# 1. Introduction

The scenario being evaluated is an ARP spoofing attack on a local area network. The premise of an ARP spoofing attack is to disguise the attacking machine as another legitimate machine on the network. An ARP table, or address resolution protocol table is used to translate IP addresses into MAC addresses (Sean Reifschneider). This translation is critical for taking communications from internet layer to link layer. In an ARP spoofing attack, the attacker associates their MAC address with the IP address of a different computer on the network. When the local router tries to send information to the spoofed computer, the packets will be routed to the attacking machine instead. There are many attacks that can then be launched from this point of infiltration, such as: Man in the middle, session hijacking, or DOS attacks (ARP Spoofing). There are several methods for defending such an attack that are commonly implemented today. An ARP table can be generated using only static ARP entries. This method is not scalable for larger networks, and is generally not used. However, ARP spoofing detection software is prevalent. The basic mechanic behind ARP spoof detection is adding another layer of validation to the ARP communication. Most often, networks without detection software will assume the first client to respond is the proper client. Through our implementation we deployed all of the concepts above in either our attack or defence (Kaur, Rajwinder & Singh, Gurjot & Khurana, Suman, 433-434).

# 2. Reconnaissance

## 2.1 SMS Verification

SMS, or short message service, is a method of two factor authentication with the goal of authenticating the user based on something they have, typically a cellphone. SMS makes several assumptions. First, that the intended user has possession of the phone where the SMS is being sent. Second, the user is the only one who can read messages sent to the phone, that is the messages being sent are themselves secure. This means that there is no meddler in the middle anywhere between the server, the SMS gateway, the cellphone tower, or even between the tower and the user’s phone. Third, this form of validation assumes that your phone has the ability to receive the message at the time of the authentication attempt, which would lead to a loss of availability for the service. Interception of the authentication code can happen from entry points on the phone such as malicious software on the phone. The message containing the authentication code may be vulnerable from any portion of the PSTN, and has even been exploited at the local SMSC. Communications to the SMS Gateway can be tapped, and the Gateway itself can be compromised (Hummer, L.).

There has been some promising research using SS7 to compromise the PSTN using methods like faking cell towers. SS7 is fundamental to the operations of cellphones, but its security relies heavily on trust. As we know, the more a security system relies on trust, the more vulnerable it is. The exploits in this system were used to man in the middle messages, as well as track the locations of the users on the network (3GPP).

The main concern of SMS authentication is that the message can be redirected to an attacker’s phone through an attack known as SIM-swapping. The SIM (subscriber identity module) card in a cellphone is where the message is directed to from the cellphone tower. If an attacker is able to get a SIM card in their phone that is authenticated as belonging to the user’s phone, then they will be receiving the user’s messages, including an SMS authentication code. Frequently attackers are able to convince phone companies that they are the legitimate user, and they can get new SIM cards which messages will be sent to. Phone companies have attempted restricting the reauthorization of SIM cards through official policies, however these have not been particularly effective (Hoffman, C). SMS messages were simply not made with security in mind. It is possible to change SMS protocols so that they communicate securely however there are other, already existing alternatives. Phone application based multi-factor authentication requires a user to have the application installed on their device, and set up to work with the specific account. It provides superior security by only sharing the authenticated credentials once, when it is first set up. The credential is sent securely and outside of that one time the codes are simply generated on both the server and the user’s cellphone.

## 2.2 BioMetrics

Biometrics is a form of multi-factor authentication based on “what you are”. The goal of biometric authentication is to ensure information confidentiality, by verifying that the user has biological traits that are specific to the authorized user. Biometrics makes three major assumptions to provide meaningful security: first, that the metrics cannot be duplicated, or that they are significantly difficult to steal. Biometrics also relies heavily on the accuracy of the scanners. Because the scanners are not perfect, biometrics is not a deterministic method of security. A password, for example, is deterministic because the entered password is either correct or incorrect, whereas the bio scanners have a possibility of giving a false positive, or vice versa. Finally, it is assumed that the metric remains constant for the user with access. Although things like n iris pattern, or fingerprint typically remain the same, things like voice cadence, or even signature style may change over time. These assumptions have some significant failures. First, many biometrics are public domain. For example, your face, fingerprints, and iris patterns are left to the public all of the time. Anything you touch can leave behind a fingerprint, and high-resolution photos are detailed enough to accurately show ones iris pattern. Often times these ‘keys’ can be taken without the users knowledge. Because so much of the security in this authentication method relies on the accuracy of the scanner, most of the hacks, and subsequent patches have to do with the scanners themselves. Today, a 1 in 1000 false match rate is tolerated to consider the scanner valid (NIST Special Publication 800-63B). The hacking group called the CCC hacked the iris scanner in the GS8. They lifted the iris pattern of users from digital photographs, printed them out, and used a contact lense to mimic the curvature of the eye (Starbug, CCC). A more robust system is Apple's Face ID. The scanners used to verify the user include: infrared cameras, flood illuminators, proximity sensors, ambient light sensors, and user-facing cameras. To make an initial, ‘key’ image of the user, the phone projects 30,000 dots onto their face, and maps it to a 2D image. Improvements to the scanner have proven, thus far, a robust enough solution to spoofing. As the scanners get closer to behaving deterministically, this method will become increasingly more secure than other methods, as you will have much larger problems if someone has your face (Apple).

## 2.3 Network Address Authentication (Source Access Control by IP Address)

Network address authentication, also known as source access control by IP address, verifies users based on their specific IP address. The goal of network address authentication is the confidentiality and integrity of services that the defenders have decided should only be accessed from specific IP addresses or network. This process assumes that no one who should not have access to the network is able to gain access. It also assumes that attackers cannot cleverly exploit the protected system without receiving responses from it.

The first, most obvious flaw is that this process trusts the network security. Once the network with the specified IP address is broken into, the attacker can completely bypass this security measure, effectively making the network only protected by one factor. On large enough networks, it is not a matter of whether a network is compromised, but how many computers are. Also, given enough knowledge about the target infrastructure, the attacker may be able to spoof their IP and send in authenticated communications without ever needing a response from the system (NIST Special Publication 800-63B). An example of this process is access of enterprise tools and software. Many organizations restrict their most important tools, such as email, so that anyone who is using them must be on the organizations network, and have an IP address originating from their network in order to sign-in and access the sensitive information.

A robust example of access control based on network location is Microsoft OneDrive. Owners of a OneDrive can specify which ranges of IP’s are allowed to connect to the server (Shumate, Kaarin, et al.). This example works as a whitelist, but the IP filtering can also be done as a blacklist.

# 3. Infrastructure

Our infrastructure simulates a two factor authentication login portal where the user needs to enter the correct password and authentication code. In the insecured implementation the authentication code is sent over the network to a specific IP address. All of the computers on the network have assigned static IP addresses, and it was incorrectly assumed that no computer can get an IP address that belongs to another computer on the network. The implementation is similar to both network address authentication and SMS authentication. In our implementation we have the client connect to a server through a login portal. This login portal is run by a simple login script that communicates via network sockets. The user at the client enters their password. Before the password is sent, it is first combined with a large number that is unique to this program, known as a salt. Together the password and salt are run through the SHA-256 hashing algorithm. The received salted hash is compared to the salted hash that is stored on the server side. By both salting and hashing the code, even if the attacker can intercept the password they will be unable to reverse it without a rainbow table specifically generated for this salt. The server compares the hash to the already calculated hash that it has stored. If the hashes match the user is prompted to enter the authentication code that was sent to their static address. If the user enters the correct code that was sent to their static address then the user is granted access and is sent the secret information contained on the server. Under our infrastructure, which relies on socket programming, it will be necessary for both the client and the attacker to both have the login script, and know what port the server is listening for connections on.

An example of a real world implementation of this infrastructure could be seen in a login portal that sends an authentication code to a mail server with a static IP address, or perhaps just a general two-factor authentication code that is sent to a specific IP address. The user can view the message containing the authentication code, which they are prompted to enter at the login portal.

# 4. Attack

## 4.1 Introduction

We implemented MAC address spoofing to break the infrastructure. The premise of our attack is that the hacker has cracked the login password of the client using MAC spoofing but cannot receive the authentication code because the server only responds with the authentication code to a specific IP address after the password has been entered correctly. In order to receive the authentication code, the hacker first need to change the MAC address to impersonate the real client. In this way, the server will send the authentication code to the hacker’s machine, instead of the client’s machine. The hacker can then use both of the credentials to log in as the client, and access secret data.

## 4.2 Real implementation

During an actual attack, the attacker will change the MAC address of his network interface controller card to the MAC address of an authorized device in the network, (Cardenas, E. D.) and pretend to be the authorized machine. Through this process, the hacker is able to intercept data transferred from one of the machines to the victim. There are various ways to use MAC spoofing attack. A common way to use it is the man-in-the-middle attack, through which the hacker could intercept all the data transmitted.

## 4.3 Our implementation

Our first attempt started with simply changing the MAC address of the attacking machine and do nothing to the victim. However, this method is highly inconsistent, as MAC spoofing requires the attacker’s machine to respond his MAC address to the ARP broadcast faster than the victim does. ARP is basically an unverified response, so the early bird gets the worm. We were able to successfully spoof the mac address 1 out of 4 tries, which proved our method to be valid, but not optimal.

As 1 success out of 4 tries is inapplicable in an actual scenario, our group thought of causing a delay on the client’s machine before spoofing the MAC address, giving the attacker a guaranteed breathing moment to respond to any ARP broadcast faster than the real client. With several tries and fails, we set script to send 5 packets of 256 bytes to the client, which can cause an average of 3 milliseconds of delay on the victim’s machine - enough to spoof the client’s MAC address almost 100% of the time.

After successfully launched a MAC spoofing attack, the connection to the attacker’s machine via the original IP address will no longer be available. However, it is now possible to connect to the attacker’s machine using the victim’s IP address. After establishing a connection to the machine once again, we were able to run the login script, receive the authentication code, and login from the attacker’s machine.

# 5. Defense

For the defense portion, we simulated an authentication app where the server and the client exchange one time seed from which random values for successive login are generated. This method is more secure than generating a new passcode on the server side and sending it to the client each time. Our simulation of authentication app works similarly to the authentication apps that exist on the market today such as Google Authenticator, Microsoft Authenticator and Duo.

When the client first contacts the server, they securely exchange a seed and each store it on their own side. After the client receives the seed, every time the client user wants to log in to the server, a new pseudo-random number will be generated based on the seed and the current time. The number is not truly random as the server and the client will both generate the same new number based off the same seed that they have which allows this method of authentication to work. However, while this number is preuso-random not random, it still makes it much more difficult for the attacker to guess. An assumption must be made that attackers will not know the securely exchanged seed. It is very important that the seed be as protected as possible, even though it is stored on both the client and server. Breaches of these seeds have occured in the past, breaking the security of this method of authentication. Assuming the attacker doesn’t know the original seed, but was still able to intercept the code sent from the client to the server, then they will have to guess a new number every minute since our defense updates the seed and therefore the random number based off the number of minutes since Epoch. The seed is generated by taking a random number such as “710873509349142670” and changing it by the number of minutes since Epoch. Because of this the seed for the random number generator changes every minute. The attacker only has one chance to intercept the seed during the initial exchange. It may be possible for an attacker who can intercept enough of the sent authentication codes to reverse potentially weak random generation. However, in our implementation of this defense intercepting the authentication code is of little use because the code hash been salted and hashed before being sent.

Because of the reasons mentioned above, this method of multi-factor authentication is the one that is gaining popularity today. Many organizations unfortunately still rely on SMS text message as their factor of authentication, however like we mentioned in the recoinnase phase, that method has a lot of flaws and security vulnerabilities and opens up the user to attacks such as SIM hijacking and social engineering attacks. Authentication app on a mobile device is a more secure, but still fairly easy method to ensure additional security. This method does require the user to install the application on their device which assumes the user doesn’t mind additional application on their phone and has technical skills to set it up. However, it is our opinion that the security benefits of this method far outweigh the additional space requirement and the additional complications in setting up this method, therefore the companies especially ones that deal with sensitive data such as online banking should urge their users to employ this method.

Our implementation is shown in the “server-defense-authenticate” and “client-defense-login” in the “Defense” folder of the zip file. Figure 3 in the appendix also shows the login screens with the defense implemented.

# 6. Conclusion

In conclusion, it is our recommendation that more web services implement a multi factor authentication for arp entries, for example, the one demonstrated in the defense portion. We have shown how vulnerable some of the commonly used multi-factor authentication methods, such as the SMS text message. While having SMS text message as the second step in better than nothing, it is still much better to use something like the authenticator app. The apps are growing in popularity, however are still not as widespread as we would like to see. Many web services are hesitant to inconvenience their users and ask them for any sort of second factor authentication, not to mention an additional app. However, after our research, we suggest several options for increasing the implementation of the authentication app.

First, we believe that the web services should at least give user the option to set up an authentication with the app. There is still an unfortunately large amount of services that contain user’s sensitive data, but rely on SMS method of authentication or nothing at all. We urge that the web services consider the security risks we have described in this paper and begin offering users an option. Ideally of course, the web services would make it mandatory to have the second step authentication, however we also understand that some services fear losing some of their user base if they inconvenience them. A typical user might not understand all the security risks associated with only having a password and might not want to be pushed into a second layer of security. Therefore, we suggest that the web services provide educational information to their user base about the benefits of securing their account and provide them the option of multi factor authentication.

Second, we encourage users to reach out to their most commonly used web service providers and request multi factor authentication. Web services might be hesitant to include this feature if they believe user’s will find it inconvenient, but if they see that there is a demand for it, they are more likely to listen.

In the end, while we recognize that even the multi authenticator app as a method of security is not foolproof, it is the method we encourage the web services and users to use since it’s a fairly cheap and easy to install and set up method that we believe could gain popularity with a large user base.

# 7. Appendix

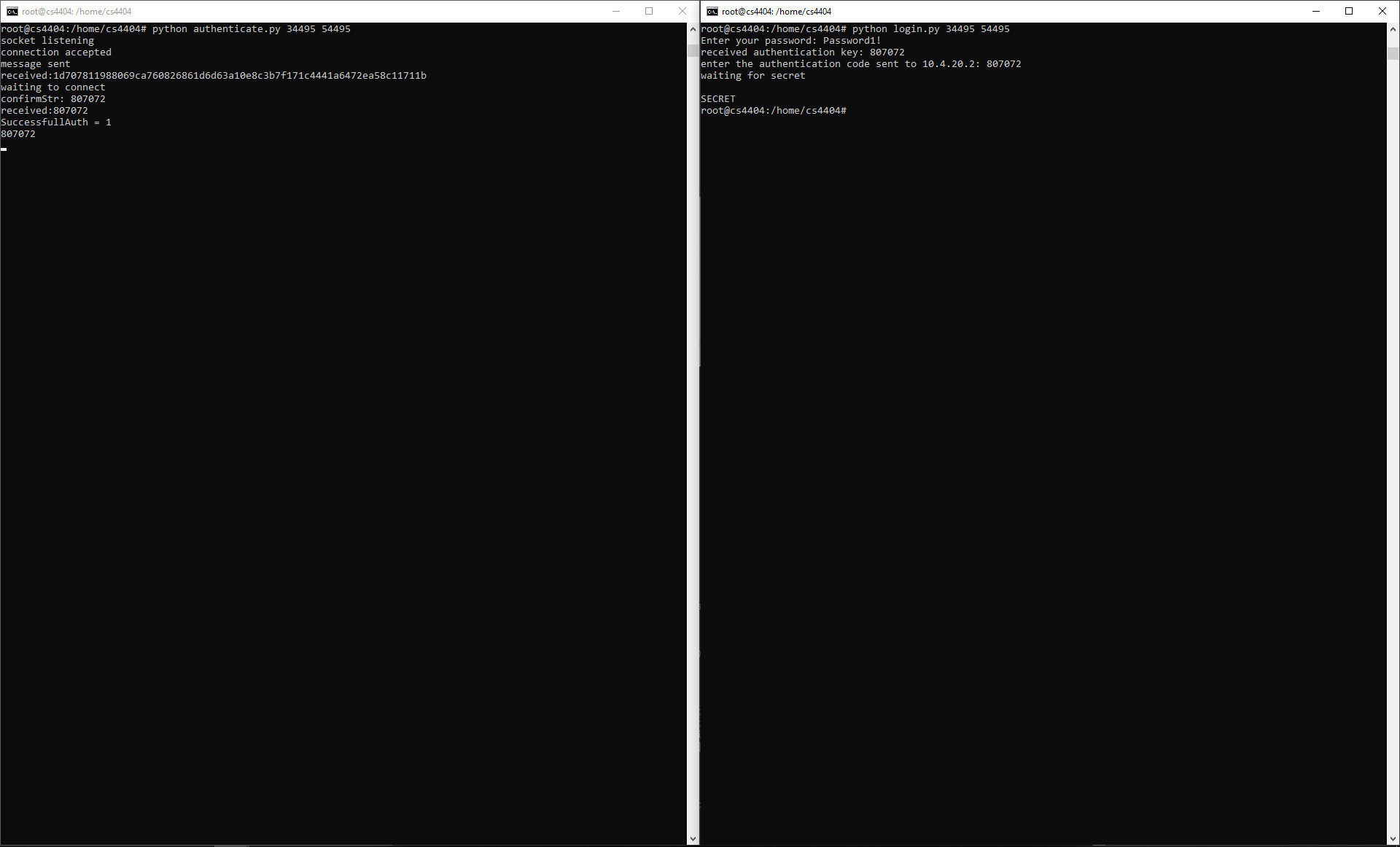
Figure 1. Normal Log in using our implementation of the infrastructure  


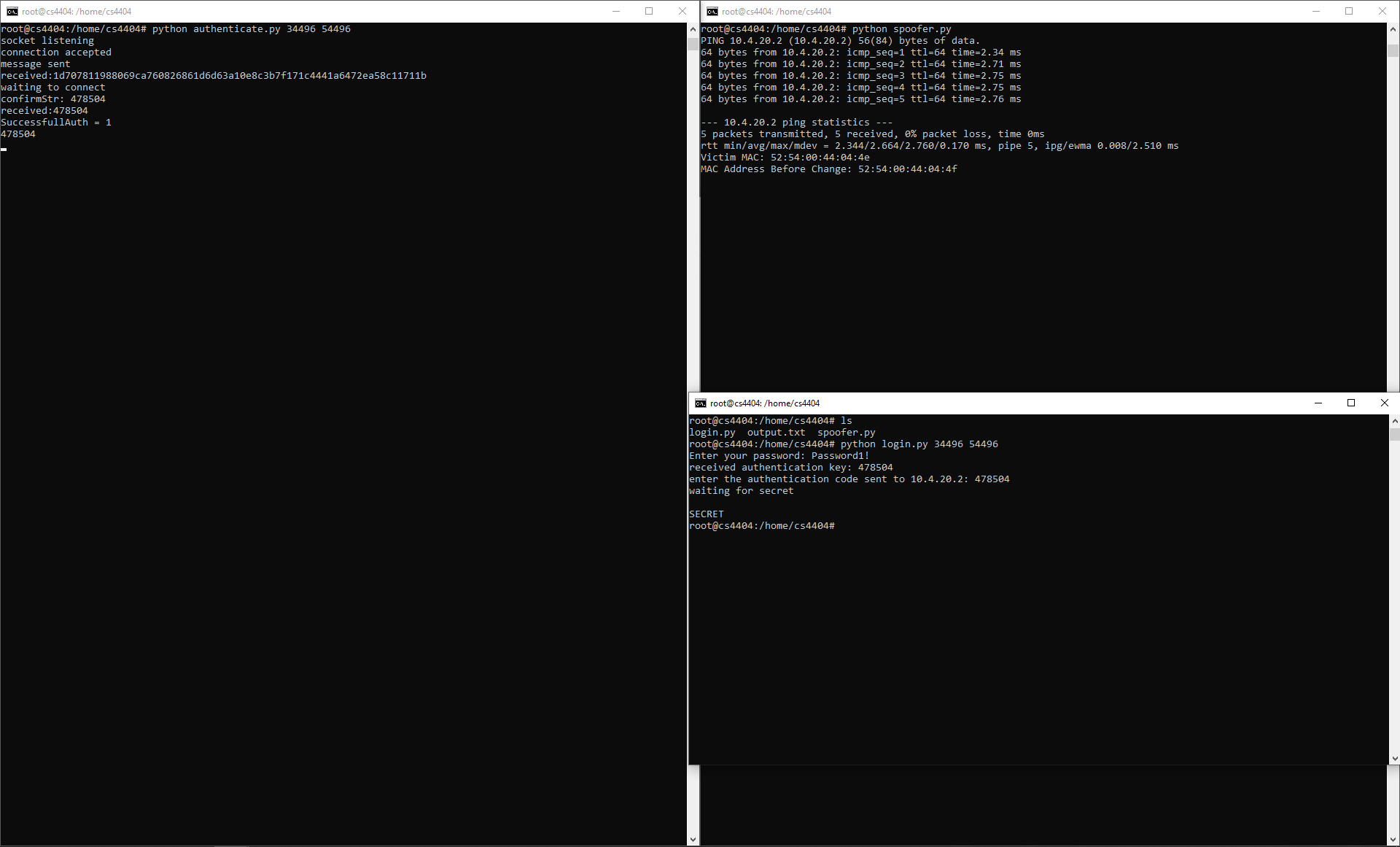
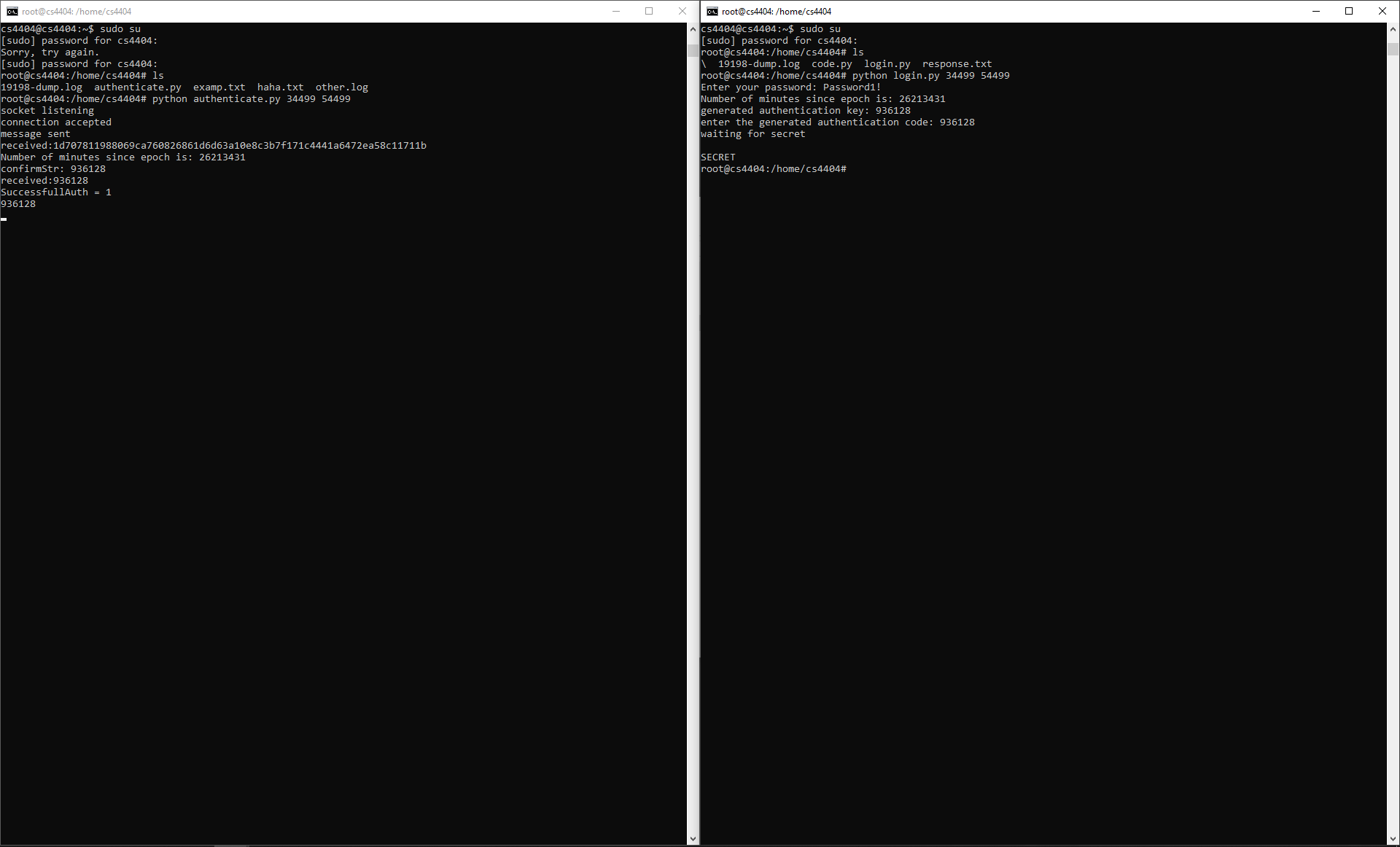
Figure 2. Log in with our implementation of the attack

Figure 3. Log in with the defense implemented simulating two factor authentication app  


**References**

ARP Spoofing. (2019, October 22). Retrieved from <https://www.veracode.com/security/arp-spoofing>.

Sean Reifschneider, L. (2013, March 2). Networking Basics: How ARP Works. Retrieved from <https://www.tummy.com/articles/networking-basics-how-arp-works/>.

Kaur, Rajwinder & Singh, Gurjot & Khurana, Suman. (2015). A Security Approach to Prevent ARP Poisoning and Defensive tools. International Journal of Computer and Communication System Engineering 2312- 7694. 2. 431-437.

3GPP. *A Guide to 3rd Generation Security*. 3GPP, 19 Jan. 2000, www.3gpp.org/ftp/tsg\_sa/wg3\_security/\_specs/33900-120.pdf.

Hoffman, C. (2017, July 3). Why You Shouldn't Use SMS for Two-Factor Authentication (and What to Use Instead). Retrieved from <https://www.howtogeek.com/310418/why-you-shouldnt-use-sms-for-two-factor-authentication/>.

Hummer, L. (2016, September 7). What's Wrong With SMS Authentication? Retrieved from <https://securityintelligence.com/whats-wrong-with-sms-authentication-two-ibm-experts-weigh-in-on-the-nist-recommendation/>.

NIST Special Publication 800-63B. (n.d.). Retrieved from <https://pages.nist.gov/800-63-3/sp800-63b.html#biometric_use>.

Starbug, CCC. “Hacking the Samsung Galaxy S8 Irisscanner.” Home, Media.ccc.de, 23 May 2017, media.ccc.de/v/biometrie-s8-iris-en.

Apple. “About Face ID Advanced Technology.” *Apple Support*, 29 Oct. 2019, support.apple.com/en-us/HT208108.

Shumate, Kaarin, et al. “Control Access Based on Network Location or App.” *Microsoft Docs*, Microsoft, 19 July 2019, docs.microsoft.com/en-us/onedrive/control-access-based-on-network-location-or-app.

Kaufman, M. (2019, September 26). IP Range Based Authentication. Retrieved from <https://www.servicenowelite.com/blog/2019/9/25/ip-range-based-authentication>.

Cardenas, E. D. (2003). MAC Spoofing--An Introduction. Retrieved from <https://www.giac.org/paper/gsec/3199/mac-spoofing-an-introduction/105315>