On preventing suspicious communication from IoT devices by utilizing DNS name resolution

Takayuki Fujino¹, Yuya Sasaki¹, Yuka Futagami¹ and Tomohiro Enomoto¹
¹Faculty of Engineering, Kindai University
¹Takaya-Umenobe, Higashi-Hiroshima, Hiroshima 739-2116, Japan
a) fujino@hiro.kindai.ac.jp

Abstract:

The number of IoT devices connected to the Internet has been rapidly increasing. On the other hand, some IoT devices may not have sufficient security capabilities because they do not have enough resources. Previous work to monitor IoT device traffic used a monitoring node running packet sniffing software, but it required an extensive resource. We propose new IoT traffic monitoring method which utilizes DNS name resolution. Our method not only decrease CPU resources, but also prevent suspicious communication from IoT devices. We implemented the prototype to verify the proposed method can detect and prevent suspicious communication with lower CPU load.

Keywords: IoT, DNS, Security

Classification: Network System

References

[1] G. Davis, "2020: Life with 50 billion connected devices," Proc. 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, United States, pp.1-1, Jan.2018, DOI: 10.1109/ICCE.2018.8326056

[2] J. A. Jerkins, "Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code," Proc. 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, United States, pp.1-5, Jan.2017, DOI:10.1109/CCWC.2017.7868464

[3] T. L. von Sperling, F. L. de Caldas Filho, R. T. de Sousa, L. M. C. e Martins and R. L. Rocha, "Tracking intruders in IoT networks by means of DNS traffic analysis," Proc. 2017 Workshop on Communication Networks and Power Systems (WCNPS), Brasilia, Brazil pp.1-4, Nov.2017, DOI: 10.1109/WCNPS.2017.8252938

[4] P. V. Mockapetris, "Domain names - concepts and facilities", RFC 1034, Nov.1987, DOI: 10.17487/RFC1034

[5] P. V. Mockapetris, "Domain names - implementation and specification", RFC 1035, Nov.1987, DOI: 10.17487/RFC1035
1 Introduction

The number of IoT devices connected to the Internet has been rapidly increasing due to its convenience. It is expected that the number of connected devices will reach 50 billion by 2020 [1]. On the other hand, some IoT devices may not have sufficient security capabilities because they do not have enough resources. In October 2016, DNS provider Dyn was suffered from DDoS attack from the Mirai botnet, which consisted of various IoT devices such as Web cameras, home routers, etc [2]. Given the number of IoT devices and their heterogeneity, approaches to managing individual IoT device configurations are unpromising. One of the better approaches is to monitor the IoT network to detect suspicious communication and prevent them. In this paper, we use the word "suspicious" in the sense that it is not clearly authorized and potentially harmful. In networks used by human users, it is very difficult to identify if a particular communication is suspicious. In contrast, IoT network is used by IoT devices, which can be considered a programmed object. This means that communication only takes place within a predetermined destination range. Therefore, communication outside that range can be determined to be suspicious.

2 Previous work

In order to monitor the IoT network, Sperling et al. proposed monitoring node based architecture [3]. Their architecture consists of two functional elements, a collector and an analyzer. A collector nodes are deployed on the IoT network and collect the network traffic through packet sniffing software. They then send the captured data to an analyzer node. The analyzer node gets traffic data from the collector nodes and investigates if there are suspicious communication. The authors conducted functional tests of their implemented prototype. Experimental results showed that they succeeded in detecting suspicious nodes that performed DNS lookups through DNS cache servers that were not approved by the administrative organization.

The Sperling’s method has some problems. The collector node needs a lot of CPU resources because it uses packet sniffing software. Their method can’t detect communication to an unauthorized destination whose name has been resolved by an approved DNS cache server. Above all, their method can only detect suspicious communication and never prevent them.
3 Overview of our proposed method

We propose new IoT traffic monitoring method which utilizes DNS name resolution [4]. Similar to [3], we also employ a monitoring node based architecture. However, our proposed method does not require packet sniffing at the IP layer, rather we primarily focus on DNS name resolution at the application layer. Basically DNS name resolution is performed by every node which hopes to communicate with the Internet Protocol. The node queries the DNS cache server for the destination host name to obtain the destination IP address. Our basic idea is to check if i) specified DNS cache server is valid (contained in approved server list), ii) queried name falls in predefined white list (composed of authorized domain name list). To achieve this, we adopt transparent DNS proxy. A transparent DNS proxy is a server which transparently provides proxy function for DNS name resolution. Specifically, it intercepts DNS query message, send same query message to the configured DNS cache server on behalf of original client, obtains the answer corresponding to the query, then return the same answer to the original client. An important feature of a transparent DNS proxy is that any client does not need additional configuration. At the same time, from the viewpoint of adversary, she/he can't determine if a transparent DNS proxy exists or where the server located. In the proposed method, the transparent DNS proxy checks if query message is suspicious or not. If the transparent DNS proxy judges that certain query is suspicious, it prevents the subsequent communication involved with that query. Specifically it returns an NXdomain response, which is a DNS error message indicating that there is no host matching to the queried name. Eventually the client (IoT device) can't establish IP connection because the NXdomain response does not contain any IP address for it.

Fig. 1 shows brief overview of the proposed method. Monitoring node is installed between IoT devices and exit point of IoT network. In terms of DNS name resolution, possible IoT devices' behavior can be categorized into following four: (1) query authorized name to approved DNS cache servers, (2) query authorized name to unapproved DNS cache servers, (3) query unauthorized name to approved DNS cache servers, (4) query unauthorized name to unapproved DNS cache servers. The monitoring node (the transparent DNS proxy) returns IP addresses only in the case of (1).
4 Implementation details

4.1 DNS message architecture and terminology
DNS name resolution uses two messages, query and response. Both messages have the same format which consists of a header and a body. The body part is further divided into four sections: Query Section, Answer Section, Authority Section, Additional Section. The details of header structure and each section are given in [5]. We will use the terms commonly used in [5] to refer certain part of the header, resource record type or specific sections. For example, "ARCOUNT" refers the header part indicating the number of resource records stored in Additional Section.

4.2 NXdomain response generation
As described in Section 3, the transparent DNS proxy returns an NXdomain response to a suspicious query. This means that it needs to generate the NXdomain response even though the queried name is resolvable. In order to better understand the NXdomain generation process, we will briefly explain the differences between the various responses in advance. Table I shows the comparison of message structure among the normal A response, the NXdomain response, and the normal SOA response. "Yes" means that the section exists, "No" means that there is no corresponding section in the message. An IoT device always issues a query for A resource record. If the transparent DNS proxy send the query for A on behalf of the device, it will receive the normal A response which does not contain the Authority Section. However, the Authority Section is necessary to generate the NXdomain response. On the other hand, the structures of the NXdomain response and the normal SOA response are similar. The only difference between them is whether Additional Section exists or not. Based on those message structure differences, NXdomain response can be generated in the following steps.

1. Change QTYPE from A to SOA.
2. Send query for SOA to the configured DNS cache server.
3. Get response for SOA query.
4. Change RCODE from NO_ERROR to NAME_ERROR.
5. Set ARCOUNT to zero.
6. Delete all resource records in Additional Section.
7. Restore QTYPE from SOA to A.

| Type               | Normal A | NXdomain | Normal SOA |
|--------------------|----------|----------|------------|
| Rcode              | NO_ERROR | NAME_ERROR | NO_ERROR   |
| Query Section      | Yes      | Yes      | Yes        |
| Answer Section     | Yes      | No       | No         |
| Authority Section  | No       | Yes      | Yes        |
| Additional Section | No       | No       | Yes        |

5 Experiment

5.1 Experimental setup
We build a test environment for the performance evaluation of both the proposed method and the Sperling’s method. We implement the proposed DNS proxy
feature in Java, which made transparent by utilizing iptables [7]. Sperling’s method is implemented by using tcpdump [8]. Same as [3], experimental network is built using Raspberry Pi. The network has one data provider node, one monitoring node, and two client nodes. Two client nodes simulate IoT devices, one is sound and another is compromised. Both clients periodically try to get some data from the data provider node. Note that DNS queries issued by the compromised node are always suspicious. It is either destined for an unauthorized DNS cache server or targeted for an unauthorized host name. We measure the CPU load on monitoring node through sysstat [9].

5.2 CPU load evaluation

Fig.2 shows the comparison of the CPU load on monitoring node. The numbers in red indicate the average value. The average values are taken from the log files. CPU load average of the proposed method is 0.77 while Sperling’s method is 3.11. As mentioned in the previous section, packet sniffing in promisc mode still requires a lot of CPU resources. In contrast, the proposed method requires much less CPU power, even running on a Java virtual machine (which implies additional overhead compared to the native implementation). Again, while the proposed method can prevent suspicious communication, Sperling’s method can only detect and notify them.

![Comparison of CPU load in monitoring node](image)

Fig. 2. Comparison of CPU load in monitoring node
6 Conclusion

We showed that our proposed method can decrease CPU load drastically as well as prevent suspicious communication. However, the transparent DNS proxy assumes that a common UDP/53 port is specified in DNS name resolution. If adversary uses DNS over https [6], she/he will be able to circumvent the proposed monitoring architecture. We are currently dealing with this problem. We will report that accomplishment in the future.