IoT Security Vulnerability: A Case Study of a Web Camera

Yogeesh Seralathan*, Tae (Tom) Oh**, Suyash Jadhav**, Jonathan Myers**, Jaehoon (Paul) Jeong+, Young Ho Kim^, and Jeong Neyo Kim^

**College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY United States of America*

***Department of Computing Security, Rochester Institute of Technology, Rochester, NY, United States of America*

+Department of Interaction, Science Sungkyunkwan University, Suwon, Republic of Korea

^Cyber Security Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

ys4815@rit.edu, tomoh@rit.edu, ssj8127@rit.edu, jlm4508@rit.edu, pauljeong@skku.edu, wtowto@etri.re.kr, jnkim@etri.re.kr

***Abstract*— *The Internet of Things (IoT) are devices which are connected and controlled over the internet. The use of IoT devices has increased exponentially over time and knowingly or unknowingly our data is captured by IoT devices on a daily basis. Recent news on malware targeting IoT devices and some current research reveals that in most cases there are no security controls implemented on these devices. The exponential rise in the use of IoT devices, more processing of sensitive data by these devices, and their mass exploitation was the motivation behind our work. Malware like Mirai is currently being used to build large botnets which are used in DDoS attacks where up to 1.2 Terabytes of networks traffic is generated every second. We will discuss the threats when there is a compromise of an IoT device's security and provide a case study of an IP camera. We also cover aspects of how and why modern malware targets IoT devices specifically. We finally discuss the importance of securing IoT and provide essential security practices for mitigating device exploitation.*

***Keywords*— *Internet of Things, security, vulnerability, Wire-Shark, nmap*

I. Introduction

The Internet of Things (IoT) are devices connected to the internet which hold the capability of doing tasks with or without human interaction. IoT devices are used for multiple purposes but are most commonly used in daily automation. These devices collect data from sensors then either store the data locally, send the data to the cloud via the internet, or process the data and internally send commands to other devices for further action.

IoT devices are used in every possible domain we can think of including medical, industrial, and home automation. Oftentimes these devices collect a vast amount of critical information which is processed by machine learning algorithms to enhance the product and build new products. A large privacy problem is introduced as these devices collect and transfer sensitive information about individuals. This becomes an even larger issue with data collected from health tracking devices, home environment monitoring devices, financial devices, and most importantly cameras. While these devices collect sensitive information they also process the data and are sometimes left to make critical decisions with this data. Take for instance a pacemaker which is a device responsible for processing vital data and directly taking action on a human body. With just a few malicious commands an adversary can even cause the death of human.

As we see IoT devices collect more sensitive information and control larger infrastructures it inevitably becomes a target for attacks. IoT devices were originally not designed with security in mind as the security community did not explore IoT devices during its early emergence. Later as attackers realized it is easily to break into these devices without much effort devices started getting exploited by the masses. Attackers scan the whole internet and target these devices to build massive botnets which lead to 1.2 Terabytes of network traffic being used in DDoS attacks. Following these incidents, the security community started creating awareness on how insecure IoT devices are and why it is important to secure them.

II. Related Study

A. IoT Hacking Case Study

The author, Craig Heffner [4] mainly focuses on IP cameras which are connected to the internet and accessible to anyone on the internet. Heffner analyses the devices using the firmware images provided by the vendor in their respective websites but verified his 0-day vulnerabilities on the live devices. The author also demonstrates how easy it is to find and exploit vulnerabilities on the targeted cameras using existing tools. 10 Dlink Camera devices which were selected by the author had a web server running on Lighttpd server. The attackers found a series of vulnerabilities which gave out the administrative password to the devices and gave the attackers access to a script which was running as root and held a remote code execution vulnerability. Further, the author checks for the tested devices in Shodan and found more than 20,000 devices running and accessible to the internet. Linksys cameras also had a remote code execution vulnerability however it was slightly more complicated to exploit compared to the Dlink. The author reverse engineered the binaries of the firmware and forced the vulnerable application to return the hardcoded admin credentials which were encoded with a very weak base64 format.

Nest thermostat is a famous product of Nest Labs which was acquired by Google for $3.2 billion. This thermostat is a smart IoT device which monitors users heating and cooling settings and eventually learns the user's ideal settings and adjusts the setting for better efficiency of electricity. This device is connected to both Wi-Fi and Nest Cloud which is used to control any registered device.

Nest Thermostat was used for the security research by researchers by Hernandez1, Arias1, Buentello2 and Jin1. In this case [3] the attack was on the bootloader, which was attacked by injecting malicious USB during boot and sending x-loader which granted access to the Nest file system. With this access, the attacker can install any binary by compiling it for the Nest environment. Netcat and Dropbear (SSH server) was installed to maintain a persistent backdoor connection with the devices. The authors claim that these backdoors can be used in a massive botnet however building large botnets with this particular vulnerability is not possible as it requires physical access to the device.

The Author's proposed solution to securing this device is manufactures to leverage secure boot and to authenticate only signed binaries for executing in the user space. Another possible solution is to encrypt the file system. During the research, it was found that the device used strong encryption for communication over the internet and had signed verification when patches were installed.

B. Major IoT Security Issues

Most IoT devices use Linux is their base operating system. Linux provides the flexibility of modifying and compiling the operating system to have just enough features to support a particular IoT device. The user also has the liberty to choose any low power Linux operating system already available. Given this, most vulnerabilities found in the operating system can also be exploited in IoT devices.

Learning from previous IoT device security analysis:

1) Weak server side controls.
2) No usage or broken cryptography.
3) Lack of binary protection.
4) Bad implementation of authentication or authorization.
5) Improper use of network services.

C. Major IoT Security Issues

Malware is simply a malicious piece of software which is installed on the host machine without the awareness of the machine's owner or administrator. Malicious software is designed with the intention to achieve a variety of tasks ranging from stealing sensitive information to building a massive botnet of slave devices.

Mirai is an example of a major IoT oriented malware which caused substantial damage. Mirai was first detected during a DDoS (Distributed Denial of service attack) on the website of journalist Brian Krebs [1]. Mirai exploits basic flaws in IoT devices like hard coded usernames and passwords for telnet. Mirai has a preloaded set of username and password combinations which it uses to brute force the device. Mirai mainly targets cameras as they have high computational power compared to other IoT devices.

Once Mirai successfully exploits a device it converts the device into a bot which is controlled by the command and control server. Mirai has the capacity of performing various types of DDoS attacks like DNS, UDP, STOMP, SYN, and ACK flooding [2].

There are many other types of IoT oriented malware which have caused significant damage including Imeij, Brickerbot, Remaiten, Linux.Darlloz, and many others.

III. Case Study Of An IP Camera

This section will cover the analysis of the existing security features in the camera and try to find the weakness in security (vulnerabilities) of the camera. Devices which are connected to the internet face the maximum amount of threat which is why we chose a camera which connects directly to the internet. The IP camera brand name and model are not revealed in article. Just a generic IP camera is mentioned in article.

Figure 1. IP Camera

A. Exploring the Camera

The IP camera provides the facility of access only through mobile platforms such as android and iOS. To add a new device or view previously added devices the user must use their IP camera account which has all the added devices attached to a IP camera account. For the first connection, the camera needs to know the SSID and the password of the Wi-Fi so it can connect to the internet. To accomplish this the information is directly pushed from the application to camera.

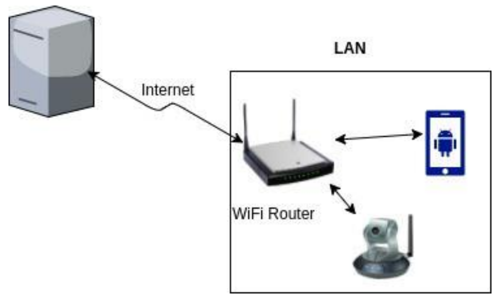


Figure 2. IP Camera Connection Overview

Once the camera is connected to the internet almost all other communication happens via IP camera's dedicated servers. A remote server is required to be in the middle of all network communication so that the user can view the video feed anywhere from the internet.

One interesting observation when working with the camera is when both the camera and mobile client are in the same LAN (local area network). When the IP camera server detects that both the streaming device and viewer have the same external IP the server finds the local IP of the camera and informs the mobile device. This can be seen as ARP messages during a packet capture within the LAN. Once the camera contacts the mobile device in the LAN all of the video streaming happens over the LAN which avoids internet usage and reduces the burden on IP camera servers. Not only does this decrease the load on IP camera servers but also decreases the lag in video streaming and increasing the quality of the video stream. This observation was found from the network capture mentioned in section III-B2.

B. The Camera's Security Perspective

Every device when added to a IP camera account, a password is attached to the device. So, it is not possible to add any device to the mobile and get the video stream from the server as it requires authentication.

To perform more in depth analysis on camera security it is required to do network and application security analysis. Easier way to find application related security flaws is by getting root access to the camera or get access to the firmware of the camera. Which would give access to applications running on device and sometimes even finding configuration files with user name and passwords. But in this case the IP camera did not provide the firmware for the camera so the analysis started with network security services.

1) Inspecting Network Services:

NMAP(Network Mapper) network scanning tool can be used to find the IP address of the camera to proceed with further testing. This can be done by scanning for all the live hosts in the network but because the camera does not have any host name it is hard to differentiate it from other devices. This scan is done over the LAN so it is easier to filter the known devices which will give the IP address of the camera. To find this device while scanning the internet a use of open ports and services running on the open ports can help drill down to the most accurate results.

To scan for open ports and finding the services running in the devices to further investigate the services NMAP was ran with the "-A" flag to output all details of services running on the device.

1) Port: 554/TCP RTSP.
2) Port: 5000/TCP SOAP.

RTSP is a real time streaming protocol which is used to control streaming media servers. It supports multiple commands like OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, GET PARAMETER, SET PARAMETER and USER_CMD_SET. Another interesting detail about RTSP is that all requests have URLs very similar to HTTP (Hypertext Transfer Protocol).

2) Inspecting Communication Between Devices:

Capturing all network level packet data can be done using network packet capture tools like Wireshark or TcpDump. In our case we used Wireshark to achieve this task. As in our setup 2 we can observe that both the camera and the mobile device are on the same LAN network, so Wireshark can be used to capture all LAN packets.

Wireshark can run in many modes like non-promiscuous mode, promiscuous mode, and monitor mode. Non-promiscuous mode is the normal mode which the tool runs on. In this mode, it captures all the data entering and existing the device. In promiscuous mode, it captures all the data which enters the network interface even it is filtered out of the Ethernet layer. But not all hardware and software support this mode so this is not a guaranteed approach. If promiscuous mode works fine on the host machine then the proceedings task becomes easy otherwise there is a need for a special network setup to capture all the data. This network setup will be addressed further in this section.

There are two flows of data which needs to be captured; data flowing in and out of the mobile device and data flowing in and out of the camera. These two data flows must be monitored to understand the inner workings of the data flow and find potentially sensitive information transferred to the server or other devices.

Capturing data flow from mobile is easy since it can be done by setting a proxy in android/iOS phone settings or by installing an emulator. We achieved this task using Androidx86 as it gives complete control over the operating system. The tricky part is capturing the data from the camera to the server or mobile device as it is not possible to set any proxy settings in the camera. This is achieved by using a network tap as shown in the setup figure 3.

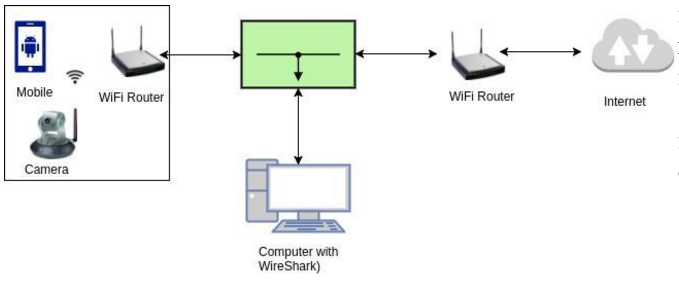


Figure 3. IP camera All Data Capture Setup

The network tap as shown in figure 4 is used to capture all the communication in between two ends of a network. As shown in the figure , two ends A and B can be a different network all by itself or it can be a device. In our A we have a device and our B is a separate network. All the network flowing between A and B is also duplicated to C

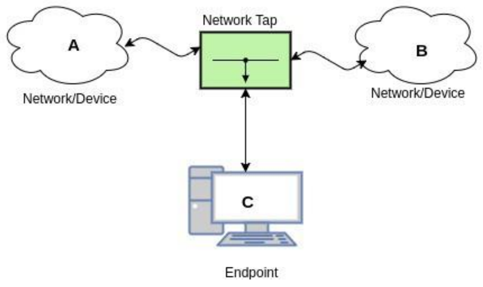


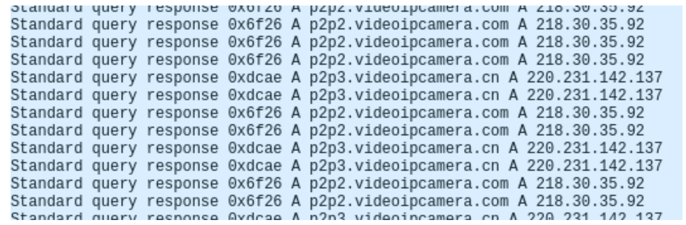
Figure 4. Overview of Network Tap

Once the network setup described in figure 4 is ready Wireshark should be started and running on device C. Wireshark will be capturing all the data but in our case we need data only between device A and C. We can leverage Wireshark filters to achieve this.

To filter only IP camera data:
"ip.src==[IP camera ip] or ip.dst==[IP camera ip]"
To filter only mobile data:
"ip.src==[mobile ip] or ip.dst==[mobile ip]"
To capture both IP camera and mobile application data concatenate either the filter with *or* operator.

Wireshark will populate with a huge amount of data once this is setup properly. To see a particular data stream a feature called "Follow TCP" can be used to get all the data of a particular context.

One of our observations was mentioned in section III-A on how the camera shifts its video streaming via LAN rather than streaming via IP camera servers. Another observation is that all IP camera media streaming servers are of format p*p*.videoipcamera.cn 5. The "*" in the URL can be replaced with any number between 1 and 6, so for example p1p5.videocamera.cn is a valid IP camera server. This information is very useful for attackers as this server stream contains all the video from all the camera's. The same servers are probably used by all the IP camera camera's which would become a single point of failure of all the companies' cameras if the IP camera servers are vulnerable. This paper only restricts its research to the IoT devices security implementation rather than on media streaming server security so we will not look into the IP camera's server security since it is not legal to perform security testing on a server which is not owned by the security analyst/tester.



```
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0xdcae A p2p3.videoipcamera.cn A 220.231.142.137
Standard query response 0xdcae A p2p3.videoipcamera.cn A 220.231.142.137
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0xdcae A p2p3.videoipcamera.cn A 220.231.142.137
Standard query response 0xdcae A p2p3.videoipcamera.cn A 220.231.142.137
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0x6f26 A p2p2.videoipcamera.com A 218.30.35.92
Standard query response 0xdcae A p2p3.videoipcamera.cn A 220.231.142.137
```

Figure 5. IP Camera Servers Contacted by Camera (WireShark Data)

Observations related to security of the camera can be found in section III-C.

C. *Exposing Weaknesses (Vulnerabilities)*

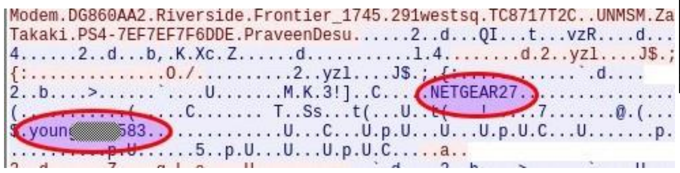
In this subsection, we will be looking various weakness in security implementation of the IP camera shown in Figure: 1.

1) *Unencrypted Communication:*

During the network data capture phase as explained in section III-B2 and the setup shown in Figure 3 we can observe that all data was transferred in plain text. This means that all data which was communicated between the camera and mobile application, the mobile app and the IP camera server, and IP camera server and camera are all readable in plain text.

There was no use of any form of cryptographic modules in the device, so all communications were *not encrypted*. Sensitive information sent from the mobile device such as IP camera registration and login accounts details are sent in clear text which can lead to compromise of all the camera devices attached to the IP camera account. The Camera id and password which is initially set for the camera's authentication can also be captured in clear text.

Sending all the data without encrypting does not only compromise the camera security but also the LAN security in which the camera is present. In the initial stages of activating the camera, the mobile application pushes the Wi-Fi SSID and password. We notice in figure 6 that this was captured during the Wi-Fi password update phase as this information was sent to the IP camera servers in clear text. If an attacker performs a man-in-the-middle attack and captures all the traffic they would be able to join the Wi-Fi network and sniff all further data.



```
Modem.DG860AA2.Riverside.Frontier_1745.291westsq.TC8717T2C..UNMSM.Za
Takaki.PS4-7EF7EF7F6DDE.PraveenDesu......2..d...QI...t...vzR....d...
4......2..d...b,.K.Xc.Z......d...........1.4........d.2..yzl....J$.;
{:.............0./..........2..yzl....J$.;.{...........`.d....
2..b....>......`....U.......M.K.3!]..C....NETGEAR27.............
(..........(.....C....... T..Ss...t(...U..t(...!...7.......@.(...
$.youn[redacted]583.............U...C...U.p.U...U...U.p.U.C...U.......p.
..........p.U......5..p.U...U...U.p.U.C.....a..
```

Figure 6. Wireshark Captured Wi-Fi Credentials in Clear Text

Figure 6 shows the screenshot of the data captured by Wireshark. The data displayed was obtained using the Wireshark filter ”Follow TCP stream”, which selects all the TCP packets of the particular packet selected and dumps all of its data. In the screenshot, the two red elliptic marks show the Wi-Fi identifier and its password (censored).

2) *Brute Forceible RTSP URL to Stream Video:*

In section III-B1 the results of the Nmap scan reveal that it an RTSP service was found running on port 554. RTSP is similar to HTTP as it has URLs and control commands. Streaming using RTSP will produce a URL that look like ”rtsp://mediaserver/stream1”. Authentication in the RTSP protocol works similarly to the HTTP basic authentication.

During the network data capture phase, there was a hunt for the authorization portion within the RTSP URL but we could not find the RTSP URL in the network data captured. A simple method to find the URL will be to simply brute force the URL for common stream names. Nmap provides the capability of running scripts which leverages the capabilities of Nmap. RTSP brute force script ”rtsp-url-brute” runs with a list of commonly known RTSP URL’s and checks if they work with the device. Nmap RTSP URL brute force command:

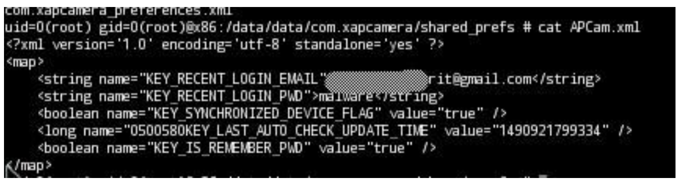
Nmap –script rtsp-url-brute –p 554 [IP camera ip]

But in our case, it failed to find the valid RTSP URL. So more common patterns of URL were added to the list of common RTSP URL’s from the internet. Then nmap was able to catch the valid URL of RTSP video streaming. Once the URL was found the video was directly streamed on the TCP connection. There is no authentication of any request to stream data. To demonstrate this, VLC can be used to stream an RTSP video stream by adding the RTSP URL in the network stream URL section.

RTSP URL for IP camera sp 009: ”rtsp://[IP camera ip]/onvif1”

3) *IP camera Account Credentials Stored in Clear Text:*

When inspecting the IP camera mobile application in android the IP camera account details were stored in clear text as shown in the figure 7.



```
com.xapcamera_preferences.xml
uid=0(root) gid=0(root)@x86:/data/data/com.xapcamera/shared_prefs # cat APCam.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="KEY_RECENT_LOGIN_EMAIL"[redacted]rit@gmail.com</string>
    <string name="KEY_RECENT_LOGIN_PWD">malware</string>
    <boolean name="KEY_SYNCHRONIZED_DEVICE_FLAG" value="true" />
    <long name="0500580KEY_LAST_AUTO_CHECK_UPDATE_TIME" value="1490921799334" />
    <boolean name="KEY_IS_REMEMBER_PWD" value="true" />
</map>
```

Figure 7. Credentials in Clear Text in Android App

The credentials can be found under the directory ”/data/datacom.xapcamera/shared_prefs”. The user email ID can be found under;

”KEY_RECENT_LOGIN_EMAIL”

Which censored and stored the password under;

”KEY_RECENT_LOGIN_PWD”.

But this bad implementation has low security severity level as all the application data is sandboxed by android. This makes it impossible to reach APCam.xml file for credentials without root privilege or another application flaw in IP camera’s mobile application.

IV. Conclusions

Three security related flaws where found while inspecting IP camera’ security posture:

1) Not using secure channel for communication, all sensitive data was transferred in clear text.
2) RTSP URL used to stream data was brute forcible as it a common RTSP URL.
3) IP camera account credentials are stored in clear text within the mobile application.

Any attacker can connect to the camera given that the camera is connected to the internet and gets video feed just by having its IP address. Attackers can even scan the internet to find open RTSP ports and try common URLs which this camera uses to broadcast its video feed.

The device successfully accomplishes a security objective is by not leaving any telnet or SSH service open. So Mirai’s initial step on hacking into this camera would have failed. But because the Mirai source code can be found online it is possible to scan for other attack vectors and create a botnet using these cameras. However, in this case we were not able to get root shell access so converting the camera into a botnet was not possible.

Suggested fix for the security flaws found in the camera:

1) Use OpenSSL library or any well-known open sourced SSL library for encrypting all the communication between the camera, application, and its servers.

2) Use Basic authentication in RTSP protocol to slow down the attacker brute forcing the URL parameter and the username-password combination.
3) Store the password using Android keystore and any similar options in iOS which encrypts the password and stores it in isolated locations.

ACKNOWLEDGMENT

This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0190-16-2032, Development of Operating System Security Core Technology for the Smart Lightweight IoT Devices)

REFERENCES

[1] Brain Krebs, *KrebsOnSecurity Hit with Record DDoS*, Sept 2016, https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos, accessed on 05/25/2017.
[2] Igal Zeifman, Ben Herzberg, and Dima Bekerman *Breaking Down Mirai: An IoT DDoS Botnet Analysis*, Oct 2016, https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html, accessed on 05/25/2017.
[3] Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin *Smart Nest Thermostat: A Smart Spy in Your Home*, Black Hat conference, 2014.
[4] Craig Heffner, *Exploiting Surveillance Cameras*, Black Hat conference, Feb 2013.

Yogeesh Seralathan received MS in Computer Science from Rochester Institute of Technology in 2017.
.

Tae (Tom) Oh (SM'09) received the B.S. degree in electrical engineering from Texas Tech University in 1991, and the M.S. and Ph.D. degrees in electrical engineering from Southern Methodist University in 1995 and 2001, respectively, while working for telecommunication and defense companies. He is currently an Associate Professor with the Department of Information Sciences and Technologies and the Department of Computing Security, Rochester Institute of Technology. His research includes mobile ad hoc networks, vehicle area networks, sensor networks, and mobile device security.

Suyash S. Jadhav born in Pune, INDIA on 31st March 1992. Holding Bachelor of Engineering in computer engineering form Pune University, India (2013); currently pursuing Master of Science in computing security at Rochester Institute of Technology.
He is currently working Google.

Jonathan L. Myers is currently pursuing his Bachelors of Science in Computing Security at the Rochester Institute of Technology.
He has worked as a Software Engineering Intern at Trademark Global in Lorain, Ohio. He currently works as a Research Assistant at the Rochester Institute of Technology. His areas of interest include binary exploitation, web application security, and security tool development.
Jonathan Myers is also an active member of the Security Practices and Research Student Association at the Rochester Institute of Technology.

Jaehoon (Paul) Jeong (M'12) received the B.S. degree from the Department of Information Engineering, Sungkyunkwan University, in 1999, the M.S. degree from the School of Computer Science and Engineering, Seoul National University, South Korea, in 2001, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota, in 2009. He is currently an Assistant Professor with the Department of Software, Sungkyunkwan University. His research interests include Internet of Things, vehicular networks, wireless sensor networks, and mobile ad hoc networks. He is a member of the ACM and the IEEE Computer Society.

Youngho Kim received his MS and BS degree in Computer Science from Korea University, Korea, in 1999 and 2001 respectively. He was a visiting research scholar at Rochester Institute of Technology (RIT) in 2013 and 2014. Since 2002, He has been a senior member of engineering staff at the Electronics and Telecommunications Research Institute (ETRI). His research interests include operating system, embedded system and mobile device security.

Jeongnyeo Kim received her MS degree and Ph.D. in Computer Engineering from Chungnam National University, Korea, in 2000 and 2004, respectively. She studied at computer science from the University of California, Irvine, USA in 2005. Since 1988, she has been a principal member of engineering staff at the Electronics and Telecommunications Research Institute (ETRI), where she is currently working as a management director of the Cyber Security System Research Department. Her research interests include mobile security, secure operating system, network security and system security