Offloading Smartphone Firewalling Using OpenFlow-capable Wireless Access Points

Daisuke Miyamoto*, Ryo Nakamura*, Takeshi Takahashi†, Yuji Sekiya*
* The University of Tokyo
2-11-16 Yayoi, Bunkyo-ku, Tokyo 113-8658, JAPAN
daisu-mi@nc.u-tokyo.ac.jp, upa@wide.ad.jp, sekiya@wide.ad.jp
† National Institute of Information and Communications Technology
4-2-1 Nukuikitamachi, Koganei, Tokyo, 184-8795, JAPAN
takeshi_takahashi@nict.go.jp

***Abstract*—This paper proposes new firewall for protecting smartphone from cyber threats. The key idea is to offload firewall functions to OpenFlow-capable wireless access points (APs). The widespread use of smartphones requires protection against cyber threats targeted to the device. Our research group therefore explored the suitable protection schemes and found that the OpenFlow-capable APs are able to facilitate configuring filtering rules, and also make defense at the closest point to the issued device, as well as saving energy consumption whereas firewall applications work on smartphone heavily drain battery life. In this paper, we design and implement our proto-type implementation along with our consideration, show information pipelining in order to provide cyber defense from threat information, and discuss its interoperabiity aspect from international standardization work.**

***Keywords*—Network Security, Smartphone Protection, Software Defined Network**

I. Introduction

The attack surface of today's cyberspace has been significantly enlarged by the rapid increase in smartphone usage and the proliferation of diverse mobile applications, which introduce a large variety of zero-day vulnerabilities. According to Schmidt et al. [1], smartphones started being targets for malware in June 2004, and as of January 2014, there are roughly 700,000 of cumulated Android malware samples observed according to a report published by Sophos [2].

There are many motivations for attackers to target smartphones. One is the number of users. According to the report from BI Intelligence [3], at the end of 2013, 6% of the global population own a tablet and 22% own a smartphone, while 20% own a PC. Another motivation is the fact that smartphones often hold much more personal information compared to PCs, keeping detailed records of users' contacts and SMS history as well as sensitive account information regarding banking, emails, social networks, and etc.

Mobile malware is one of the most significant cyber threats on smartphones. One outstanding example is iBanking, a criminal software targeting Android terminals. According to the report from RSA [4], iBanking Mobile Bot is controlled over HTTP or via SMS, and it allows bot herders to steal personal information by reading incoming SMS messages, intercept calls to the phone, and obtain files as well as contact lists from the phone. It also has a function of phone fraud which allows a bot master to gain money by calling premium rate telephone service and charging the victim an expensive toll fee. Besides, smartphones are sometimes used in DDoS attacks. Android DDoS Origin [5] is a malware controlled via SMS messages, including the victim's host and port number. It makes smartphones generate anomalous traffic.

In this paper, we focused on offloading firewall functions to OpenFlow-capable wireless access points (APs). Since OpenFlow provides powerful traffic control schemes, it facilitates the implementation of firewall functions which can filter traffic based on the header information of the network and transport protocols such as IP address, and TCP/UDP port numbers. The key idea is that our implementation runs on an access point for filtering packets. It contributes to save network bandwidth resource between the access point and other filtering devices. Besides, our implementation is developed based on open source applications and standard protocols. It may therefore have flexibility and operability without using vendor specific technology employed by proprietry products.

II. Related Work

Generally, smartphone platforms have a method for allowing users to grant permission to an application that needs to access certain types of data. In the case of Android, whenever an application wants to read the contact list, it declares the request for this permission in its manifest file, effectively asking users to grant the permission. If a user thinks something is wrong, he/she may prevent the application from accessing the contact list. In the case of iOS, a permission request dialog box appears at runtime whenever an application first requests access to any of several resources such as user's geophysical location [6]. The permission request process is intended to inform users about the risks of installing applications, and hence, users can only make correct security decisions based on permissions.

However, in the case of Android, users had limited understanding of the permission warnings according to Felt et al. [7]. The authors also observed that Android permissions fail to be informative to most users, while not being completely ineffective. In the case of iOS, Tan et al. reported that permission requests that include explanations are significantly more likely to be approved [8].

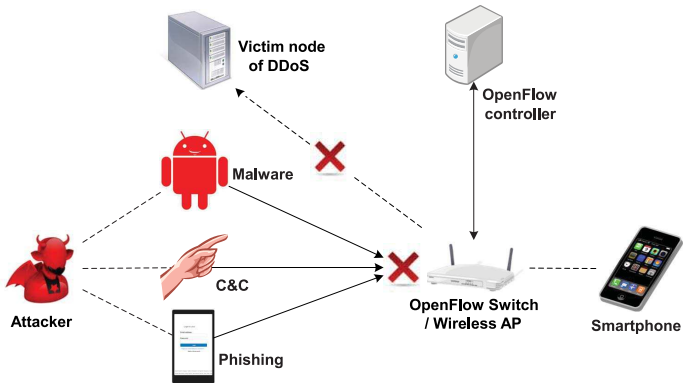

Fig. 1. An overview of smartphone protection using OpenFlow-capable APs

Another concern is the battery consumption of smartphones. Recently, almost all client OSes are equipped with the personal firewall function. However, there is a serious factor making the protection of the smartphone difficult, and that is battery consumption. Firewall often acts as a service and it stays active on the smartphone throughout the smartphone's operation. It is therefore important to ensure it has no severe impact on the battery [9].

III. Design

As we mentioned in Section II, the defense at the smartphone itself is difficult. Instead, we decided to offload smartphone firewalling function to network switching devices.

One type of defense mechanism is URL filtering, intended to block specific web sites for all smartphones. The objective of filtering is to prevent malware infection, botnet C&C, and phishing. In such attacks, the HTTP protocol is respectively used for malicious applications download, communications with the bot master, and fraudulent websites access.

Packet filtering is another possible defense to prevent smartphone devices from joining DDoS campaigns. Once a smartphone device is infected with a trojan horse, it starts to send data packets to a specified host (the victim) whenever it receives a DDoS attack command. It should be noted that the scope of DDoS mitigation is usually at the infrastructure layer rather than the endpoints, even when protecting smartphones. However, a major principle of DDoS protection stipulates that the mechanism should filter DDoS traffic as close to the attacker as possible. We therefore address DDoS mitigation at the AP, which may be the closest to the source as possible when we consider the case of a smartphone joining a DDoS campaign.

We also consider that the smartphone firewall has the ability to prevent infected smartphones from attacking neighboring smartphones in some types of network. For example, carrier networks tend to prohibit that a smartphone connects to other smartphones in the same network. However, in future networks, there is a possibility that carrier networks allow a smartphone to connect to other smartphones (e.g., virtual access point as proposed in IETF NVO3 WG[1]). In such case, a smartphone firewall will be helpful to thwart the attacks.

[1]https://datatracker.ietf.org/wg/nvo3/documents/

TABLE I. Specification of Wireless Access Point

Product	WZR-HP-1G300H
Vendor	Buffalo
CPU	Atheros AR 7161 (680Mhz)
Ram	128 MB
Disk	32 MB
Ethernet	AR 8136
Wireless	Atheros AR9223 (2.4GHz) and AR9220 (5.0GHz) 802.11abgn

Fig. 1 illustrates a high level overview of the architecture we propose. As we can see, APs are placed at the closest point to smartphone devices, and provide *URL filtering* to thwart smartphone malware, C&C, and phishing, as well as *packet filtering* to prevent smartphones from participating in DDoS campaigns.

From these consideration, we have two requirements of APs. One is *Filtering ability*. The APs must have the ability to protect smartphones from malicious entities. It is therefore capable to filter malicious URLs and to block suspicious IP addresses. Another requirement is *Operability*. The APs should be able to scale with the load of configurations.

IV. Implementation

We explored the suitable methods for URL filtering and packet filtering. In the case of URL filtering, web proxy systems have been widely used to block accessing malicious websites, and it is easier to be configured. In the case of packet filtering, we found a Software Defined Network technologies can provide powerful schemes for adding and/or deleting IP address in order to prevent smartphone from accessing to malicious hosts.

We chose RYU [10], a python framework for OpenFlow Controller (OFC), to develop our implementation as an OpenFlow application. Since there are few available wireless access points which are capable to OpenFlow, we installed OpenWrt [11], a Linux-based firmware for extensible configuration, into wireless access points and then used Open vSwitch [12] for OpenFlow Switch (OFS).

Our configuration for OFS are as follows.

1) Network Configuration: We configured three types of network, namely, wireless, internet, and management interfaces. The wireless interface is used for inside network where users' smartphone devices are connected. The internet interface is connected to the Internet, and the management interface is for interacting to the OFC. We also setup a pseudo device that we called bridge interface, in order to forward packets between wireless and internet interfaces.

2) Access Point Configuration: To accept the connection from smartphone devices, each AP is configured to capable to 802.11 n/g wireless networking protocols. The configuration also includes AP's Service Set Identifier (SSID), encryption, authentication, and signal strength.

3) OpenFlow Configuration: For interacting between OFC and OFS, we assign the private IP address to AP's management interface, and connect to OFS with Fast Ethernet.

Aside from OFS, our algorithm for packet forwarding running on OFC is summarized as Algorithm 1. Each packet incoming to OFS will ask OFC to check the IP address with the

Algorithm 1 Pseudo Code of Resilient Firewall

```
for all packets do
    Read the list of suspicious IP addresses
    if the source or destination IP address is listed then
        Discard the packet
    else
        Forward the packet
    end if
end for
```

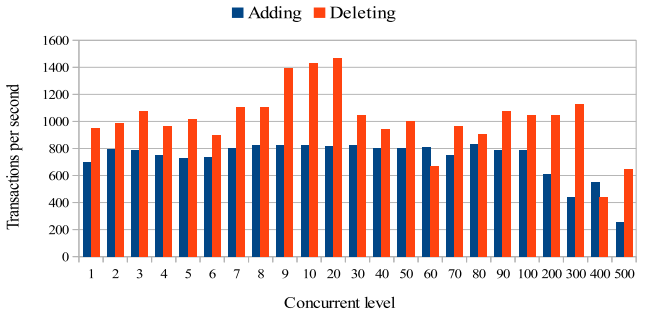

Fig. 2. Stress Test Analysis of the APIs using Siege

```
<?xml version="1.0" encoding="UTF-8"?>
<IODEF-Document xmlns="urn:ietf:params:xml:ns:iodef-1.0">
  <Incident purpose="reporting">
    <DetectTime>2015-11-01T22-00-00+09:00</DetectTime>
    <ReportTime>2015-11-02T09-00-00+09:00</ReportTime>
    <IncidentID>123456</IncidentID>
    <Description>DoS</Description>
    <Assessment>
      <Impact type="dos"/>
    </Assessment>
    <EventData>
      <Flow>
        <System category="source">
          <Node>
            <Counter type="event">10000</Counter>
            <Address category="ipv4-addr">192.168.1.1</Address>
          </Node>
          <Service ip_protocol="4"/>
        </System>
      </Flow>
      <Expectation action="investigation"/>
    </EventData>
  </Incident>
</IODEF-Document>
```

Fig. 3. Information Pipelining with IODEF message

list of suspicious IP addresses. If it is listed, the OFC controls OFS to discard the issued packet.

Our application developed for OFC runs HTTP server and accepts requests by using following APIs.

`/add_blocking_ip/{IPADDR}`
: This API is for adding specified IP address to the list of the suspicious IP addresses.

`/del_blocking_ip/{IPADDR}`
: This API is for deleting specified IP address from the list of the suspicious IP addresses.

`/delassoc_client/{IPADDR}`
: This API is used for deassociating a smartphone device which is assigned the specified IP address.

It should be noted that the disassociation of the smartphone device is the out of support in OpenFlow protocols. When this API is called, it runs such OS commands that extracting the MAC address from the IP address, and disconnecting the issued MAC address powered by IW [13], a CLI configuration utility for wireless devices.

V. Preliminary Evaluation

This section provides the results of our preliminary evaluation. We ran our implementation on a PC which has Intel Core i7 2.8GHz (2core) processors and 16GB of memory, and configured an OFC and an OFS. We then evaluated performance with Siege [14], a stress test tool. We queried these APIs via HTTP 1000 times, and observed the average number of transactions per second.

Fig. 2 shows the benchmark results for adding and deleting blocking IP address APIs on several concurrency levels, i.e., the number of clients which concurrently send requests. For each concurrency level, 1000 of randomly generated IP addresses were inputted to the APIs. X axis denotes the average number of transactions per second and Y axis denotes the concurrency level. The blue and orange bars denote the performance for adding and deleting the IP addresses, in respectively. Given the concurrent level was 500, we observed the worst performance, however, it was still more than 200 transactions per second. We will analyze its feasibility aspect from performance along with the reasonable threshold in our future work.

VI. Discussion

A. Information Pipelining

Threat information needs to be reported, exchanged, and shared among organizations in order to cope with cyber threats. Such exchanges, however, are mostly done by individual operators based on their own personal networks, and they are usually done by manual operations such as telephone calls, emails, and face-to-face meetings.

The viewpoint from the automated protection, there are a few pieces missing toward practical cyber defense. Our firewall should therefore address the information pipeline from the exchanged threat information to actual cyber defense. This section briefly introduces the common format for exchanging threat information and how we can deal with.

1) IODEF message exchange: Incident Object Description Exchange Format (IODEF) [15] is an international standard for exchanging incident information. It defines a data model and its XML schema for describing incident information and enables exchange of the information between computers. It has been already used by several organizations, such as US-CERT, and has advanced the automated exchange of information.

Our developed system supported a translator which can convert IODEF message exchange to our REST-ful APIs. Fig. 3 shows an example for IODEF message to inform DoS attacks. Our translator uses iodeflib [16], a python framework to read and edit IODEF message, and then extract the issued IP address, in this case, 192.168.1.1. It finally accesses to our REST-ful APIs to mitigate the DoS Attack.

2) n6 message exchange: n6 is a platform for processing security-related information, developed by NASK, CERT Polska [17]. Its API provides a common and unified way of representing data across the different sources that participate

```
{
  "address": [ { "ip": 192.168.1.1, "asn": 65001 } ],
  "category": "dos",
  "source": "source-name",
  "confidence": "high"
}
```

Fig. 4. Information Pipelining with n6 message

in knowledge management. n6 exposes a REST-ful API over HTTPS with mandatory authentication via TLS client certificates, to ensure confidential and trustworthy communications. Moreover, it uses an event-based data model for representation of all types of security information. Each event is represented as a JSON object with a set of mandatory and optional attributes.

Our developed system also supported a translator which can convert n6 message exchange. Fig. 4 shows an example for n6 message to inform DoS attacks. Our translator reads JSON format to extract the issued IP address, 192.168.1.1.

B. Interoperability

This paper demonstrated the feasibility of dynamic firewall management for smartphone devices. Our prototype used IODEF and n6, which are designed for the purpose of incident information sharing, and is not designed for firewall management. We do not particularly stick to these data model since it is more important to control the firewall and protect the smartphones regardless of the data model of the messages. Having said that, if we have a standardized common data model for that, it would help facilitating interoperability.

Until now, we do not have any prominent and standardized technique for firewall management, but we believe having such a standardized method will help reinforcing the android users. One such activity was just initiated in IETF I2NSF [2] working group. The working group aims at controlling and managing firewall on the network. It for instance defines the framework of the remote firewall management, and the data model of the information exchanged with firewall entities. We are hoping to see that this particular activity will be the striking one.

Though our current implementation does not cope with I2NSF at this moment since it has not produced any RFC yet, we are happy to cope with its deriverables once it produce a standardized technique. On the other hand, we hope that our activities could also contribute to the development of these standardization activities.

VII. Conclusion

In this paper, we presented new firewall work on OpenFlow-capable wireless access points (APs). Our prototype was based on OpenWrt, and was configured to deal with an OpenFlow protocol in order to facilitate adding new filtering rules. We then implemented an OpenFlow controller to provide REST-ful APIs that are available for blocking or unblocking the specified IP address, as well as deassociating a smartphone device which was assigned the specified IP address.

Since our system addressed the information pipelining toward automated cyber defense, we reviewed several threat information exchange formats and confirmed that our system could deal with them. We also discussed the interoperability among firewall functions powered by software defined network, and speculated that our system can be feasible to have the interoperability.

[2]https://datatracker.ietf.org/wg/i2nsf/documents/

Acknowledgment

This research has been supported by the Strategic International Collaborative R&D Promotion Project of the Ministry of Internal Affairs and Communication, Japan, and by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 608533 (NECOMA). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the Ministry of Internal Affairs and Communications, Japan, or of the European Commission.

References

[1] A.-D. Schmidt, H.-G. Schmidt, L. Batyuk, J. H. Clausen, S. A. Camtepe, S. Albayrak, and C. Yildizli, "Smartphone malware evolution revisited: Android next target?" in *Proceedings of the 4th International Conference on Malicious and Unwanted Software*, Oct. 2009, pp. 1–7.

[2] V. Svajcer, "Sophos Mobile Security Threat Report," Available at: http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.pdf, 2014.

[3] J. Heggestuen, "One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet," Available at: http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10, Dec. 2013.

[4] RSA Online Fraud Resource Center, "The Current State of Cybercrime 2014," Available at: http://www.emc.com/collateral/white-paper/rsa-cyber-crime-report-0414.pdf, May 2014.

[5] Dr.Web, "New Trojan for Android can mount DDoS attacks," Available at: https://news.drweb.com/show/?i=3191&lng=en, Dec. 2012.

[6] Apple Inc., "iOS Developer Library," Available at: https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS9.html, Oct. 2015.

[7] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*, Jul. 2012, pp. 3:1–3:14.

[8] J. Tan, K. Nguyen, M. Theodorides, H. Negrón-Arroyo, C. Thompson, S. Egelman, and D. Wagner, "The Effect of Developer-specified Explanations for Permission Requests on Smartphone User Behavior," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Apr. 2014, pp. 91–100.

[9] J. Vincent, C. Porquet, M. Borsali, and H. Leboulanger, "Privacy Protection for Smartphones: An Ontology-Based Firewall," in *Proceedings of the 5th Workshop in Information Security Theory and Practice*, Jun. 2011, pp. 371–380.

[10] Nippon Telegraph and Telephone Corporation, "Ryu Network Operating System," Available at: http://osrg.github.com/ryu.

[11] OpenWrt Project, "OpenWrt," Available at: http://wiki.openwrt.org/about/start.

[12] Open vSwitch, "Production Quality, Multilayer Open Virtual Switch," Available at: http://openvswitch.org.

[13] IW, "Linux Wiless," Available at: https://wireless.wiki.kernel.org/en/users/documentation/iw.

[14] J. Fulmer, "Siege," Available at: https://www.joedog.org/siege-home/.

[15] R. Danyliw, J. Meijer, and Y. Demchenko, "The Incident Object Description Exchange Format," Available at: http://www.rfc-editor.org/rfc/rfc5070.txt, Internet Engineering Task Force, RFC 5070, Dec. 2007.

[16] Decalage, "iodeflib - a python library to create, parse and edit IODEF incident reports," Available at: http://www.decalage.info/python/iodeflib.

[17] CERT-Polska, "n6sdk," Available at: https://github.com/CERT-Polska/n6sdk.