A Combined Encryption Method for transferring data between the smart TV and TV's server

Bohan Zhang, Anirruth Ragav, Richard Yeung, Xiaokang Jiao, Donggeun Lee

Abstract— **In these years, more and more devices are being connected to the internet, including the television. Although we are welcome to the new changes, few of us consider the possible risks. As an article from Norton mentions: “Smart” televisions come with a host of privacy risks[1]. These security and privacy issues need to be solved. However, current Smart TV manufacture focuses more on the media they provide instead of security protection. In order to build a solution to protect users’ private information, our group design to use AES encryption to encrypt the data transferred between the TV and the TV manufacturer's server. Besides using AES for data encryption, we also add an HMAC tag with SHA256 hashing to make sure data is well protected from being hacked. And we will check it on the server side to see whether the tag is tampered from the data transferring process.**

***Keywords*— Smart TV, Privacy risk, Data transfer, AES Encryption, HMAC, Sha256 Hashing.**

INTRODUCTION

Like a smartphone, a smart TV provides a number of “Internet-based services” that normal TV does not have. It has the equivalent of a “computer” built into it which is to provide apps, media streaming, web browsing, games and, perhaps most importantly, Internet Protocol Television(IPTV). IPTV can provide live video streams from the internet. For example, if you download a video from the net, IPTV can directly access the content without having a copy to the disk. Potential privacy issues come up as customers access media from the smart TV. For example, a smart TV usually has a webcam, the hackers could hack the webcam and see if there is anything worthy to steal in your home. Even for the smart TV that does not have a webcam, the company who provide you content will track everything you do. Hence, users’ privacy should be protected while they use the smart TV. For example, a report written by Catalin Cimpanu from Bleeping Computer mentions that a backdoor could be used to run IoT DDoS botnet. Smart TV can be used as a relay point to spy on users via the TV's microphone and camera[2]. The solution to this problem is to encrypt the data being transferred. Only the client and the server side can get the real content of the data being transferred by encryption and decryption. The hackers, without knowing the encryption and decryption method, will be difficult to hack the smart TV.

Obviously, the encryption method should be complicated and hard for the others to hack. This is the basic goal of protecting the data transferred from smart TV to its’ server. A possible solution is to use more secure API structure like Repy API which is developed by Cappos Justin, et al. Repy APIs has its own security layer built in which could o mitigate potential security risks[3]. However, adding a more complicated API structure layer would cost a redesign of the system that would lead to hiring more engineers with the increasing cost of the TVs. If encryption on data transferred is considered to be the solution, Matt Blumenthal mentions that RSA, as an encryption method, is more computationally costly since it requires long keys to make it safe[4]. AES encryption, which runs faster for encryption, also has its drawbacks. As Dawood mentions in his paper, most AES algorithm focuses on increasing block size or number of rounds to increase the security level, which is not considered as the best solution for the development experiments [5].

In this paper, we discuss an efficient solution which is to use AES as the main encryption method on the client side and then use HMAC to generate a tag using SHA256 hash. Instead of adding rounds or block sizes, we use HMAC to enhance protection on the data[5]. After data is being sent as a packet to the server, the server will check the tag to see if data is tampered. So even the data is hacked, at least the users will get the notice based on the transferring process. After making it sure that the data is not hacked, the server side uses the same AES and secret key to decrypt the data and get the original data.

BACKGROUND

The best solution to any kind of security problem always falls upon the degree of trust which the user places upon the companies that make the product. Even if the company that makes the TV's say that they have accounted for every variable, every possible point of attack when they developed the communication systems and their televisions. However, as is often the case, even the most sophisticated security always has a vulnerability somewhere. Companies don't always work with the idea that no product or technology is 100% infallible.

Since smart TV's are a relatively recent phenomenon which was developed with the rise of operating systems like Android and ios, researchers are still running various diagnostic tests on the most popular smart TV's produced today from companies like Roku, Samsung, Sharp, LG, Sony etc. While it is possible to turn your inherent normal television into a smart television through using Amazon's Firestick, Google's ChromeCast and Apple TV, the smart TV's come built in with an operating system, NIC's built into them. And like any device connected to the internet, there are a variety of ways to tamper their usage through the sue of various malware.

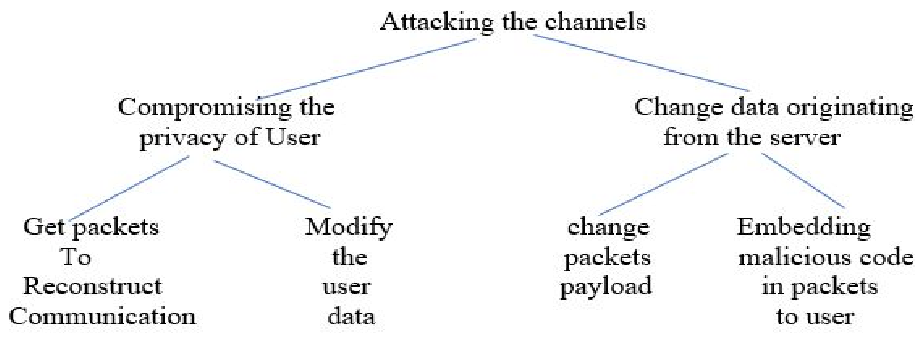
In a series of in house testing conducted by researchers from Norton[1], it has been found that it is possible to raise the volume up and down without using a remote, disconnect the TV off the wifi network and open inappropriate content on Youtube, rapidly cycling through channels, in TV's manufactured by Samsung. However, even more concerning was devices and software developed by Roku, where inherent flaws were found in the API's developed by Roku which were in turn used by other manufacturers like Sharp, Hisense and LG.

Also, in a hackathon conducted by the CIA and MI6, computer experts were able to hack into some of the most commonly used smart TV's and able to use them as listening devices. This after the PRISM and the Five Eyes programmes, furthers the needs to investigate and close vulnerabilities that occur in these smart devices.

In this paper, we have not proposed solutions to the backdoors or vulnerabilities that have been found inside the smart TV, but securing the transmission between the TV and the company servers. There might be data collected by the TV to be sent back to the company for knowing about consumer taste, and also you have various app stores built inside that communicate with the company's servers. We have proposed a solution to secure the communication channels between the company's servers and the TV. We use a combination of AES to encrypt the channel and HMAC to verify the integrity of the data that is being transferred.

THREAT MODEL

Now that we have established that we are securing the communication channels with the use of AES encryption system combined with HMAC, it is time to construct a tree and look how the system works under review addresses them.


Attacking the channels
Compromising the privacy of User
Change data originating from the server
Get packets To Reconstruct Communication
Modify the user data
change packets payload
Embedding malicious code in packets to user

Communication channels are commonly attacked in order to listen on to conversations without being spotted, or to deliver wrong information to the end user or to tamper with the server.

When the channels are not encrypted or uses weaker encryption systems, the data might be compromised. If packets are sniffed, the attacker does not know the encryption used and it will take a lot of time to figure out what is being transmitted to and from the user. If user data is modified, since we are using HMAC to authenticate the data from the server, kind of like CRC, the system will know that it's channels have been attacked and will raise a red flag.

In the same way, if packets payload is changed or if malicious code is embedded in the packets to user, it will fail the integrity check and the system would know that there is an attack.

DESIGN

Since our main goal is to protect the data transferred from server side and client side, we just need to enforce the protection during the process. Encryption of the transferred data is a good solution because it is both efficient and reliable.

Speaking of reliability in encryption, AES (Advanced Encryption Standard) is a proven encryption algorithm along with HMAC (hash-based message authentication code) would be a good combination for this solution.

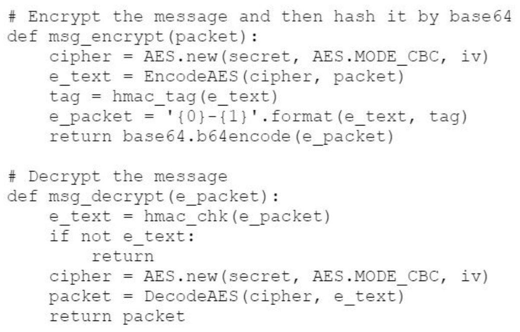
Specifically, AES, Advanced Encryption Standard, is a specification for the encryption of data created by the U.S. National institute of Standard and Technology to replace the outdated and insecure DES (Data Encryption Standard) in 2001. AES has much stronger algorithm than DES which could convert plaintext to ciphertext with much longer key length. If someone tries to break the encryption by brutal force, it would take very long time and it is known impractical to attempt such decryption. Since we are focusing on smart TV data transmissions instead of safe of banks, it would also be inefficient to use large amount of resources to obtain very little valuable information. So applying AES on the smart TV data could protect users' data confidentiality.

On the other hand, we use HMAC is to protect the integrity of the users' data. By using a hash function by HMAC, we generates a random hash that is very hard to invert. So we could use this as a tag for the data transferred. When it is time to verify the tag, HMAC can re-generate the tag and compare it with the received tag. Thus, the integrity of the data could be preserved. Even if the data changed a little bit, the hash will never look like the same according to Avalanche Effect[6] of good hash functions such as HMAC, which greatly decrease the margin of error.

For example, the "Red Button Attack" that happened in New York City in 2014 [7] could be prevented using combination of AES encryption and HMAC tagging. If the data is tampered during the process, the server side would determine the data sent by the client side invalid which eventually protected the confidentiality and integrity of users' data.

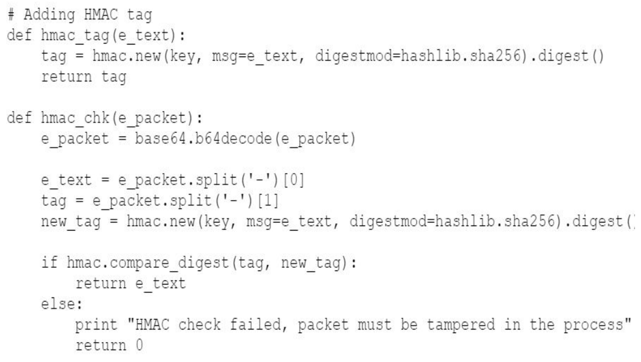
IMPLEMENTATION

We solve the smart TV problem by encrypting the data transferred between the TV and the TV manufacturer's server. We wrote a Python program to simulate the data transmission. In this program, we used the AES encryption that professor mentioned for the crypto lecture so that our data is hard to be decrypt. The role of the RPC server in the Python program is to provide services to both server and client sides using a unique initialization vector (IV, to make the encryption more random) and a secret key that only known by the RPC server. In other words, RPC is not the actual TV manufacturer's server, instead it serves as a utility just to process requests instead of doing the data transmission.



```
# Encrypt the message and then hash it by base64
def msg_encrypt(packet):
    cipher = AES.new(secret, AES.MODE_CBC, iv)
    e_text = EncodeAES(cipher, packet)
    tag = hmac_tag(e_text)
    e_packet = '{0}-{1}'.format(e_text, tag)
    return base64.b64encode(e_packet)

# Decrypt the message
def msg_decrypt(e_packet):
    e_text = hmac_chk(e_packet)
    if not e_text:
        return
    cipher = AES.new(secret, AES.MODE_CBC, iv)
    packet = DecodeAES(cipher, e_text)
    return packet
```



```
# Adding HMAC tag
def hmac_tag(e_text):
    tag = hmac.new(key, msg=e_text, digestmod=hashlib.sha256).digest()
    return tag

def hmac_chk(e_packet):
    e_packet = base64.b64decode(e_packet)

    e_text = e_packet.split('-')[0]
    tag = e_packet.split('-')[1]
    new_tag = hmac.new(key, msg=e_text, digestmod=hashlib.sha256).digest(

    if hmac.compare_digest(tag, new_tag):
        return e_text
    else:
        print "HMAC check failed, packet must be tampered in the process"
        return 0
```

Figure 1, Encryption and Decryption **Figure 2, HMAC adding and checking**

To explain the implementation in details, in figure 1 on client side, it acquires the encrypt function from the RPC server and encrypt the hidden message using the secret key and the IV that only the RPC server knows. Then we used HMAC (hash-based message authentication code) to generate a tag using SHA256 hash from figure 2. Lastly, we sent the tag along with the encrypted message as a packet to the server side(not the PRC server, the actual data receiver end server).

For the receiver server side, it receives the packet. Then it re-calculate the tag of the encrypted text and at the same time, compare it with the original tag. If the tags are the same, then it shows that at least the packet is not tampered during the data transmission. If the tags are not the same, then the packet should be dropped. After the verification of the tags, the server side will get decrypt function from the RPC server then decrypt the message using the AES with IV and secret key only known by RPC server. Lastly, the server side would get the original message sent by the client side.

EVALUATION

Since our method of implementation relies on the data being encrypted before being sent out, once it is received on the other end, the data can be easily checked to see if it has been tampered with by comparing the HMAC tag with the original tag. Our method either ensures that the information will be confidential and that it will not be read or fixed while being transferred to the other end or it will be read or fixed and the other end will know that it has been tampered with and can then discarded and alternate methods can be then used.

Based on the sample code that was developed, the implementation of this code onto real world smart TVs and devices can be feasible. The cost of integration would be low and the amount of resources would be low since the code handles the private data transfers.

RELATED SOLUTIONS

The earliest types of encryption were very simple. However, as there become more hackers and they got better at cracking codes, the encryption had to become more sophisticated so that the data could be kept secret and secured. Now, AES encryption method is standard and one of the most best security method. Here is brief explanation of how AES encryption works. The entire AES encryption process goes: key expansion, add round key, byte substitution, shift rows, mix columns, add round key, then byte substitution, shift rows, and add round key. These are the basic steps of AES encryption. AES has key sizes of either 128, 192 or 256-bits, and when 128-bit key is used, there are nine of these rounds, when a 192-bit key is used, there are 11, and when a 256-bit key is used, there are 13[10]. Therefore, the data goes through the byte substitution, shift rows, mix columns and round key steps up to thirteen times each depending on the key size, being altered at every stage. 128-bit AES is good enough for most practical purposes, and highly sensitive data should probably be processed with either 192 or 256-bit AES.

Here are some other types of encryption methods. One of them is called Triple Data Encryption Standard, 3DES. It is a block cipher. 3DES is a symmetric-key encryption that uses three individual 56-bit keys. It encrypts data three times, meaning 56-bit key becomes a 168-bit key[11]. However, it processes slower than others, and because it uses shorter block lengths, it is easier to decrypt and leak data.

Another one is called Twofish, which is a symmetric block cipher based on an earlier block cipher, Blowfish. It has a block size of 128-bit to 256-bit, and it works well on smaller CPUs and hardware[11]. It implements rounds of encryption to turn plaintext into ciphertext just like AES encryption method, however, the number of rounds doesn't vary as with AES.

CONCLUSIONS

In this paper, we developed a sample program that mimics our methodology of handling data being transferred between a smart TV and the TV's server. The solution is based on the AES encryption that was mentioned by professor Cappos in his lectures and powerpoint slides. Briefly, how the solution generally works is that from the client side, which is the smart TV, it encrypts the message or data being sent and uses a secret tag that only the server is being sent to knows. Once the data is sent over, the server checks the data by looking at the tag and comparing it with the original tag created before to see if the data has been tampered with. Overall, our method seems to be feasible and with more research or explorations, this solution can be a possibility in securing information transfer between a smart TV and the TV's server.

REFERENCES

[1] Norton, “What is a smart TV and what are the associated risks” https://us.norton.com/internetsecurity-iot-smart-tvs-and-risk.html.
[2]Catalin Cimpanu, “Your smart TV is watching you watching TV, Consumer Reports finds”, BleepingComputer, March 29, 2017 https://www.bleepingcomputer.com/news/security/about-90-percent-of-smart-tvs-vulnerable-to-remote-hacking-via-rogue-tv-signals/.
[3]Cappos, Justin et al. “Detecting latent cross-platform API violations.” 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE) (2015): 484-495.https://ieeexplore.ieee.org/document/7381841/metrics#metrics.
[4]] M. Blumenthal, “Encryption: Strengths and Weaknesses of Public-key Cryptography,” dalam Computer Science Research Symposium, Villanova, 2007. http://www.csc.villanova.edu/~mdamian/Past/csc3990fa08/csrs2007/01-pp1-7-MattBlumenthal.pdf.
[5]Dawood, Omar A. & I Hammadi, Othman. (2017). An Analytical Study for Some Drawbacks and Weakness Points of the AES Cipher (Rijndael Algorithm). 10.25212/ICoIT17.013. https://www.researchgate.net/publication/316167904_An_Analytical_Study_for_Some_Drawbacks_and_Weakness_Points_of_the_AES_Cipher_Rijndael_Algorithm.
[6]Wikipedia, “Avalanche effect”, https://en.wikipedia.org/wiki/Avalanche_effect
[7]Upbin, B. (2014, June 17). Red Button Flaw Exposes Major Vulnerability In Millions of Smart TVs. Retrieved from https://www.forbes.com/sites/bruceupbin/2014/06/06/red-button-flaw-exposes-major-vulnerability-in-millions-of-smart-tvs/#7f723c7c20d9.
[8]”Which Types of Encryption are Most Secure?,” https://www.toptenreviews.com/software/articles/secure-encryption-methods/.
[9]Josh Lake, “What is AES encryption and how does it work?,” comparitech, Oct.05.2018, https://www.comparitech.com/blog/information-security/what-is-aes-encryption/.
[10]“What Is AES Encryption (with Examples) and How Does It Work?” *Comparitech*, 5 Oct. 2018, www.comparitech.com/blog/information-security/what-is-aes-encryption.
[11]“5 Common Encryption Algorithms and the Unbreakables of the Future.” *StorageCraft Technology Corporation*, StorageCraft Technology Corporation, 3 Oct. 2016, blog.storagecraft.com/5-common-encryption-algorithms.