects of valid inconsistencies, that can raise false positives, we require several consecutive inconsistencies to be reported for the same nodes. That is we require more than one sample of the network to have the same incoherency to raise an alarm. This is described in Algorithm 3 as *FaultThreshold* which is a global state kept between consecutive runs of the detection algorithm. In RPL the rank between any host and its parent is at least *MinHopRankIncrease* [7]. We utilize this in our algorithm to better conform to the RPL standard.

Algorithm 3. Finding Rank Inconsistencies

```
Require: N – A list of nodes
  for Node in N do
    if
   Node.rank + MinHopRankIncrease < Node.parent.rank
   then
      Node.fault = Node.fault + 1
    end if
end for
for Node in N do
    if Node.fault > FaultThreshold then
      Raise alarm
    end if
end for
```

A sinkhole attack would in most cases be detected by this algorithm. As the attacker advertises a beneficial rank it will most likely have to advertise a better rank than its parent and as such would be detected by the detection scheme described above. If a sinkhole attack is to remain undetected the advertised rank of a malicious node must not be better than that of its parent. This would in turn result in the adversary's rank only being slightly improved over a non-adversarial node and thus yield little benefit.

In RPL, the rank as well as the parent selection is calculated via an objective function, which might use factors such as link quality in its calculation; for example when the Expected Transmission Count (ETX) [13] is used to calculate rank. The ETX is an approximation of the link quality and as such a bad link might affect the choice of parent more than a slight difference in rank. This would further lower the impact of a sinkhole attack that is undetectable by Algorithm 3.

Algorithm 4. Adapt to End-to-end Losses

```
Require: dest – The destination with packet loss
  nexthop = getNexthop (dest)
  nexthop.metric = nexthop.metric ∗ 0.8
```

3.2.4. End-to-end packet loss adaptation

We design an intentionally simple system to take end to end losses into account when calculating the route and to mitigate the effects of filtering hosts. If a reliable higher layer protocol such as TCP or CoAP (with *confirmable* messaging) is used, packet loss can be detected using the protocol's acknowledgement mechanism. The reasoning behind a host-to-host packet loss indication is that if an attacker is filtering packets some hops down the path we want to be able to adapt to it. In the RPL-based network, if a packet is filtered somewhere on the path a new parent should eventually be tried.

The approach is not able to adapt to every form of filtering, for example, when the attacker is located such that all packets have to go through it. If however a collection-scheme with acknowledgements is also running in the network all data losses should be corrected for. Since all nodes will try to send data to the sink all nodes with a path through the attacker will also notice the losses and correct for them, given that the attacker filters all application data. If a packet is not able to reach its destination, we slightly alter the route metric of the route, that is the next-hop neighbor for that packet, (20% in Algorithm 4) to reflect that there might be an attacker along the path. Algorithm 4 describes end-to-end packet loss adaptation.

3.2.5. Sybil and CloneID attacks protection

In a Sybil attack an attacker copies several logical identities on one physical node whereas in a cloned identity (CloneID) attack the attacker copies the same logical identity on several physical nodes. Both attacks are aimed to gain access to a large part of the network or in order to overcome a voting scheme. The 6Mapper only considers the latest information received from each host in the net-

work where a host is identified by an IP address. A sybil attack has no direct effect on the 6Mapper as it makes no difference if the identities are on the same physical node as if they are separate physical entities, each host is treated individually in both cases. While cloned identities can interrupt the routing in a network it does not affect the 6Mapper directly as the 6Mapper only considers the latest information received from one of the identity. As a result if two cloned nodes send information to the 6Mapper there is no difference compared to if one node sends the information twice, thus not directly affecting the operations of the 6Mapper. Sybil attacks and cloned identities are both often used to disrupt different voting schemes by giving an attacker more votes. Voting schemes based upon 6Mapper collected data will be unhindered by both sybil attacks and cloned identities.

3.3. Distributed mini-firewall

Though SVELTE can protect 6LoWPAN networks against in-network intrusion, it is also important that the resource constrained nodes are protected against global attackers that are much more powerful. For example, it is easier for hosts on the Internet than constrained nodes in 6LoWPAN networks to perform denial of service attacks. Firewalls are usually used to filter external hosts and/or messages destined to local networks. As the end-to-end message confidentiality and integrity is necessary in the IoT, the SVELTE module in the 6BR or a firewall cannot inspect the contents of the encrypted messages; therefore, it is hard to distinguish between the legitimate and malicious external traffic.

We propose a distributed mini-firewall that protects a 6LoWPAN network from external hosts. The firewall has a module in the 6BR and in the constrained nodes, and is integrated with SVELTE. Our firewall, besides providing typical blocking functionally against well-known external attackers specified manually by the network administrator, can block the external malicious hosts specified in *real-time* by the nodes inside a 6LoWPAN network.

The destination host inside a 6LoWPAN node can see the encrypted contents and hence analyze the malicious traffic and notify the 6BR in real-time to filter traffic coming from the compromised host, therefore stopping the traffic before it reaches the constrained nodes. When a constrained node notices an external host being abusive it sends a packet with the host IP to the firewall module in the 6BR. As is the case with the 6Mapper, if IPsec with Authentication Header is used the nodes own ID can be omitted. Otherwise, the nodes own ID need to be included. If the node ID is included it can be compressed down to 2 bytes using 6LoWPAN header compression mechanisms. The external host however can neither be compressed nor omitted as it can be any valid IPv6 address. Therefore the minimal size of the filtering-request packet is 16 bytes. With the node ID the size of the packet is 18 bytes.

In order to make sure that no internal compromised node can abuse this mechanism by requesting filtering of traffic from a legitimate external host, both the source and the destination is taken into account when filtering. The node inside a 6LoWPAN network can only choose to filter the traffic destined to itself. Such a firewall is still easy to circumvent as the attacker can simply target another node in the network and start the attack again; therefore, we extend the firewall to adapt and block any external host if a minimum set of nodes complain about the same external host. Our mini-firewall is described in Algorithm 5.

Algorithm 5. Mini-firewall

```
Require:Host – The host to report
Require:Source – The node that sent the report
Require:GlobalFilter – A set of external hosts to filter
  towards all nodes
Require:LocalFilter – A map mapping an external host
  to a set of local nodes. The set describes all nodes
  that have reported that specific external host.
ifHost in GlobalFilterthen
  return Host already filtered
end if
ifHost in LocalFilterthen
  Filter = LocalFilter.get (Host)
  {Add Source to the list of nodes blaming Host}
  Filter.add (Source)
  ifFilter.size () ⩾ ReportThresholdthen
    GlobalFilter.add (Host)
    LocalFilter.remove (Host)
  end if
end if
```

To be more *preventive* against global attackers, our mini-firewall can be extended with AEGIS [14], a rule-based firewall for wireless sensor networks.

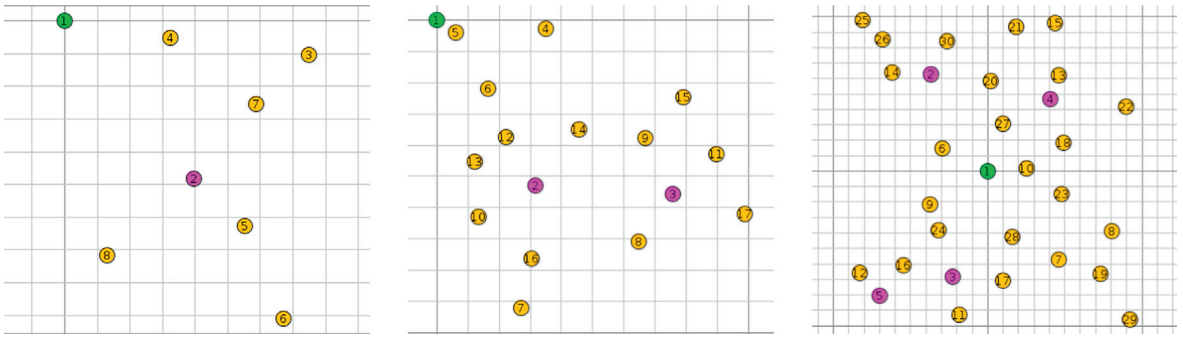
4. Implementation

We implement SVELTE and the mini-firewall in the Contiki OS [9], a well known operating system for the IoT. Contiki has a well tested implementation of RPL (ContikiRPL). As SVELTE is primarily designed to detect routing attacks we make use of the RPL implementation in the Contiki operating system to develop the 6Mapper, the firewall, and the intrusion detection modules. The RPL implementation in Contiki utilizes in-network routing where each node keeps track of all its descendants. We borrow this feature to detect which nodes should be available in the network. To provide IP communication in 6LoWPAN we use μIP, an IP stack in Contiki, and *SICSLoWPAN*- the Contiki implementation of 6LoWPAN header compression. We also implement the sinkhole and selective forwarding attacks against RPL to evaluate SVELTE. SVELTE is open source[2] and is available to researchers and industry.

5. Evaluation

In this section we present the empirical evaluation of SVELTE. After describing our experimental setup, we quan-

[2] For the source code visit: http://www.shahidraza.info.



(a) 8 nodes with 1 (ID 1) malicious node.

(b) 16 nodes with 2 (ID 2 and 3) malicious nodes.

(c) 32 nodes with 4 (ID 2, 3, 4, 5) malicious nodes.

Fig. 4. Network configurations and node placement that are used in the experiments in this section.

titatively evaluate the detection rate and the true positives for each experiment. We also measure the overhead of SVELTE both at the node-level and network-wide. We evaluate the overhead in terms of energy consumption and the memory footprint.

5.1. Experimental setup

We run our experiments in Contiki's network simulator Cooja [15] that has shown to produce realistic results [16]. Cooja runs deployable Contiki code. In our simulations, we use emulated Tmote Sky [17] nodes.

In general, we expect that the 6BR is not a constrained node and it can be a PC or a laptop; however, currently there exists no PC equivalent 802.15.4 devices, therefore we run the 6Mapper natively, i.e., on Linux, and communicate with Cooja using a *serial socket*. For RPL with 6Mapper we run each test 10 times, and calculate the average and standard deviation to show the accuracy and precision of our results. On the other hand, the experiments with RPL only (without the 6Mapper) have no processing intensive components and hence require no native parts. Therefore, the experiments with RPL-only yield the same results for all experiments as we use the same seed.

5.2. SVELTE detection and true positive rate

Here we quantitatively evaluate the detection rate, i.e., the number of malicious nodes successfully detected against the total number of malicious nodes present in the system, and the true positives rate, i.e., the total number of successful alarms divided by the total number of alarms. We use three different configurations shown in Fig. 4a–c. In each configuration node no 1 (green[3]) is the 6BR. Using these settings, we run experiments for 5, 10, 20, and 30 min. In all experiments, the 6Mapper is configured to request data and to perform analysis every 2 min. Therefore, the first 6Mapper request will be sent after 2 min. The first analysis is also performed after 2 min but will however not yield any results as no data is yet gathered. Therefore the earliest possible detection time is after 4 min.

[3] For interpretation of color in Fig. 4, the reader is referred to the web version of this article.

It is important to note that these are the settings in our experiments and not the requirements for SVELTE. The malicious nodes can spoof or alter information, and/or can perform sinkhole or selective forwarding attacks. In the following experiments SVELTE first performs network graph inconsistency detection as described in Section 3.2.1, before detecting sinkhole or selective forwarding attacks. Each experiment is run in a lossy and in a lossless network. Lossless links provide the perfect scenario for 6Mapper, as all requests and responses return without delay and loss, and we get a true picture of the network. This is further improved by the fact that nodes more quickly can propagate their ranks down in the network graph. The real 6LoWPAN networks are mostly lossy, therefore we consider both cases in our evaluation. The loss model is Cooja's default radio model that uses a Unit Disk Graph Medium (UDGM): Distance Loss [15]. UDGM models the transmission range as a circle in which only the nodes inside the circle receive packets. The UDGM Distance Loss model, an extension of UDGM, also considers interference.

Sinkhole Attacks with and without losses: The results for the sinkhole attack in a lossless network scenario show almost 100% true positive rate on the first possible attempt to analyze the network and no false positives are detected during the simulations. A lossless network configuration means that all requested data is gathered quickly and without losses, which implies that the map of the network is a perfect representation of the actual network. Because of this it is very easy to detect all sinkhole attacks without any false positives. In the lossy network configuration, Fig. 4a, the true alarm rate is approximately 90%, as shown in Fig. 5. However, with the increase in network size the true alarm rate decreases; this is because for the larger network configurations it takes some time before the RPL network and our map of the network become stable and complete enough to arrive at a higher true positive rate. For example, in the scenario with 16 nodes it takes 30 min to arrive at the same true positive rate as is done with 8 nodes after 10 min. The reason Fig. 5 shows a non-existent detection rate for the case of 5 min is because we only raise an alarm if the same node has been misbehaving for more than two consecutive executions of our algorithm. Hence, the current configuration implies that a sinkhole attack can be detected after 6 min. Our approach

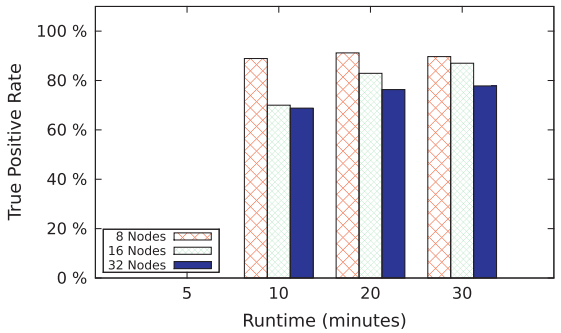


Fig. 5. For the smaller lossy network, SVELTE has 90% true positive rate against sinkhole attacks which decreases for larger networks but gets better when RPL becomes stable.

does not require collection of two consecutive messages or executions to work. Collecting multiple messages is advantageous to make sure that it was actually an attack and not a sudden link fluctuations, for example due to interference. If the attack persists for two consecutive executions of our algorithms then we raise an alarm; this is done primarily to reduce false positives.

From these results it is evident that SVELTE is very effective against sinkhole attacks in a network with no or few losses, and in lossy networks it is more effective when the RPL network has become stable.

Selective Forwarding Attack with and without losses: In a selective forwarding attack a malicious node filters traffic going through it. Hence, the 6Mapper will not be able to get any data from any children of the malicious nodes in the network. This in turn has the effect that the results of the 6Mapper depend on the actual network topology, i.e., in the lossless case, unlike with sinkhole attacks, the results are not always 100%. We can see the effects of this phenomenon in Fig. 6a. In a lossy network, as shown in Fig. 6b, there is a gradual increase in the true positive rate going towards a bit over 80% in all cases. As the network is lossy messages are naturally lost, and if that happens several consecutive times when mapping we are going to get more false positives. If we raise the various thresholds in our detection algorithms it is possible to lower the number of false alarms, possibly at the cost of a decreased detection rate. In order to reduce number of false positives we may use location information of the nodes as discussed in Section 7.

We also measure the detection rate during all of the above experiments. We achieve 100% *detection rate* meaning that we can detect all malicious nodes that launch sinkhole and/or selective forwarding attacks. It should be noted that the 100% detection rate is for the current set of experiments with the current setting; we do not claim that SVELTE should achieve 100% detection rate in all settings. As can be seen in Figs. 5 and 6 the *true positive rate* is not 100%, i.e., we have some false alarms during the detection of malicious nodes. This is mostly caused by our configuration. It might be possible to alter the behavior of our detection algorithm, for example, by changing the threshold used in Algorithm 2 and thus possibly get a different result with regards to detection rate and/or false alarm rate.

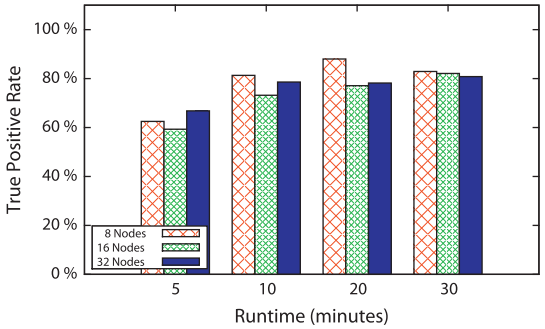
5.3. Energy overhead

The nodes in the IoT are usually battery powered and hence energy is a scarce resource. Here we measure SVELTE's power consumption both at node-level and at system-level. We use Contiki Powertrace [18] to measure the power consumption. The output from the Powertrace application is the total time the different parts of the system were on.

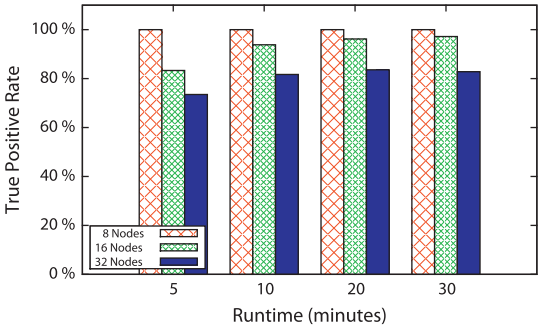
We calculate the energy usage and power consumption using the nominal values, the typical operating conditions of the Tmote sky, shown in Table 1. We use 3 V in our calculations. In the rest of this paper MCU idle while the radio is off is referred to as low power mode, or *LPM*. The time the MCU is on and the radio is off is referred to as *CPU* time. The time the radio is receiving and transmitting with the MCU on is referred to as *listen* and *transmit* respectively. We measure energy in both duty cycled 6LoWPAN networks, where the radio is mostly off, and in non-duty cycled networks where the radio is always on for listening and transmitting.

5.3.1. Network-wide with duty cycling

Here we evaluate network-wide energy consumption of an RPL network with and without the 6Mapper and intrusion detection mechanisms in a duty cycled network. We use ContikiMAC [19], a duty cycling MAC protocol in Con-



(a) *Lossy* network suffering from selective forwarding attack



(b) *Lossless* network suffering from selective forwarding attack

Fig. 6. SVELTE has acceptable true positive rate in both lossy and lossless network considering that we have almost 100% detection rate for selective forwarding attacks.

Table 1
Operating conditions in Tmote sky

Typical conditions	Min	NOM	Max	Unit
Voltage	2.1		3.6	V
Free air temperature	−40		85	C
MCU on, Radio RX		21.8	23	mA
MCU on, Radio TX		19.5	21	mA
MCU on, Radio off		1800	2400	μA
MCU idle, Radio off		54.5	1200	μA
MCU standby		5.1	21.0	μA

tiki. We use the default ContikiMAC setting that has 8 wakeups per second and without traffic the radio is on for 0.6% of the time. We run each experiment in a network of 8, 16, 32 and 64 emulated Tmote sky nodes, with nodes placed at the same locations.

Fig. 7a shows the network-wide energy usage for 30 min by all the nodes, calculated as follows

$$Energy(\text{mJ}) = (transmit * 19.5\ \text{mA} + listen * 21.8\ \text{mA} + CPU * 1.8\ \text{mA} + LPM * 0.0545\ \text{mA}) * 3\ \text{V}/4096 * 8$$

From the network wide energy usage, we calculate the average power as:

$$Power(\text{mW}) = \frac{Energy(\text{mJ})}{Time(\text{s})}$$

which when divided by the total number of nodes gives us the per node average power consumption during the experiment. Fig. 7b shows the power consumption per node. As can be seen in Fig. 7a and b the overhead of the 6Mapper is negligible for small networks (up to 16 nodes) and increases with the number of nodes. The total overhead of SVELTE is approximately 30% more than running RPL only for networks with 64 nodes. Recall that with duty cycling the radio is off for approximately 99% of the time.

5.3.2. Network-wide without duty cycling

We use the same network settings as in Section 5.3.1 and run the experiments in a non-duty cycled network where the radio is always turned on to receive and transmit packets. When we compare the results of RPL with the 6Mapper plus intrusion detection algorithms we see that the overhead is negligible. This is because the radio is always on and most of the nodes' energy is consumed on idle listening.

Table 2
Energy consumption for handling a single event inside a constrained node.

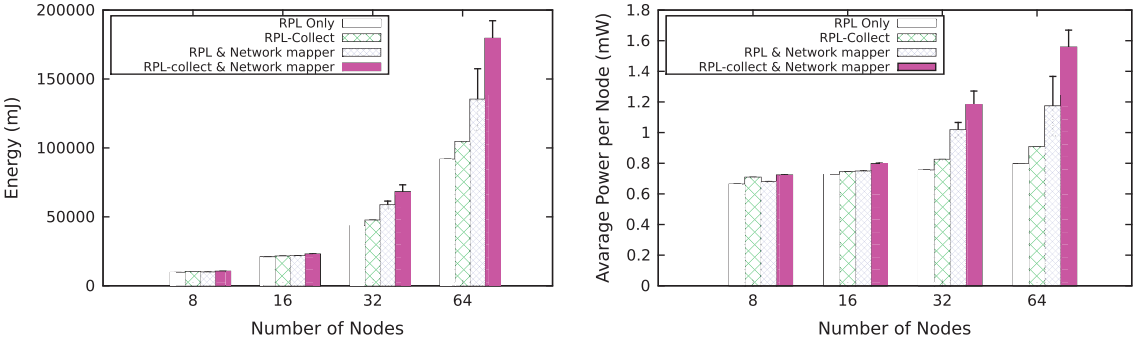
Event	Energy (mJ)
6Mapper response handling	0.1465
Firewall handling	0.0478
Packet lost correction	0.0483

5.3.3. In-node energy overhead

Here we measure the energy consumption of handling a single event of the 6Mapper and the firewall inside a constrained node. Table 2 lists the energy required to perform different tasks; this does not include the energy needed to send/receive packets which we have included in Sections 5.3.1 and 5.3.2. As can be seen in Table 2, a constrained node consumes very little energy for local processing as most of the processing intensive tasks are performed in the 6BR where the 6Mapper and the main SVELTE detection modules reside. Therefore, the energy consumed for in-node processing is clearly negligible.

5.4. Memory consumption

In Table 3 we show the extra ROM requirements of SVELTE's different modules. The baseline for each configuration is different as some depend on different parts of the Contiki system. For example, the 6Mapper that resides in the 6BR (typically a PC) requires more ROM than other nodes. However, the total additional ROM required to host SVELTE's modules inside a constrained node is 1.76*k* which is well below the total available ROM in constrained devices such as 48*k* in Tmote sky. In Table 3 it is important to note the overhead column which shows the pure overhead of SVELTE modules in Contiki. Even though 6Mapper is not targeted towards running on constrained nodes it is



(a) Energy usage for the entire network (with *duty cycling*) in 30 minutes.

(b) Average power consumption per node in a *duty cycled* RPL-based network

Fig. 7. Network-wide energy usage in a *duty cycled* RPL-based network of different sizes shows that for network with less nodes SVELTE overhead is very small; however, per node overhead grows with the increase in number of nodes as more and more nodes act as routers.

Table 3
Out of total 48*k* of ROM size in a constrained device (Tmoke sky), SVELTE requires 1.76*k*. However, in the 6BR (typically a PC) the size grows when the number of nodes increases.

Configuration	Total ROM (byte)	Overhead (byte)
6Mapper client	44,264	1414
Firewall client	43,556	0246
Packet loss improvement	43,264	0122
6Mapper server (1 node, 1 neighbor)	46,798	3580
6Mapper server (8 node, 1 neighbor)	46,798	3846
6Mapper server (16 nodes, 1 neighbor)	46,800	4152
6Mapper server (16 nodes, 8 neighbors)	46,924	4724

Table 4
Additional RAM usage by SVELTE for handling a single event inside a constrained node.

Event	RAM (byte)
6Mapper response handling	162
Firewall handling	24
Packet lost correction	188

still lightweight enough and can be used for small networks.

We also measure the RAM size of 6Mapper response handling, firewall, and packet loss correction which we show in Table 4. The total RAM size in the Tmoke sky is 10 kb, hence SVELTE modules with 0.365*k* additional RAM requirement can easily run in constrained nodes.

6. Related work

The IoT is a rather old concept and for many years RFID-based sensors were considered as *things* in the IoT. With the inception of 6LoWPAN, lightweight IP is being standardized and used in the IoT for the unique identification and global connectivity of the *things*. Even when confidentiality and integrity are enforced by message security solutions such as IPsec [4] it is possible to disrupt the IoT. A number of attacks against the IoT have been identified [8] in addition to those against WSN [12] that are also applicable to the IoT. Therefore, it is important to have systems that detect such attacks.

The concept of intrusion detection is quite old and extensive research is carried out in this field mostly against the Internet attacks and attacks against WSN. However, no IDS are specifically designed in the context of IoT. Most of the IDS approaches for WSN are based on a distributed architecture and are built on the limitation that there is no centralized management and control point. A common IDS approach for WSNs is to utilize several special nodes distributed evenly throughout the network. These special nodes can either be physically different [20] or dynamically distributed throughout the network [21,22]. In real deployments, however, it cannot be guaranteed that particular nodes are always present in specific locations in the network; also, the cost of employing mobile agents that move through the network might be too high. Clustering based approaches have similar issues as each cluster often requires a powerful entity for coordination [23]. The IoT has a novel architecture where the 6BR is always assumed to be accessible and is a potential place for centralized management and control. SVELTE make use of this novel IoT architecture and presents a new placement for IDS. Using a mix of centralized and distributed architecture SVELTE takes advantage of both realms.

Many IDS approaches are based upon watchdog techniques [21,24] which could be used in the IoT. In addition to being distributed and fully deployed on sensor nodes, a general problem with watchdog based approaches is that they require promiscuous listening, which consumes a lot of power and therefore is not suitable for constrained devices. Advanced anomaly detection approaches are proposed [25,26], not primarily for WSNs, which on one hand can detect many intrusions efficiently but on the other hand requires intelligent learning, which is both expensive and difficult in low powered 6LoWPAN networks.

Most current IDS approaches require different routing schemes that are not based on standardized mechanisms. As far as we are aware, no approach is built around 6LoWPAN and RPL in the context of the IoT. Our approach considers RPL to decrease the cost of performing intrusion detection. Likewise, we have taken into account the fact that there is a central entity, the 6BR, that connects the sensor network with the conventional Internet, which is a standard based networking solution [1,2,7].

We do not claim that no other IDS approach can be used in the RPL-connected IoT. Rather we argue that these approaches are built on different assumptions that do not fully hold in the IoT architecture. Also, the IoT gives rise to new challenges that do not exist in typical WSNs. However, there is a potential to incorporate already available approaches in the SVELTE *architecture*. We discuss below the possibilities to integrate available lightweight IDS approaches in SVELTE.

7. SVELTE extensions

One of the main advantages of our approach to intrusion detection is that the proposed and developed system is very easy to extend. There are a number of potential attacks against the Internet of Things and it is likely that more attacks will be discovered. As such extendability is very important for an IDS. The 6Mapper is easy to extend both conceptually and in practice. If a new detection scheme requires more data to be added to the network graph the response packets can easily be extended. Also, using the already available data that we collect through the 6Mapper it is possible to apply anomaly detection techniques, for example via the use of Support Vector Machines [27], feature vectors [28], or automata based approach [29].

Wormhole detection: One of the important to detect attacks in wireless networks is wormhole [30]. If the 6Mapper is extended with the signal strength of each node's neighbor it is also possible to detect wormhole attacks [31].

Pinpointing filtering node: If a node is filtering traffic it is beneficial to be able to pinpoint more accurately which node is performing the filtering. The most straight forward approach is to perform a traceroute [32] towards one of the missing nodes.

Location information: RPL is primarily designed for static networks, though it can be extended to support mobility [33], it is possible to add node's location in the 6Mapper at the deployment time. The location of a node can also be estimated in real-time using localization techniques [34]. These location information help SVELTE to build a physical map of the network that will ultimately enhance its intrusion detection capabilities. For instance, with this physical map rank modification and hence the sink-hole attack can be detected with even lesser false positives alarms. The location information of nodes will also help SVELTE to mitigate the sybil and CloneID attacks aimed to disrupt the routing information [35].

8. Conclusions

6LoWPAN networks will be an integral part of the IoT. Considering the potential applications of the IoT it is important that 6LoWPAN networks are protected against internal and external intrusions. To this end we present SVELTE, the first IDS for the IoT which consists of a novel architecture and intrusion detection algorithms. We implement and evaluate SVELTE and show that it is indeed feasible to use it in the context of RPL, 6LoWPAN, and the IoT. To guard against global attacks we also design and implement a mini-firewall.

The detection algorithms in SVELTE currently target spoofed or altered information, sinkhole and selective forwarding attacks. However, it is flexible and can be extended to detect more attacks. Therefore, we plan to complement SVELTE with novel and/or available intrusion detection techniques that are feasible to use in the context of the IoT.

Acknowledgements

This work was financed by the SICS Center for Networked Systems (CNS), SSF through the Promos project, and CALIPSO, Connect All IP-based Smart Objects, funded by the European Commission under FP7 with contract number FP7-ICT-2011.1.3-288879.

References

[1] J. Hui, P. Thubert, Compression Format for IPv6 Datagrams Over IEEE 802.15.4-Based Networks, RFC 6282, September 2011.
[2] N. Kushalnagar, G. Montenegro, C. Schumacher, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, RFC 4919, August 2007.
[3] T. Kothmayr, W. Hu, C. Schmitt, M. Bruenig, G. Carle, Securing the internet of things with DTLS, in: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, ACM, 2011, pp. 345–346.
[4] S. Raza, S. Duquennoy, A. Chung, D. Yazar, T. Voigt, U. Roedig, Securing communication in 6LoWPAN with compressed IPsec, in: 7th International Conference on Distributed Computing in Sensor Systems (DCOSS'11), Barcelona, Spain, 2011, pp. 1–8.
[5] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, T. Voigt, Secure Communication for the Internet of Things – A Comparison of Link-Layer Security and IPsec for 6LoWPAN, Security and Communication Networks, Wiley. http://dx.doi.org/10.1002/sec.406.
[6] Z. Shelby, K. Kartke, C. Bormann, B. Frank, Constrained Application Protocol (CoAP), draft-ietf-core-coap-12, October 2012.
[7] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550, March 2012.
[8] O. Garcia-Morchon, R. Hummen, S. Kumar, R. Struik, S. Keoh, Security Considerations in the IP-Based Internet of Things, draft-garcia-core-security-04, March 2012.
[9] A. Dunkels, B. Grönvall, T. Voigt, Contiki – a lightweight and flexible operating system for tiny networked sensors, in: EMNets'04, Tampa, USA, 2004, pp. 455–462.
[10] S. Kent, R. Atkinson, IP Encapsulating Security Payload (ESP), RFC 2406, Obsoleted by RFCs 4303, 4305, November 1998.
[11] S. Kent, R. Atkinson, IP Authentication Header, RFC 2402, Obsoleted by RFCs 4302, 4305, November 1998.
[12] C. Karlof, D. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, Ad Hoc Networks 1 (2) (2003) 293–315.
[13] D. Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, Wireless Networks 11 (4) (2005) 419–434.
[14] M. Hossain, V. Raghunathan, Aegis: a lightweight firewall for wireless sensor networks, Distributed Computing in Sensor Systems (2010) 258–272.
[15] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with Cooja, in: Proceedings of 31st IEEE Conference on Local Computer Networks, IEEE, 2006, pp. 641–648.
[16] F. Österlind, Improving Low-Power Wireless Protocols With Timing-Accurate Simulation, Ph.D. Thesis, Uppsala University, 2011.
[17] J. Polastre, R. Szewczyk, D. Culler, Telos: enabling ultra-low power wireless research, in: IPSN'05, 2005, pp. 364–369.
[18] A. Dunkels, J. Eriksson, N. Finne, N. Tsiftes, Powertrace: Network-Level Power Profiling for Low-Power Wireless Networks, 2011. <http://www.soda.swedish-ict.se/4112/>.
[19] A. Dunkels, The Contikimac Radio Duty Cycling Protocol, 2011. <http://www.soda.swedish-ict.se/5128/>.
[20] I. Atakli, H. Hu, Y. Chen, W. Ku, Z. Su, Malicious node detection in wireless sensor networks using weighted trust evaluation, in: Society for Computer Simulation International Proceedings of the 2008 Spring Simulation Multiconference, 2008, pp. 836–843.
[21] R. Roman, J. Zhou, J. Lopez, Applying intrusion detection systems to wireless sensor networks, in: Proceedings of IEEE Consumer Communications and Networking Conference, 2006, pp. 640–644.
[22] T. Hai, E. Huh, M. Jo, A lightweight intrusion detection framework for wireless sensor networks, Wireless Communications and Mobile Computing 10 (4) (2009) 559–572.
[23] C. Rong, S. Eggen, H. Cheng, An efficient intrusion detection scheme for wireless sensor networks, Secure and Trust Computing, Data Management, and Applications 187 (2011) 116–129.
[24] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom '00, ACM, New York, NY, USA, 2000, pp. 255–265.
[25] A. Mishra, K. Nadkarni, A. Patcha, Intrusion detection in wireless ad hoc networks, Wireless Communications, IEEE 11 (1) (2004) 48–60.
[26] K. Hwang, M. Cai, Y. Chen, M. Qin, Hybrid intrusion detection with weighted signature generation over anomalous internet episodes, IEEE Transactions on Dependable and Secure Computing 4 (1) (2007) 41–55.
[27] S. Kaplantzis, A. Shilton, N. Mani, Y. Sekercioglu, Detecting selective forwarding attacks in wireless sensor networks using support vector machines, in: ISSNIP 2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information, IEEE, 2007, pp. 335–340.
[28] M. Livani, M. Abadi, A PCA-based distributed approach for intrusion detection in wireless sensor networks, in: International Symposium on Computer Networks and Distributed Systems (CNDS), IEEE, 2011, pp. 55–60.
[29] S. Misra, K. Abraham, M. Obaidat, P. Krishna, Laid: a learning automata-based scheme for intrusion detection in wireless sensor networks, Security and Communication Networks 2 (2) (2008) 105–115.
[30] Y. Hu, A. Perrig, D. Johnson, Wormhole attacks in wireless networks, IEEE Journal on Selected Areas in Communications 24 (2) (2006) 370–380.
[31] W. Wang, B. Bhargava, Visualization of wormholes in sensor networks, in: Proceedings of the 3rd ACM Workshop on Wireless Security, ACM, 2004, pp. 51–60.

[32] G. Malkin, Traceroute Using an IP Option, RFC 1393 (Experimental), January 1993.
[33] K.C. Lee, R. Sudhaakar, L. Dai, S. Addepalli, M. Gerla, RPL under mobility, in: 2012 IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2012, pp. 300–304.
[34] Y. Chen, D. Lymberopoulos, J. Liu, B. Priyantha, Fm-based indoor localization, in: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, ACM, 2012, pp. 169–182.
[35] J. Newsome, E. Shi, D. Song, A. Perrig, The sybil attack in sensor networks: analysis & defenses, in: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, ACM, 2004, pp. 259–268.

Shahid Raza is a researcher at the Swedish Institute of Computer Science and a final year PhD student at the Mälardalen University, Västerås, Sweden. His main research interests are security issues in the IP-based wireless sensor networks and lightweight security solutions for the Internet of Things.

Linus Wallgren is a final year master student at the School of Computer Science and Communication at the Royal Institute of Technology in Stockholm, Sweden. He is currently a researcher at the Swedish Institute of Computer Science. His main research interests are computer security, embedded systems, autonomous systems and their real world applications.

Thiemo Voigt is a Professor of Computer Science at the Department of Information Technology and the VINN Excellence Centre for Wireless Sensor Networks at Uppsala University, Sweden. He is the leader of the Networked Embedded Systems Group in the Swedish Institute of Computer Science, Sweden. He received his Ph.D. in 2002 from Uppsala University, Sweden. His current research focuses on wireless sensor networks and system software for embedded networked devices and the Internet of Things.