Low-Cost Flow-Based Security Solutions for Smart-Home IoT Devices

Arunan Sivanathan*, Daniel Sherratt*, Hassan Habibi Gharakheili*, Vijay Sivaraman* and Arun Vishwanath†
*Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia.
†IBM Research, Melbourne, Australia.
Emails: {a.sivanathan, d.sherratt}@student.unsw.edu.au, {h.habibi, vijay}@unsw.edu.au, arvishwa@au1.ibm.com

***Abstract*—The rapid growth of Internet-of-Things (IoT) devices, such as smart-bulbs, smoke-alarms, webcams, and health-monitoring devices, is accompanied by escalating threats of attacks that can seriously compromise household and personal safety. Recent works have advocated the use of network-level solutions to detect and prevent attacks on smart-home IoT devices. In this paper we undertake a deeper exploration of network-level security solutions for IoT, by comparing flow-based monitoring with packet-based monitoring approaches. We conduct experiments with real attacks on real IoT devices to validate our flow-based security solution, and use the collected traces as input to simulations to compare its processing performance against a packet-based solution. Our results show that flow-based monitoring can achieve most of the security benefits of packet-based monitoring, but at dramatically reduced processing costs. Our study informs the design of future smart-home network-level security solutions.**

I. Introduction

The rapid proliferation of Internet of Things (IoT) devices is giving rise to "smart-homes" that can be monitored and controlled remotely over the Internet. Consumers can now control their lighting systems from anywhere [1], receive alerts on their mobile phone when smoke is detected in the house [2], monitor their kids from afar [3], and turn appliances on/off while driving to/from work [4]. Recent surveys in the US have shown that many consumers are willing to pay in excess of $500 for a fully-equipped smart-home, citing personal or family safety, property protection, lighting/energy management, and pet monitoring as top motivators [5].

The increasing prevalence of smart-homes with Internet-connected devices creates security concerns at unprecedented levels. An eavesdropper can illegitimately snoop into family activities, and a malicious entity can take control of the home to either harm the household or to use it as a launch-pad for attacks into other domains. The vulnerabilities of existing IoT devices have been documented by several earlier studies [6], [7], and there is evidence of anecdotal [8] as well as large-scale attacks [9] on IoT devices.

Securing IoT devices from attacks remains a formidable challenge. The large heterogeneity in IoT devices, each with its own hardware, firmware, and software, makes the security vulnerabilities diverse and the attack vectors manifold. Worse, device manufacturers have been lax in embedding security in consumer IoT devices, dissuaded by low margins, time-to-market pressures, and limited resources. One could argue that the home router, by virtue of its NAT/firewall functionality, provides an effective protection against external attacks by dropping unsolicited Internet traffic directed to household IoT devices. However, our recent work in [10] has shown that even this perimeter defense can be bypassed via malware embedded into mobile Apps; such malware can scout the internal network for vulnerable IoT devices, and disable the home firewall to allow Internet attacks on such devices.

We believe that a promising approach to the above problem is to embed security solutions at the network-level, whereby network traffic to/from IoT devices is monitored to detect (and block) attacks, much the way today's intrusion detection systems (IDS) monitor enterprise network traffic for security threats. Indeed, our prior work [11] has demonstrated the utility of traffic-flow monitoring in securing access to devices such as light-bulbs in the smart-home; concurrently, work in [12] has developed a method for specifying IoT security policies, that are then applied to the network data-plane traffic via specialized middle-boxes (termed *μmbox*es).

While the above methods show promise in securing smart-homes at the network-level, it remains unclear what their cost/benefit trade-offs are, particularly since inspection of network traffic can involve significant costs that may be difficult to recover in a residential market wherein profit margins are low. In this paper, we undertake an evaluation of the cost-benefit trade-offs of a flow-level approach (that predominantly uses information about active flows in the network) against a packet-level approach (that looks into the content of the data packets) for securing smart-home IoT devices. In this context, our contributions are:

- We develop an architecture and method for flow-level characterization of IoT traffic, that can effectively detect malicious IoT activity while minimizing the need to inspect content of IoT data packets;
- We validate our approach experimentally by launching internal and external attacks on real IoT devices, and analyzing the resulting traffic traces; and
- We evaluate our approach to show that our flow-based technique can achieve comparable security performance to packet-level inspection techniques, but at dramatically reduced processing cost.

The rest of this paper is organized as follows: §II describes prior work on IoT security solutions, and §III describes our solution approach that captures and evaluates flow-level information. In §IV we describe our experimental setup including attack emulation and trace collection, used to validate our solution, while in §V we evaluate the cost-benefit trade-offs via simulation. The paper is concluded in §VI.

II. Background and Prior Work

Prior work on IoT security can be characterized as *host-based* or *network-based*. *Host-based* schemes embed security in the IoT device itself, either adapting existing security mechanisms, or developing new ones, to suit the resource constraints of IoT devices. For example, the work in [13] explores the use of popular IP-based network management protocols for IoT, and shows that session key generation can be very costly, making SNMP more suitable in terms of resource usage than NETCONF. An architecture for secure end-to-end communication for IoT is proposed in [14], that moves the expensive authentication and encryption operations out of the IoT device into an external entity with abundant resources. A similar approach is taken by [15], which points out that the DTLS (Datagram Transport Layer Security) handshake requires significant resource requirements when employing public-key cryptography for peer authentication and key agreement – by offloading the DTLS connection establishment to a non-resource constrained delegation server, the authors show that memory and computation overhead can be reduced by 64% and 97% respectively.

A lightweight secure protocol for IoT devices called Lithe is proposed in [16]. Resource constrained devices are expected to use CoAP (Constrained Application Protocol) at the application layer, and DTLS (Datagram Transport Layer Security) at the transport layer for secure communication. However, since DTLS is not well suited for use in resource constrained devices, Lithe proposes a novel method to compress the DTLS protocol using the 6LoWPAN header compression mechanisms. Experiments demonstrate the suitability of Lithe for application in IoT devices. The work in [17] identifies challenges in the handshake phase of HIP DEX (Host Identity Protocol Diet EXchange), another IP security protocol for use in constrained IoT devices, and proposes lightweight extensions to it. The standardization effort underway at the Internet Engineering Task Force (IETF) for securing IoT devices using CoAP and DTLS is described in [18]. The work in [19] gives a perspective of how concepts from information centric networking (ICN), which is still in its infancy, can be used in IoT security. A comprehensive survey of security solutions in IoT with a focus on key management, authentication, and confidentiality can be found in [20].

The host-based IoT security solutions above not only have to contend with the constrained resources on IoT devices (processing, memory, battery, and radio), but also the lack of motivation amongst vendors to implement these schemes. In their rush to market, vendors often do not have the time, skills, resources, or financial incentives to embed security in their products. This has motivated recent proposals to develop *network-based* security solutions [12] that are better suited to the scale of deployment (i.e. billions of devices), nature of communication (device-to-device than human initiated), diverse use cases (e.g. motion sensory triggering an alarm, temperature sensor opening the windows, etc.), and interoperability constraints (devices from different vendors unable to communicate with each other). Our prior work in [11] also explores the potential of network-level security for specific IoT devices like the Phillips Hue light-bulb and Nest smoke-alarm. While both proposals advocate the use of Software Defined Networking (SDN) in the control plane, [12] makes use of specialized micro-middleboxes (μ*mbox*es) in the data plane to inspect IoT packets. Among the objectives of the present work is to compare the efficacy and cost of flow-based versus packet-based security solutions, as described next.

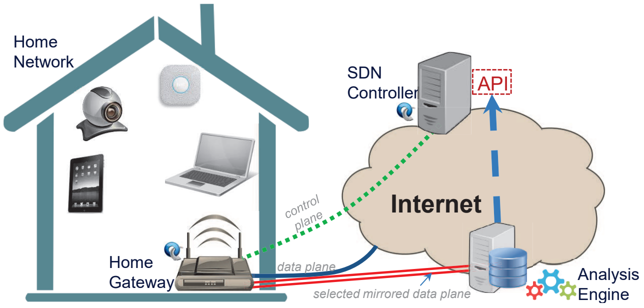


Fig. 1. System architecture showing data and control planes

III. Our Flow-Level Security Solution

We outline our solution architecture that operates at the flow-level method to detect intrusions on IoT devices in the home network. The aim is to discover attack patterns or suspicious network activity inside the home at low cost and in a programmatic way, so that the network resources are used as efficiently as possible for protecting IoT devices against security attacks. Unlike other proposals that require the use of deep packet inspection (DPI) or other virtualized network functions (NFV) for detecting threats, we advocate the use of dynamic characterization of the traffic at the flow-level. This requires us to inspect only a tiny fraction of data-plane traffic, thereby limiting the processing cost and network bandwidth overheads. The type of flows that need to be inspected are chosen dynamically and can change according to the vulnerabilities. Lastly, we manage and process flows from cloud-based software, instead of embedding the processing unit into the home gateway that is difficult to maintain and upgrade.

A. Operational Scenario

Fig. 1 shows our system architecture. Each residence has a home gateway to which household devices connect. The home gateway offers Internet connectivity via a (DSL, Cable, or PON) broadband link. The home gateway is SDN-enabled and managed by a controller hosted on the cloud. We propose an "analysis engine" that interacts with the SDN controller via northbound APIs. It issues requests to the SDN controller on which flows are inspected. The controller then programs the home gateway with rule(s) to mirror selected traffic flows toward the analysis engine. Therefore, the analysis engine will be able to actively monitor the network activity of IoT devices by inspecting few packets to/from IoT devices with specific headers as well as measuring the load of selected flows. Whenever traffic analysis is concluded, then traffic mirroring can be stopped by deleting the pertinent rule(s) inside the home gateway.

B. The Analysis Engine

When the mirrored traffic comes in to the analysis engine operating in the cloud, an algorithm is run to inspect the flow, e.g. recording source and destination entities of certain requests and responses. If needed, the analysis engine requests the controller to install rule(s) corresponding to IoT devices that are accessed from the Internet. This may involve setting up subsequent rules ensuring not all data-plane traffic is forwarded to the analysis engine. In what follows we describe the rules in more detail and elaborate on their specific match and action fields.

C. The Network Rules

A home network is protected by NAT service from potential Internet-based attackers. However, client devices can be exposed to Internet attacks by abusing the Universal Plug-n-Play (UPnP) port forwarding capability on a typical home gateway. We have shown in [10] that a malware application running on an unsuspecting user's mobile device can discover IoT devices within the home by using a standard Simple Service Discovery Protocol (SSDP); this can be followed by a UPnP port forwarding command to the home router that allows an external attacker to directly attack the IoT device. We note that SSDP and UPnP port forwarding messages are common in a typical home network environment for various peer-to-peer applications (e.g. Skype) and multi-player games.

Since protocol-specific traffic is characterized by known packet headers, we propose to push rules into the home gateway to capture certain traffic types and and forward them to the analysis engine. Note that these rules will ensure normal forwarding of traffic, along with sending a "mirror" copy to the analysis engine. This allows the home gateway to provide standard data-plane forwarding without being affected by the intrusion detection process. By receiving flow-level data, the analysis engine will gain flow-level visibility into clients (i.e. IP and MAC addresses) and their network activity within the home network. For example, SSDP uses UDP transport protocol on port 1900. Thus, having rule entries that match UDP source/destination port 1900 would capture all SSDP requests and responses transferred in the home network. On the other hand, a port forwarding request is communicated with the home gateway using HTTP post mechanism. It consists of a sequence of data exchanges that can be characterized by a network rule which matches the destination IP address of the home gateway.

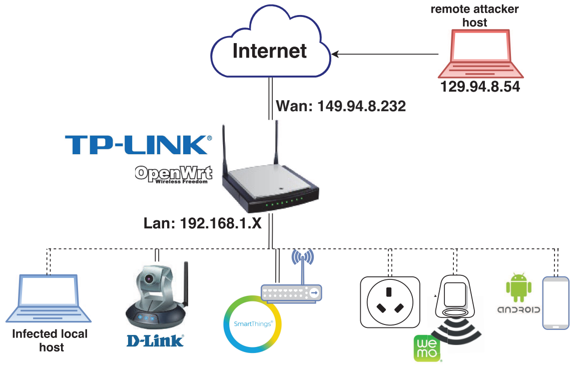


Fig. 2. Testbed showing IoT devices, SDN home gateway, and external attacker

Detection of SSDP and port forwarding messages are not sufficient to discover suspicious activity or an attack. We propose to monitor Internet traffic flows to/from an IoT device to create a richer context around access of a specific device. The remote access traffic of an IoT device inside the home network can be identified by a rule that matches the source MAC of the default gateway as well as the destination MAC address of the device for downstream traffic from the Internet (and vice versa for the upstream direction towards the Internet). Once the analysis engine receives the first packet of such flow and extracts the IP address of the remote host, it instructs the controller to install a pair of new higher-priority rules into the home gateway that match source/destination IP of remote host and destination/source IP of the IoT device with action "normal" only (i.e. no more packets for this flow are forwarded to the analysis engine). This allows us to reduce the cost of our inspection process.

IV. Experimental Setup and Validation

Setup: We built a small testbed in our lab, depicted in Fig. 2, to emulate a typical home network. We used a standard TP-Link WR1043ND home gateway and flashed it with OpenWrt firmware version 15.05. We also installed additional OpenWrt packages such as `tcpdump` for capturing traffic, `bash` for scripting, and `miniUpnp` for managing the home gateway. We connected the WAN interface of the home gateway to our university campus Internet while the IoT devices were attached to the LAN/WLAN interfaces. The IoT devices connected to the gateway include Samsung's Smart Things, D-Link DCS-5300G IP security camera, Belkin's Wemo switch and Wemo Motion detector. The Samsung Smart Things kit includes a collection of IoT devices that connect to the router through an internal hub. These IoT devices communicate with a mobile App when an activity is triggered such as motion detection or a clip is removed. The D-Link DCS-5300G is a standard IP surveillance camera that a user can control it (i.e. pan, tilt or move the preset position) and access its video/audio stream over the network. Belkin's Wemo switch is a smart switch that can be turned on/off using pertinent mobile App while Belkin's motion detector uses an App to alert the user when someone has moved near the device. A laptop and an Android phone were also present in the network to communicate with the IoT devices and generate user traffic on the network respectively. To collect traffic flowing through the home gateway, we used `tcpdump` application that continuously recorded all traffic of all interfaces. We then stored the collected data into a `pcap` file on an external hard drive attached to the home gateway via its USB port.

Traffic: The aim is to experiment on a smart-home environment comprising real IoT devices and validate our approach by launching internal and external attacks to IoT devices. Therefore, we recorded all network traffic over an 18 minute (1080 second) period. During this period all the devices on the network generated their usual traffic. This included the application on the smart phone communicating with the Samsung SmartThing hub when the multipurpose sensor was triggered. Similarly the Belkin Wemo Motion detector was continuously transmitting data to the smart phone when it detected movement. The IP camera was also accessed, and a live video stream was watched on the laptop, all of this was occurring on the internal network. Moreover, the android phone and laptop were accessing popular Internet content such as Youtube and Facebook to emulate user traffic on the network. All of this traffic would signify everyday network usage in a typical home environment.

Attack: While the traffic was being captured, an attack was launched from a script running on the laptop. We assume that the laptop was "infected" by malicious code (i.e our script), and carried out an attack by enabling port forwarding for selected IoT devices. An infection can occur if a user unintentionally executes a malicious application. In our case, we manually ran two python scripts to emulate this attack. This device would already be authorized to connect to the internal network and would be connected inside of the NAT. The infected host would then instigate the attack in three different stages: (a) Discovery, (b) Port Forwarding and (c) Access.

SSDP scan: The first python script (i.e. "discovery.py") performed an SSDP broadcast. The script sends a `msearch` multicast packet to `239.255.250:1900`. This is an assigned IP address set by IANA for the UPnP protocol. The IoT devices that implement UPnP would respond with basic information about themselves and a URL which contains an XML file that describes the device functionalities. In this XML file, there is a control URL which is then used to trigger an action in the device via a HTTP POST request. For the D-Link IP camera, this URL is the user/browser presentation (192.168.1.124:5004) and for the Belkin Wemo switch it is the URL which manipulates the switch state (192.168.1.223:49153/upnp/control/basicevent1). If the home gateway has UPnP enabled, then its control URL is contained under the `WANIPConnection` service. One of the services that WANIPConnection provides is to set up a new port mapping between an internal IP address/port to an external port.

Port-forwarding: Setting up the port forwarding was the next step in our attack. Our script (i.e "port-forward.py") was run which maps an external port to the control URL of the IoT device, allowing an external user to have direct access the internal device. The script created a specific SOAP-based HTTP POST request and sent it to the WANIPConnection control URL of the home gateway which was discovered earlier by the first python script. The SOAP Envelope is depicted in Fig. 3. It can be seen that the IoT device (i.e. `$Internal-Port` and `$Internal-IP`) is exposed to an external attacker via a direct access to port `$External-Port` from the Internet. Accessing the device is the final stage of the emulated attack.

```
<?xml version="1.0"?>
<s:Envelope
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 s:encodingStyle="httpL//schemas.xmlsoap.org/soap/encoding">
    <s:Body>
        <u:AddPortMapping xmlns:u="urn:schemas-upnp-
org:service:WANIPConnection:1">
            <NewRemoteHost></NewRemoteHost>
            <NewExternalPort>$External_Port</NewExternalPort>
            <NewProtocol>TCP</NewProtocol>
            <NewInternalPort>$Internal_Port</NewInternalPort>
            <NewInternalClient>$Internal_IP</NewInternalClient>
            <NewEnabled>1</NewEnabled>
            <NewPortMappingDescription>Description</NewPortMappingDescription>
            <NewLeaseDuration>0</NewLeaseDuration>
        </u:AddPortMapping>
    </s:Body>
</s:Envelope>
```

Fig. 3. SOAP message

Attack detection: After conducting SSDP discovery and port forwarding from an internal infected host, we then used an external host (public IP address 129.94.8.54) to launch the attack. Given the information collected in previous stages, the Wemo Switch and IP Camera were successfully accessed from the external host. This attack has many implications; we can imagine if the SmartThings multipurpose sensor was on a door, acting as a security alert when the door was opened. However, the SmartThings hub was powered through the Wemo Switch. An external attacker would be able to use the camera and see whether the Smart Home was occupied. If the attacker saw that no-one was home, they could turn off the Wemo Switch, disabling the SmartThings security, thereby allowing the attacker physical access to the home. In this case the added layer of physical security has been bypassed due to insecurity in the network layer. The trace file above enacts this scenario. Using our flow-based analysis, the sequence of activity (SSDP scan followed by port-forwarding command followed by external access) is logged, and the potential malicious activity is flagged.

V. Evaluation via Simulation

We now evaluate the computational cost of our solution by applying it to real trace data obtained from our testbed. Our `pcap` trace file of size 327 MegaBytes covers a 1080 second period and comprises 334,088 packets. A time trace of the traffic load is shown in Fig. 4. We note that the average load on the network is 301 Kbps, however it is seen that the load sometimes spikes to over 1 Mbps due to the buffering behavior of Youtube traffic.

We wrote a native simulation in C that takes packet arrivals from the trace as input, and performs discrete event simulation on an SDN-based home gateway along with a controller and an analysis engine. The simulator manages a table of rules (i.e. a flow table) inside the home gateway according to the events

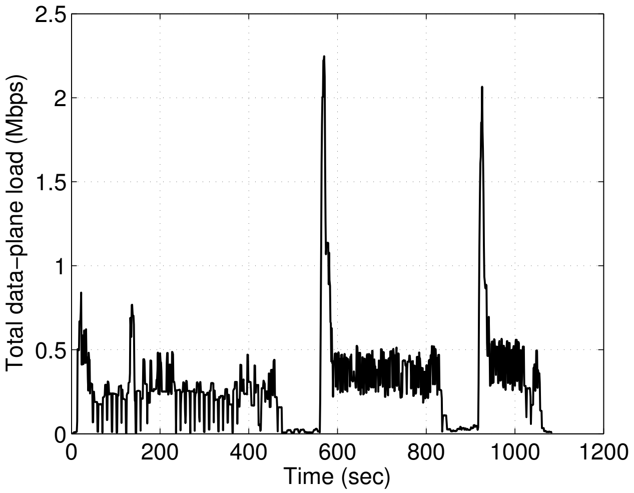


Fig. 4. Total captured load of data-plane

that occur at run time and keeps track of certain events (e.g. detection of port forwarding attack) and performance metrics (e.g. mirrored load).

As explained earlier in III-C, the flow table of the home gateway is initialized by the controller module of our simulator to contain proactive rules that capture SSDP and port forwarding traffic.

SSDP specific rules: In our simulation, the following proactive rules capture SSDP requests and responses:

- SSDP-Request: *Priority*:1, *Match*:`"udp_dst=1900"`, *Action*: `"normal,tunnel"`
- SSDP-Response: *Priority*:1, *Match*:`"udp_src=1900"`, *Action*: `"normal,tunnel"`

In real practice, a “tunnel” interface on the home gateway is needed to feed the remote analysis engine on the cloud. Therefore, in our simulation, a packet is deemed to be received by the analysis engine module only if the packet matches a rule that has a `"tunnel"` interface in the action field.

UPnP Port forwarding rules: The DHCP server of our testbed provides local clients with IP addresses in the range of 192.168.1.100-254/24 with a default gateway of 192.168.1.1. Therefore, the following proactive rule captures all traffic sent directly to the local default gateway:

- *Priority*:1, *Match*:`"ipv4_dst=192.168.1.1"`, *Action*: `"normal,tunnel"`

In the analysis engine we use a regular expression pattern match to identify the port forwarding request and extract the internal IP and port number. If a host enables a port forwarding for an IoT device, the analysis engine flags it as suspicious activity that is trying to create a back-door to access the IoT device.

Remote access rules: Upon processing SSDP messages, the IP and MAC addresses of IoT devices inside the home are discovered by the analysis engine at run time. Therefore, the following reactive rules are installed to capture intial Internet traffic from/to each IoT device in order to detect the flows:

- to-IoT: *Priority*:10, *Match*:`"eth_dst=mac-iot, eth_src=mac-gw"`, *Action*: `"normal,tunnel"`
- from-IoT: *Priority*:10, *Match*:`"eth_src=mac-iot, eth_dst=mac-gw"`, *Action*: `"normal,tunnel"`

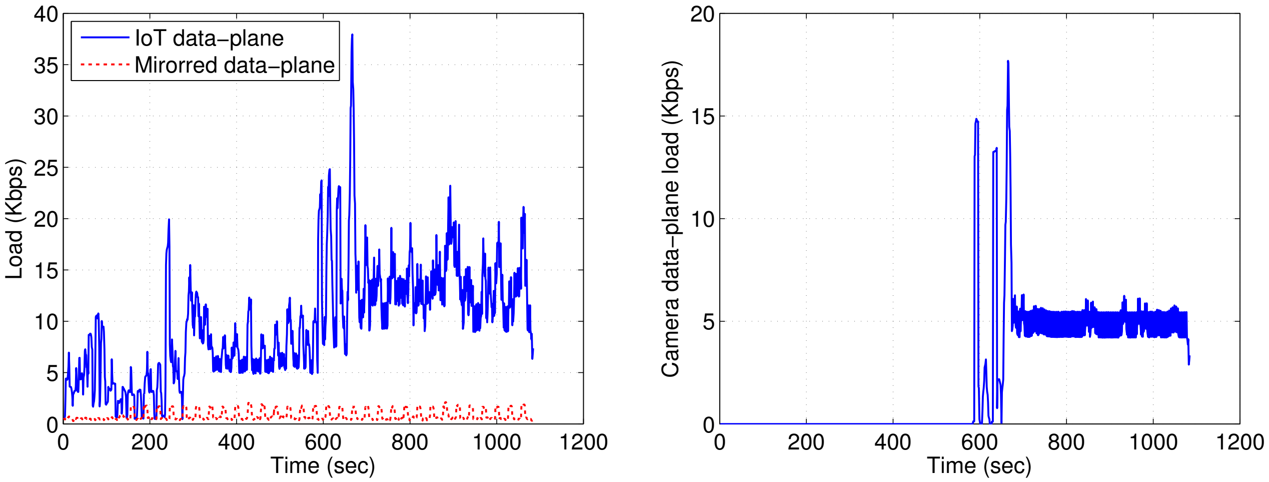
We assume that the MAC address of the default gateway (i.e. `"mac-gw"`) is a known parameter by the controller.

In order to limit the processing cost of mirroring IoT traffic (specifically for a device like a camera that is sending a continuous video stream), once the first few packets of a flow are detected and processed by the analysis engine, the flow-level details (such as IP address of the remote host) are recorded. The analysis engine thereafter requests the controller to install another rule with a higher priority, the match field is updated with the IP address of the remote host and the action field is set as `"normal"` so as to stop mirroring traffic pertaining to this flow:

- to: *Priority*:100, *Match*:`"eth_dst=mac-iot, ipv4_src=remote-ip"`, *Action*: `"normal"`
- from: *Priority*:100, *Match*:`"eth_src=mac-iot, ipv4_dst=remote-ip"`, *Action*: `"normal"`

With the above rule inside the home gateway, the analysis engine is able to track the flow activity by periodic reading of counters via the controller to maintaining real-time statistics. Therefore, the load or the volume of the flow can provide the analysis engine with more information about such remote access into an IoT device.

Results: Our simulation keeps track of performance metrics such as the size of the flow table, total network load, and volume of mirrored traffic. We ran our simulation using the collected traces from the previous section as input, and make the following observations. Our analysis engine was able to detect all SSDP and port forwarding messages exchanged over the emulated home network, by virtue of the pro-active SDN rules installed in the home gateway. In Fig. 5(a) we show the processing cost of our solution: the solid blue line shows the total data-plane load arising to/from IoT devices, averaging around 9.65 Kbps, and is proportional to the processing cost of a packet-based solution that inspects all data-plane IoT traffic. By contrast, the dashed red line in the figure shows the volume of traffic that is forwarded to the analysis engine in our flow-based approach – the average load of mirrored traffic is only 0.84 Kbps, which is roughly an order of magnitude lower than required by the packet-based monitoring approach. This is not surprising, because our approach only needs the first few packets of each flow to be sent to the analysis engine, which then inserts a reactive flow-entry to stop the packet mirroring, and uses packet/byte-counts thereafter to monitor flow activity without inspecting packet contents. This dramatically reduces processing costs; indeed Fig. 5(b) shows the traffic load pertaining to the IP camera that was accessed from a remote host during our simulation – in this case forwarding the video packets to the analysis engine is wasteful, and our method only logs the flow activity rather than the actual video content. In Fig. 6 we show the volume of IoT data plane traffic over the 18-minute duration; of the total IoT data



(a) IoT traffic load

(b) IP-camera traffic load

Fig. 5. Processing cost: (a) IoT data-plane load and mirrored traffic load; and (b) Traffic load of IP camera

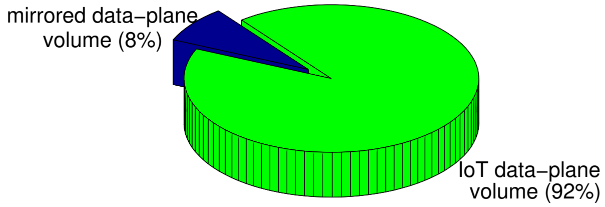


Fig. 6. Pie chart of total and mirrored traffic volume

plane traffic volume of $10,454$KB, only 908KB, corresponding to 8% of the total IoT traffic volume, needs to be forwarded to the analysis engine in our approach, making the processing cost acceptable for home users. Lastly, we would like to point that our method does not inspect packet contents, and would therefore not detect attacks (such as use of default plain-text passwords) that require examining the data inside packets.

VI. Conclusion

Securing IoT devices in the emerging smart-home is a critical yet challenging problem. This paper has examined the use of network-level monitoring to detect and mitigate IoT security threats. Specifically, we have investigated the use of SDN to monitor network traffic at flow-level granularity. We have shown that this is effective in detecting security threats, without incurring the high costs of packet-level monitoring of traffic to/from IoT devices. We have validated our solution in an experimental test-bed with real IoT devices. We have also evaluated via simulation of gathered traffic traces the processing overheads of our solution, and found them to be low is cost. We hope that our work will spur more research into the use of flow-level monitoring to mitigate security threats on the future smart-home.

References

[1] Phillips. Hue Personal Wireless Lighting. http://www2.meethue.com/.

[2] Nest. Nest Smoke Alarm. https://nest.com/.

[3] Withings. Withings Smart Baby Monitor. http://www.withings.com/us/en/products/baby/smart-baby-monitor?

[4] Belkin. WeMo Switch. http://www.belkin.com/au/p/P-F7C027/.

[5] iControl. (2014) 2014 State of the Smart Home. http://www.icontrol.com/docs/pdf/2014_State_of_the_Smart_Home_-_Final.pdf.

[6] HP Enterprise. (2015) Internet of Things Research Study. http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf.

[7] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, “An Experimental Study of Security and Privacy Risks with Emerging Household Appliances,” in *Proc. First International Workshop on Security and Privacy in Machine-to-Machine Communications (M2MSec)*, Oct 2014.

[8] abcNEWS. (2013) Baby Monitor Hacking Alarms Houston Parents. http://abcnews.go.com/blogs/headlines/2013/08/baby-monitor-hacking-alarms-houston-parents/.

[9] Business-Insider. (2014) Refrigerator Hacked. http://www.businessinsider.com.au/hackers-use-a-refridgerator-to-attack-businesses-2014-1?op=1.

[10] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, “Smart-Phones Attacking Smart-Homes,” in *Proc. ACM WiSec*, Jul 2016.

[11] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, “Network-Level Security and Privacy Control for Smart-Home IoT Devices,” in *Proc. IEEE WiMoB Workshop on Internet of Things Communications and Technologies (IoT-CT)*, Oct 2015.

[12] T. Yu, V. Sekar, S. Sheshan, Y. Agarwal, and C. Xu, “Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things,” in *Proc. ACM HotNets*, Nov 2015.

[13] A. Sehgal, V. Perelman, S. Kuryla, and J. Schnwlder, “Management of Resource Constrained Devices in the Internet of Things,” *IEEE Communications Magazine*, pp. 144–149, Dec 2012.

[14] R. B. et al., “Secure Communication for Smart IoT Objects: Protocol Stacks, Use Cases and Practical Examples,” in *Proc. IEEE WoWMoM*, USA, Jun 2012.

[15] R. Hummen, H. Shafagh, S. Raza, T. Voigt, and K. Wehrle, “Delegation-based Authentication and Authorization for the IP-based Internet of Things,” in *Proc. IEEE SECON*, Singapore, Jun/Jul 2014.

[16] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, “Lithe: Lightweight Secure CoAP for the Internet of Things,” *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, 2013.

[17] R. Hummen, H. Wirtz, J. H. Ziegeldorf, J. Hiller, and K. Wehrle, “Tailoring End-to-End IP Security Protocols to the Internet of Things,” in *Proc. IEEE ICNP*, Germany, Oct 2013.

[18] S. L. Leoh, S. S. Kumar, and H. Tschofenig, “Securing the Internet of Things: A Standardization Perspective,” *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, 2014.

[19] M. A. et al., “Information-Centric Networking for the Internet of Things: Challenges and Opportunities,” *IEEE Network*, vol. 30, no. 2, pp. 92–10, 2016.

[20] ——, “M2M Security: Challenges and Solutions,” *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1241–1254, Q2, 2016.