Software-Defined IDS for Securing Embedded Mobile Devices

Richard Skowyra rskowyra@bu.edu
Sanaz Bahargam bahargam@bu.edu
Azer Bestavros best@bu.edu

Computer Science Department
Boston University

***Abstract*—The increasing deployment of networked mobile embedded devices leads to unique challenges communications security. This is especially true for embedded biomedical devices and robotic materials handling, in which subversion or denial of service could result in loss of human life and other catastrophic outcomes.**

In this paper we present the Learning Intrusion Detection System (L-IDS), a network security service for protecting embedded mobile devices within institutional boundaries, which can be deployed alongside existing security systems with no modifications to the embedded devices. L-IDS utilizes the OpenFlow Software-Defined Networking architecture, which allows it to both detect and respond to attacks as they happen.

I. Introduction

Mobile embedded devices are increasingly deployed in mission-critical applications ranging from biomonitoring and control in hospitals to robotic materials handling and transport in factories. These devices have unique security challenges: many are custom-built, low-power, resource-constrained devices lacking the capabilities of a full-fledged computer. Furthermore, successful subversion (or even partial service disruption) by an adversary can cause severe damage, ranging from financial loss to property damage and actual loss of life.

Much of the existing work on embedded system security aims to secure the devices themselves from physical subversion or reverse engineering, relying on anti-tamper techniques from cryptography and electrical engineering, among other fields. Many of these devices, however, communicate wirelessly with one another and with a larger on-site infrastructure network hosting command and control functions. These communications channels present another attack surface, and one which is difficult to secure using existing cryptographic techniques. Consider, for example, the unique access challenges associated with implanted biomedical devices (*e.g.* insulin pumps, pacemakers, *etc.*) in hospitals. These devices often communicate wirelessly with a wearable external monitoring and control unit, which must be accessible to an unpredictable set of emergency responders and medical personnel. This channel is difficult to secure with any key-based technique due to the substantial complexities related to key management and revocation, as well as the extremely limited power and heat-dissipation requirements of the device itself.

In this paper, we present a network-based intrusion detection system (IDS) for embedded mobile devices which communicates with an on-site infrastructure network. It can be deployed alongside any existing on-device security, with no modifications to the devices themselves. An intrusion detection system relies on a mathematical or profile-based model of expected network activity to detect anomalous traffic indicative of adversarial activity. Traditionally, IDS deployments have suffered from an inability to cope with end-host mobility, as well as a limited set of actions (*e.g.* logging the anomaly or alerting a system administrator) which can be taken in response to anomalies being detected. Both of these difficulties have limited the adoption of IDS outside of traditional enterprise networking environments.

Our Learning IDS (L-IDS) utilizes the OpenFlow Software-Defined Networking architecture to address both of these difficulties: mobility of embedded end-hosts can be transparently handled by the network, and a variety of rapid responses and network reconfigurations are available to handle the attack in realtime (*e.g.* flow dropping, reauthentication, honeypot redirection, and system isolation) rather than simply reacting after the fact.

We consider example cases in the domain of implanted biomedical devices and robotic transport. Both of these scenarios use a similar threat model: mobile embedded devices move throughout a building site, and communicate wirelessly with an infrastructure network. The adversary may be an insider who has subverted a device on the network (*e.g.* a webserver or printer), or may be external to the site but possesses a sufficiently powerful transmitter to communicate with the devices and infrastructure access points inside of the facility. For each case, we consider how a model of expected traffic patterns can be constructed, how adversarial actions can be characterized as anomalies in this model, and how the system can be programmed to react automatically to the detection of adversarial behavior.

The remainder of the paper is laid our as follows. In Section II we expand on the example cases of biomedical devices in a hospital and robotic materials transport on a factory floor and discuss related work. Section III discusses our threat model and anomalies, and Section IV presents the Learning IDS.

II. Motivation and Related Work

Biomedical Devices: Implanted Medical Devices (IMDs) for hospital patients are becoming increasingly popular, in part due to the accuracy and rapidity of measurement they make available to health professionals, as well as the precision control they offer with respect to automatic dosage and treatment. These devices range from emergency response systems such as implantable cardiac defibrillators (ICDs) [2] and bedside drug delivery systems to health maintenance devices such as pace makers, insulin pump infusion systems, implanted pulse generators, and implanted biosensors.

The security and privacy issues associated with IMDs have been explored from a hardware perspective in [16], [11], [19] and from a cryptographic perspective in [7], [29], [35]. The security concerns associated with IMDs are especially challenging, as incorrect operation due to adversarial interference can result in patient injury or death.

Consider, for example, hospital patients who suffer from Type-1 diabetes. Their blood glucose levels must be continuously balanced through controlled insulin injection and monitoring. Insulin pumps are often used to automate this process, via surgical implantation of a subcutaneous cannula connected to an external, wearable pump and insulin reservoir. This procedure also offers a much more responsive and fine-grained control loop for insulin delivery in response to expected or measured blood glucose levels [25]. Many insulin pumps also provide health professionals with remote control and communications functions, which can be used to integrate the pump with blood-glucose monitors or adjust insulin infusion schedules as determined by the patient's doctor.

Recent work has shown that these remote communications channels are vulnerable to attack by an adversary within wireless transmission range [4]. Note that this does not necessarily mean that the adversary needs to be physically close to the reciever: a strong antenna or subversion of a networked device (and therefore use of the building's own wireless access points) would suffice. Attacks that include both forging of messages from authorized senders and replaying of recorded messages are possible [4]. Either approach can be used to cause potentially fatal blood glucose levels. Injection of 100mg of insulin into a patient with normal blood sugar, for example, can induce a diabetic coma.

Cryptographic methods are a commonly suggested solution for securing IMD wireless communications channels. Unfortunately, these techniques are expensive in terms of processor usage and power consumption. Since the lifetime of IMDs is bounded by their battery life, cryptographic algorithms can noticeably reduce the useable lifetime of implanted devices. Once these batteries are depleted, surgery replacement is necessary.

Additionally, cryptographic methods introduce substantial challenges related to key management and dissemination. Patients may be treated at a variety of hospitals by a variety of personell, the identities of whom are frequently difficult to predict in advance. If an emergency room doctor has to administer an insulin treatment while a patient is travelling, for example, how can that responder learn keying material fast enough to not interfere with treatment, but securely enough that the key is not compromised?

We posit that an alternative, complementary approach to these techniques is deployment of in-network, off-device security systems designed to detect and respond to adversarial communications as they happen. The Learning Intrusion Detection System (L-IDS) presented in Section IV is one such technique.

Robotic Materials Transport: Mobile embedded devices are also utilized in robotic materials transport for control, sensing, and communications. These transport systems are deployed in three primary domains: military logistics [22], [27], [20], warehouse package picking [33], and robotic trucking through industrial/agricultural facilities [17], [32], [8].

A common feature of all of these devices is some form of remote operator control unit or link to a management system via wireless communication channels. Security of these channels is critical: subverted military units might be used for surveillance, destruction of property, and as improvised weapons. Industrial and agricultural systems are often designed to haul heavy loads and are consequently large, and could cause substantial damage to both the site and employees if diverted from their programmed course. Automated package pickers, if subverted, could halt warehouse operations or potentially exfiltrate goods by picking adversarially chosen packages.

Cryptographic techniques can be used to secure military communications channels, but may not be available (or feasible) on industrial or commercial systems designed to work in a trusted environment. An alternative approach, collectively called location security, relies on physics and computational geometry to confirm whether a transmission originates in the trusted zone bounded by the building site, or from outside of the site (and thus from an adversary) via a strong transmitter. These techniques utilize trilateration [21], signal strength analysis from multiple receptors [34], the differing speeds of radio waves and ultrasound in atmosphere [28], onboard radar [36], and Euclidean geometry of the physical network topology [12], among other physical indicators.

Unfortunately, location security cannot detect adversarial transmissions originating from inside of the trusted region, either from subverted mobile devices or from compromised network elements outside of the 'secured' region (*e.g.* network printers, webservers, *etc.*). Embedded devices also remain susceptible to jamming of the wireless channel, which could be damaging in the case of an ongoing sense or control stream between the mobile device and a human operator or management system. In any case, location security provides only a way to detect (and ignore) adversarial transmissions; there is no mechanism to react to the adversary or to prevent future transmissions.

The L-IDS system described below is a complementary technique that can be deployed alongside location security, and shore up the weaknesses of a purely location-based or cryptographic approach: adversarial behavior from within the trusted region (or jamming from outside) can be detected and the network can be dynamically reconfigured in realtime to respond to such attacks.

OpenFlow Software-Defined Networks: OpenFlow [23] is a popular architecture for software-defined networking. Network traffic is not required to conform to a particular network stack (*e.g.* the IP stack) and can be tailored to work with any protocol suite, including in-house or proprietary protocols. OpenFlow's fundamental unit of organization is the *network flow*, representing the sequence of packets sent from a source to a destination. Openflow switches route traffic by matching flows against an ordered sequence of rules, called a flow table. Each rule in the table has a pattern to match against a packet header and an associated set of actions to take on all packets which trigger a match. The OpenFlow specification currently defines 15 packet header fields which can be matched against, 11 of which can be used outside of an IP network [14]. Rules, at a minimum, can specify a port to route over, packet dropping, and forwarding of packets to the controller. Whenever a rule is triggered, the switch updates a set of traffic counters. These counters are disscussed more in Section IV, but allow for detailed measurement of network activity. If an incoming network packet's header does not match any flow rule, it is forwarded to the controller. Rules may also be set to expire after some time or duration, or may be removed by the controller.

The OpenFlow controller is a software process connected to each switch via a secure, dedicated link. The controller handles packets sent to it by OpenFlow switches and installs flow rules in the switches' flow tables. The functionality of the controller is determined completely by the application that it is being used to implement, but all controller programs must communicate with switches only by installing flow rules. Controllers can be written in a number of languages designed for the purpose. Popular choices include NOX/POX [18], Beacon [13], or Maestro [5], in addition to others [1].

Intrusion Detection for Embedded Devices: IDS have been developed for a variety of embedded platforms and applications. da Silva et al. [9] deploy an IDS on monitoring nodes in static wireless sensor networks, in order to detect both network-based attacks (packet replay, drop, *etc.*) as well as attacks on the routing structure and physical wireless channel. Banerjee [3] et al. also focus on WSNs, but distribute the monitoring task across existing sensors and use novel machine-learning techniques to compute an aggregate view of the network. Onat and Miri [26] exploit directional communication, static physical geometry, and other properties of sensor networks to detect node-impersonation and resource-depletion attacks. Mitchell and Chen [24] use behavior rules for tunable intrusion detection detection probabilities in stationary, wired medical cyber physical systems. Their threat model focuses on compromised nodes which can be used to forge, replay, drop, and modify messages. Darji and Trivedi [10] propose an IDS for wireless implanted medical devices which runs on a patient's smartphone and monitors the local wireless channel for activity not matching a programmed specification.

III. Threat Model

Setting: We consider security in the confines of a building site, base, or other location in which a wired networking infrastructure is deployed. Specifically, we model the facility as having a physical geometry representable as a map, with OpenFlow [23] wireless switches deployed at specific points on this map. They are connected to the larger infrastructure network, and are the primary point of communications used by the embedded mobile devices moving around the facility.

The embedded devices themselves transmit and receive information from the nearest wireless switch. We assume that the temporal and volumetric components of these transmissions, but not necessarily the content, can be modeled statistically. The embedded devices also move throughout the facility in (un)predictable patterns or courses. Even if mobility is unpredictable in terms of destination, we do assume that a physics model exists capturing simple physical limitations on speed and navigable geometry. This model can be used to determine the set of possible next locations of an embedded device given its current location and a time interval (*e.g.* it is probably impossible to move across the entire facility in one second).

In the context of the above scenario, our adversarial model considers the following malicious activities (or any combination thereof):

1) Transmissions may be made from outside the facility which are strong enough to be received anywhere within the facility.
2) The wireless channel may be degraded due to jamming.
3) A networked wireless end-host within the facility (*e.g.* a printer, desktop, *etc.*) may be subverted, and is within range of at least one wireless switch used by embedded mobile devices.
4) Via transmission from outside the facility, or via subverted hosts within it, the adversary may forge, record, and replay network packets.

Anomalies: An anomaly is characterized as a statistically significant deviation of a measured value (*e.g.* packets sent, position, time passed, *etc.*) from an expected value. We do not consider a specific anomaly detection algorithm in this work, as significant prior work has been conducted on these algorithms [15], [6] and L-IDS is modular enough to allow most anomaly detectors to be used without modification. All that we require of an anomaly detector is the ability to consume a measurement and expectation, and return the probability of that measurement being an anomaly.

Stateless Flow Anomalies are detectable based on pattern-matching against the headers of received packets. The L-IDS controller installs per-switch detection rules based on the current network configuration and the location of end-hosts. These rules can be used to, for example, limit the switches that specific hosts can communicate with or the protocols that they are allowed to use.

Stateful Flow Anomalies are detected by the controller via traffic counters collected by each switch. Anomalies of this form can, for example, correspond to violation of sense-actuate rules: a flow using a specific control protocol that originates at control server B and terminates at embedded device A should only be permitted when preceded within a time interval by a sensing protocol flow from A to B.

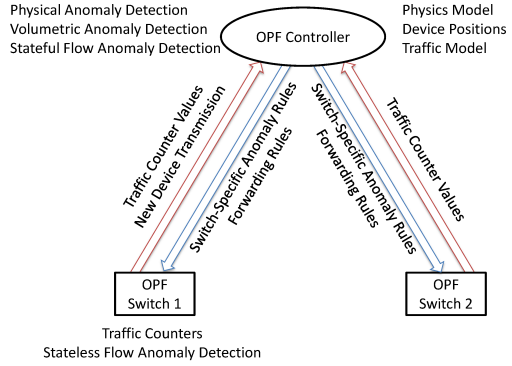

Fig. 1. L-IDS Controller and Switch Roles

```
1  if pkt.src_addr in device_list:
2   last_position[pkt.src_addr] = (src_sw, src_port)
3   for all switch in network:
4     if switch==src_sw:
5       routes[pkt.src_addr][switch]=src_port
6     else:
7       routes[pkt.scr_addr][switch]=routes[src_sw][switch]
8  if pkt.dst_addr in routes:
9    install_rule(src_sw, OUT, (dst_addr=pkt.dst_addr,
                routes[pkt.dst_addr][src_sw]))
10 else:
11   install_rule(src_sw, OUT, (dst_addr=pkt.dst_addr,
                 OPF_FLOOD))
```

Fig. 2. Learning Device Positions

Volumetric Anomalies are also detected via traffic counters, and correspond to the under- or overflows of traffic. Traffic volume can be defined over a time interval, over the output of a smoothing function, or any other computable metric. Volumetric anomalies can indicate jamming (underflow) or traffic injection (overflow) via replay attacks and forged packets, among others.

Physical Anomalies are violations of the physics model governing possible locations of mobile end-hosts given their prior location and the time since their last transmission. Anomalies of this sort can indicate that an adversary is forging transmissions (if end-hosts appear to be in multiple locations), have subverted a mobile embedded device and diverted from an expected course, or have disrupted or disabled end-host mobility.

IV. Learning IDS

The Learning-IDS (L-IDS) is an intrusion detection system for networks used to communicate with embedded mobile devices. Unlike traditional IDS deployments, L-IDS can transparently adapt to changing network state in the presence of end-host mobility. Furthermore, it can also respond to attacks (anomalies) as they happen and reconfigure the network to a heightened state of security. This contrasts sharply with traditional intrusion detection systems, which are very limited in both the time scale and flexibility of their response to an intrusion.

L-IDS relies on the OpenFlow [23] Software-Defined Networking architecture. The bulk of the IDS logic resides in the OpenFlow controller, but part of the anomaly detection process and all of the traffic measurement process resides in the network itself, on the OpenFlow switches. This division of labor is shown schematically in Figure IV. The controller in L-IDS serves three roles: detection of certain anomalies based on the evolution of traffic over time, adaptation of routing state and anomaly rules to end-host mobility, and network reconfiguration in response to intrusion. OpenFlow switches control dataplane routing, as well as detect anomalies based on matching of packet headers.

The L-IDS controller can accommodate any anomaly detection algorithm which can compare a measurement and statistical model, and return the probability of that measurement being an outlier given the model. A significant body of work already exists on creating anomaly detectors [15], [6]; these may be used with L-IDS or an application-specific detector can be developed.

Despite using an off-the-shelf anomaly detector, the mobility of end-host devices renders a simple comparison of measured values to a static model of expected network traffic, as would be normal in traditional IDS deployments, impossible. The mobility of end-hosts requires that all per-switch and per-flow traffic expectations must be adjusted to consider the current position of all mobile devices. The set of switches a device is permitted to send from (or the kind of traffic it is expected to originate), for example, may change as the device moves into different areas of a facility. Network routes must also be updated as the positions of end-hosts change.

A. Mobility and Routing

Routing correctness in the face of end-host mobility can be ensured by using a learning switch network, shown as pseudocode in Figure IV-A. We discuss the verification of these networks in detail in [31], but their basic operation is straightforward. Recall that OpenFlow switches forward any packet which does not match a flow rule to the controller. The controller then processes the packet using the logic in Figure IV-A. First, the source address is checked against a list of mobile devices. If the message originates from one of these, its position (*e.g.* the switch it is transmitting to) is updated (line 2) and routes to that device are changed to reflect its new position (lines 4-7).

Forwarding rules are then installed between the source and destination (lines 8-11). If the destination's last position is known, a route is installed in the OpenFlow switches on the path between each device. If the destination is unknown, a flooding rule is installed.

B. Threat Detection and Response

Intrusion detection in L-IDS is based on inputs from two sources: OpenFlow switch counters, and data plane packets forwarded to the controller. Switch counters are maintained for a variety of useful metrics, including per-flow duration and bytes transmitted/received, per-port errors, retransmits, and traffic volume, and per-flow table rule matches. These measurements are retained on-switch, and may be polled by

the controller using the `read-state` OpenFlow protocol message. Data plane packets are forwarded from a switch to the controller whenever no flow rule at that switch matches the header of a received packet. Note that flow rules may include wildcard characters, and may quantify over a variety of header fields ranging from hardware address of source and destination to VLAN tags and metadata. A complete list of both counters available and packet fields able to be matched over can be found in [14].

If an anomaly is detected, L-IDS can implement a wide variety of responses. Recall that the OpenFlow controller is a full-fledged software process running on commodity hardware: not only can the OpenFlow network be reconfigured, but out-of-band responses such as emailing/paging personnel, activating a physical alarm system, or interfacing with other control systems are all viable responses to a network intrusion.

Several off-the-shelf network reconfigurations have already been developed by the OpenFlow community. The FRESCO system [30] enables deployment of (among others) honeynets, in which an attacker is re-directed to a subnet devoted to distracting and studying the attacker; tarpits, designed to slow automatic worms by emulation of unused IP address/ports; and phantom networks which give attacks an incorrect view of network topology. In addition, simple flow rules can be used to lockdown or partition comprised network segments or to send emergency broadcasts. Note that many other responses are possible, and that all of these can be implemented by the controller automatically and in realtime.

C. Using L-IDS with IMDs

In this section, we provide an example illustrating how L-IDS can be used in a hospital setting to secure the communications between Implanted Medical Devices (IMDs) and doctors who are permitted to send actuation commands to them. Specifically, we consider the scenario in which a patient's blood glucose sensor sends updates to a doctor via the hospital network if measurements fall out of a safe boundary area. Depending on the value of these measurements, the doctor may send a response indicating an adjustment to the patient's basal rate, or specify an additional correction bolus.

Furthermore, we assume that a patient may be moved between a variety of locations in the hospital. These may be testing facilities (cardiology labs, medical imaging facilities), recreation or physical therapy areas, *etc.* This movement should not interfere with the communication between the patient's IMDs and the doctor monitoring them. There may also be areas of the hospital in which the patient should never be present; if IMD signals are sent from these areas, medical professionals should be alerted.

An adversary in this scenario could induce fatal blood glucose levels via two techniques: forging/replaying packets from the doctor, or forging/replaying packets from the patient. In the former case, the adversary could easily induce fatal blood glucose imbalances by specifying an extremely large correction bolus. In the latter case, false sensor data or sensor data recorded during an anomalous event and then replayed could be used to trick the doctor into specifying a fatal dose, given the patient's actual condition.

```
1  if (pkt.src_addr in p_dev && pkt.dst_addr in drs):
2   lst_upd[pkt.src_addr][pkt.dst_addr] = time()
3   install_rule(src_sw, OUT, (src_mac=pkt.src_addr,
                 switch_ports[src_sw][pkt.dst_addr])
4  if (pkt.src_addr in drs && pkt.dst_addr in p_dev &&
      ((time() -lst_upd[pkt.dst_addr][pkt.src_addr]) > n:
5   install_rule(src_sw, IN, (src_mac=pkt.src_addr,
                 switch_ports[src_sw][pkt.dst_addr])
6  else:
7    install_rule(src_sw, IN, (dst_mac=pkt.dst_addr,
                   expires=m, DROP))
8    alert_personnel()
```

Fig. 3. Paired Flow Rules for IMD Communication

```
1 if (src_sw in permitted_switches[pkt.src_addr]):
2     if physics_model(src_sw, pos[pkt.src_addr], time()):
3       last_position[pkt.src_addr]=(src_sw, time())
4       install_rule(src_sw, OUT, (src_mac=pkt.src_addr,
                 switch_ports[src_sw][pkt.dst_addr])
5  else:
6    install_rule(src_sw, OUT, (src_mac=pkt.src_addr, expires=m,
                 DROP))
7    alert_personnel()
```

Fig. 4. Anomalous Mobility Rules for IMD Communication

The former attack can be addressed using stateful flow anomaly detection rules, the pseudocode of which is illustrated in Figure IV-C. These are installed on the L-IDS controller. The default behavior of OpenFlow switch, upon receipt of a packet for which a forwarding rule does not already exist, is to forward it to the controller. In this case, the controller checks, on line 1, if the packet originates from a patient IMD and is being sent to a doctor's machine. If so, this transmission is noted and timestamped (line 2), and a forwarding rule to the destination machine is installed on the switch (line 3).

If the transmission is instead from a doctor's machine to a patient's IMD (line 4), a forwarding rule is installed only if an update was sent from the device to that machine within the time interval n. Otherwise an alert is triggered which blocks further remote control of the device for a time period m and contacts medical personnel out-of-band (via, e.g, triggering a patient's bedside alarm, paging a doctor, *etc.*). Alternative responses could also easily be implemented, including buffering of the message until authentication by the doctor, *etc.*

The above technique is not designed to be foolproof: adversarial transmissions originating in the allowed time interval may still be processed, and the alert procedure could be deliberately triggered to create a denial of service situation. L-IDS should not be seen as a complete security solution, but as a complementary technique which can be deployed alongside existing security mechanisms to create a defense in depth.

The latter attack, in which old or forged sensor data is sent to the doctor, can be addressed by exploiting known patient mobility using physical anomaly detection rules. Pseudocode for this is illustrated in Figure IV-C.

As in the previous example, this pseudocode is evaluated on packets sent from a switch for which a forwarding rule is not present. In this case, the controller checks to see if the patient

is transmitting from a switch which is in the permitted region of the hospital (line 1). If so, the physics model (comprising a facility map, maximum speed at which the patient may be moved, *etc.*) is checked to confirm that the patient's current position is within the set of reachable switches given the current time and time of its last transmission (line 2). If so, the device's position is updated (line 3) and a forwarding rule is installed (line 4).

If the device is transmitting from an un-permitted location, or from a location not reachable given its last position, future transmissions from that device are ignored and medical personnel are alerted.

Using this approach, the adversary would only be able to inject forged or old sensor data via switches that the patient's device could communicate with given its current position and set of permitted switches. Depending on the physical geometry of the hospital, this might be a substantial obstacle. Note that simply increasing transmission power may not work: if multiple switches register the transmission simultaneously, the physics model will almost certainly be violated.

D. Using L-IDS with Robotic Transport

In this example, we consider how L-IDS can be applied to robotic materials transport between locations at a manufacturing facility. Specifically, we consider a scenario in which robotic vehicles follow one of a series of paths through a facility (*e.g.* transporting precursor materials from the receiving depot to the manufacturing floor, from manufacturing to packaging, or from packaging to shipping). Which course a vehicle takes is assumed to follow a schedule which may vary be hour or day, but is known to the L-IDS controller.

While moving, vehicles are assumed to send a stream of sensor data to a remote operator or controller. For the purposes of the example we assume this stream is sent at a constant rate with equally sized packets, but another statistical model (*e.g.* exponentially distributed packet inter-arrival times) can also be used. The only requirement we impose on choice of model is that an expected traffic volume over time can be computed. The remote operator sends a stream of control data back to the vehicle, which also modeled statistically.

The adversary in this scenario is assumed to have subverted a stationary wireless device within the bounds of the facility (*e.g.* a network printer or workstation) which is within transmission range of at least one OpenFlow wireless switch used by the robotic vehicles. This adversary may attempt to jam the channel to degrade or disrupt communication, forge/replay sensor data from the vehicle, or forge/replay control data from the remote operator. Since the malicious transmitter is within facility itself, location security techniques which rely on the existence of a trusted geometric region [21], [34], [28], [12] cannot be used.

In order to address the possibilities of jamming or sending a stream of forged sensor data, the L-IDS controller can use OpenFlow switch-based counters to check for volumetric anomalies, as shown in Figure IV-D. The controller polls the switch regularly using the OpenFlow protocol's `read-state` functionality [14] (line 1).

```
1  counters=opf_read_state(switch)
2  anomaly,type=a_detector(counters.switch.bytes_received,
               expected_swich_volume[switch])
3  if anomaly > detection_threshold: alert(type)
4  for flow in counters.per_flow:
5    anomaly,type=a_detector(flow.bytes_received,
                 expected_flow_volume[flow.id])
6    if anomaly > detection_threshold: alert(type)
```

Fig. 5. Volumetric Detection Rules

As L-IDS does not rely on a particular anomaly detector, we abstract the choice into the `a_detector` procedure, which compares a measurement and statistical model, and returns the probability of that measurement being an outlier given the model (as in lines 2-4). The example pseudocode also assumes some information about the anomaly type (*e.g.* whether the measurement is anomalously large or small) is also available. Anomalously small received traffic volumes correspond to channel degradation and possible jamming, while anomalously large values indicate traffic injection. Note that these checks can be augmented with additional counter-derived measurements for more fine-grained detection, including dropped packets, transmit/receive errors, *etc.*

Attacks based on forged data from the remote operator can also be detected, as long as the switch is within range of the transmission from the adversary. Recall that all packets for which no forwarding rule exists are sent to the controller. If an adversary forges a message from the remote operator (or from the network switch, *etc.*) and transmits it to the mobile device, the switch will also receive it. OpenFlow switches distinguish flows based on both address and port pairs: a message arriving from a known address on an unexpected port (*e.g.* originating from the remote operator but arriving on the local wireless channel) will not match the flow rules already established for that address. The anomalous message will be forwarded to the controller, which can then trigger an alert. As this is a simple conditional check, pseudocode is omitted for brevity.

Finally, the positions of robotic vehicles can also be monitored for anomalous behavior. As long as their schedule and physical characteristics (speed, size, *etc.*) are known in advance, an identical approach to the IMD's anomalous mobility example can be taken.

V. Conclusion

In this paper, we demonstrated how an OpenFlow-based Software-Defined Network could be used to build a a Learning Intrusion Detection System (L-IDS) for embedded mobile devices within an institutional site. L-IDS can be deployed alongside existing security systems with no modifications to the embedded devices. A variety of attacks can be detected using this approach, and the network can be dynamically reconfigured in realtime to mitigate the attacks.

We provided example scenarios using implanted medical devices in a hospital and robotic material transport systems in a manufacturing facility. In each example, we considered a variety of possible attacks on the system, showed how anomaly rules can be created to detect the attack, and provided mechanisms to mitigate this attack or alert building personnel.

We are currently in the process of simulating the examples discussed in this paper. We intend to both study the performance of L-IDS and to formally verify critical safety properties of the network during reconfiguration in response to intrusion detection. We are also attempting to obtain traffic traces for representative real-world systems, in order to compare the performance of a variety of anomaly detection algorithms.

Acknowledgment: This work was supported in part by the following NSF grants: CISE/CNS Award #1239021, CISE/CNS Award #1012798, and ENG/EFRI Award #0735974.

REFERENCES

[1] Openflow components, 2011.

[2] A. H. Association. Implantable cardioverter defibrillator (icd).

[3] S. Banerjee, C. Grosan, and A. Abraham. Ideas: intrusion detection based on emotional ants for sensors. In *Intelligent Systems Design and Applications, 2005. ISDA '05. Proceedings. 5th International Conference on*, pages 344–349, 2005.

[4] Blackhat.com. Hack into diabet imd.

[5] Z. Cai, A. Cox, and T. Maestro. A system for scalable openflow control. Technical report, Technical Report TR10-08, Rice University, 2010.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[7] S. Cherukuri, K. K. Venkatasubramanian, S. K. S. Gupta, and E. K. S. Gupta. Biosec: A biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In *in Wireless Networks of Biosensors Implanted in the Human Body, Workshop on Wireless Security and Privacy (WiSPr), International Conference on Parallel Processing Workshops, 2003*, 2003.

[8] Cisco-Eagle. Seegrid vision-guided, unmanned robotic industrial trucks.

[9] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong. Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, Q2SWinet '05, pages 16–23, New York, NY, USA, 2005. ACM.

[10] M. Darji and B. Trivedi. Imd-ids a specification based intrusion detection system for wireless imds. *International Journal of Applied Information Systems*, 5(6):19–23, April 2013. Published by Foundation of Computer Science, New York, USA.

[11] T. Denning, K. Fu, and T. Kohno. Absence makes the heart grow fonder: new directions for implantable medical device security. In *Proceedings of the 3rd conference on Hot topics in security*, HOTSEC'08, pages 5:1–5:7, Berkeley, CA, USA, 2008. USENIX Association.

[12] E. Ekici, S. Vural, J. McNair, and D. Al-Abri. Secure probabilistic location verification in randomly deployed wireless sensor networks. *Ad Hoc Networks*, 6(2):195–209, 2008.

[13] D. Erickson. Beacon, 2012.

[14] O. N. Foundation. Openflow switch specification.

[15] P. Garcia-Teodoro, J. E. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

[16] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu. They can hear your heartbeats: non-invasive security for implantable medical devices. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pages 2–13, New York, NY, USA, 2011. ACM.

[17] T. Gordon. Autonomous haulage system (ahs).

[18] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[19] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy*, pages 129–142, 2008.

[20] A. Kerbrat. Autonomous platform demonstrator. Technical report, DTIC Document, 2010.

[21] S. Kraxberger, G. Lackner, and U. Payer. Wlan location determination without active client collaboration. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, pages 1188–1192, New York, NY, USA, 2010. ACM.

[22] L. Martin. Smss.

[23] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[24] R. Mitchell and I. Chen. Behavior rule based intrusion detection for supporting secure medical cyber physical systems. In *21th IEEE International Conference on Computer Communication Networks*, Munich, Germany, 08/2012 2012.

[25] NBCI. A review of the security of insulin pump infusion systems.

[26] I. Onat and A. Miri. An intrusion detection system for wireless sensor networks. In *WiMob (3)*, pages 253–259, 2005.

[27] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, et al. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control*, pages 10823–10825, 2008.

[28] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 1–10. ACM, 2003.

[29] S. Schechter. Security that is meant to be skin deep using ultraviolet micropigmentation to store emergency-access keys for implantable medical devices.

[30] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson. Fresco: Modular composable security services for software-defined networks. In *To appear in the ISOC Network and Distributed System Security Symposium*, 2013.

[31] R. Skowyra, A. Lapets, A. Bestavros, and A. Kfoury. Verifiably-Safe Software-Defined Networks for CPS. In *Proceedings of the 2nd ACM International Conference on High Confidence Networked Systems (HiCoNS 2013)*, Philedelphia, PA, USA, April 2013.

[32] A. T. Stentz, C. Dima, C. Wellington, H. Herman, and D. Stager. A system for semi-autonomous tractor operations. *Autonomous Robots*, 13(1):87–103, July 2002.

[33] K. Systems. Automated materials handling systems.

[34] A. Vora and M. Nesterenko. Secure location verification using radio broadcast. *Dependable and Secure Computing, IEEE Transactions on*, 3(4):377–385, 2006.

[35] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li. Imdguard: Securing implantable medical devices with the external wearable guardian. In *INFOCOM*, pages 1862–1870, 2011.

[36] G. Yan, S. Olariu, and M. C. Weigle. Providing vanet security through active position detection. *Computer Communications*, 31(12):2883–2897, 2008.