Radio-Frequency Identification Exploitation

Max “RIVAL”
www.siteofmax.com
rival@riseup.net

Abstract

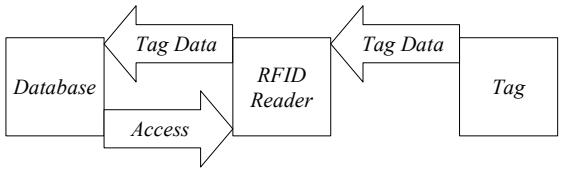
The aim of this paper is to discuss the principles behind Passive Radio-Frequency Identification and the principles behind the exploitation, as well as the practical methods that can be used to exploit RFID.

1. Introduction

RFID is everywhere. Most buildings that require people to authenticate themselves will use Radio-Frequency Identification in order to do so. A lot of modern debit cards use it for quick and easy checkout at shops. Even travel cards and lots of passports use them. You would expect a large amount of consideration when it comes to the security of such an authentication method, however this is not the case. As you can imagine, RFID is of an interest to hackers for this exact reason and although many people are aware of the security risks associated with it, very few vendors have designed a secure method of using RFID for authentication without the risk of a hacker eavesdropping or spoofing another individual's data. Once you understand exactly how RFID works, you'll understand why it's so simple to exploit and use for an attacker's advantage.

2. Principles of Passive RFID

RFID, as the name suggests, uses radio waves to transmit data. There are two parts to a Passive RFID system; a tag, and a reader. Passive RFID tags, unlike active RFID tags, do not have their own power source, such as a battery. In fact, they remain completely powerless until they go within proximity to a source of power such as an Active RFID reader. RFID tags contain two parts. These parts are the Antenna and the RFID Chip. Active RFID Readers emit electromagnetic waves which are picked up by the RFID tag's antenna. The tag's antenna uses the collected energy from the wave to power the chip. When the chip receives power, it modulates a Radio-Frequency signal with the data contained within the RFID chip. The signal is picked up by the reader and the information is checked for validity. Most of the time, this is done through an internet database or on a local database synchronized from an external source. Below is a diagram of this system.



Above: RFID Diagram

3. Principles of RFID Exploitation

Exploiting RFID is not too far from how RFID normally works. The concept is simple; all we need to do is read the tag's data. To read the data we need an RFID reader. A long range RFID reader can be built with an Arduino Pro Micro 5V or bought pre-built. Once this is done, it is possible to use an RFID writer to write to your own RFID tag and create a tag clone. This is known as a Replay Attack. However, there are things to consider when doing this:

- If the target's tag has both read and write capabilities then protection via a token may be a possibility. An example is where a person's tag is read by the reader and the token sent is then changed and the current token is overwritten on the tag.
- Some tags also have a built in method that allows them to be shut down if the manufacturer or owner wishes.
- Remember that a lot of the time, data sent from the tag is actually secured via RSA Encryption or a Rolling Code.

For a full view of RFID exploitation, we can't just look at the RFID tag itself, we have to look at the

reader. RFID readers have been known to suffer from Denial of Service Vulnerabilities and even Buffer Overflow Vulnerabilities that can be used to execute code on the system connected to the RFID reader.

Denial of service attacks can be easy to execute. The aim of a DoS attack on an RFID system is usually to disrupt service to a target or a group of targets [1]. A simple example of an RFID attack is to attach a non-working RFID tag's chip to the side of an RFID reader, thus permanently creating an error. This means that when a working RFID tag comes within proximity to the reader, the non-working chip is still causing the reader to think that something's wrong. This can also be known as RFID Collision.

The main vulnerabilities that we will cover during this paper will not be to do with disrupting service, but instead to exploit the computer behind the RFID reader.

4. RFID Website Exploitation

Some RFID tags contain URLs to user pages that are checked via a web browser on the computer instead of directly via a database. The web browsers used to view the URL result may be vulnerable to browser exploits or code injection. For example, getting the browser to go to a web page containing exploit code could be as simple as setting the tag's data to *<script>document.location='http://exploit/';</script>*

Other RFID tags contain data that is used as an input into a web page. For example, the URL where the user's tag data, or in this case, their user key, is input could be something such as *http://site/check.php?usr=user_key* which could be vulnerable to XSS. The XSS vulnerability could be used to attack the RFID reader's machine via client-side Javascript injection. Remember that SSI may also be used to exploit a server from the web interface by executing shell commands.

5. RFID SQL Injection

Databases are the main method of checking whether the user data is correct or not. Of course, because of this, it means that SQL Injection could be a problem. The problem arises because the input is not properly sanitized. For example, *SELECT id FROM users WHERE key=%user_key%* could cause an error if the tag's data was edited from *mark20* to *mark20'* because the apostrophe means the final query would be *SELECT id FROM users WHERE key='mark20''*. This alone would produce an error, however, changing the string to be *mark';* would allow you to insert your own code after the previous query in order to access the database however you choose.

6. Buffer Overflows

Buffer Overflows are the single biggest problem when it comes to operating system exploitation. They occur when more data is read than expected. In this case the problem is that the program that interacts between the RFID reader and the software has a buffer for a very limited amount of data. This size is usually determined by information written on the tag. However, the data that will be read could exceed this amount and thus result in a buffer overflow. If we know how much data is required to fill the buffer, we can append additional data that can allow us to execute code on the target machine. For example, once we have filled up the buffer on the tag, we can append the return address as the address +4 in order to jump to the data held in the stack (that we want to execute from). Next we can push the address of the command we want the system to execute to the stack and then call the system function to execute it. Following this should be the command data with an appended null byte (The address of this is the address that you push to the stack). Remember that the bytes on the tag will generally be in reverse order due to the Little-Endian architecture.

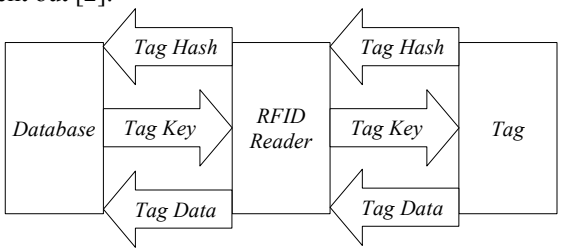
7. RFID Malware

Malware is the most common threat to computer users. However, when the item that is infected is of something such as a bank database, it becomes a much bigger problem. RFID malware is possible because of databases the readers are connected to. This in itself links in with what we learned in the SQL Injection section. SQL Injection isn't just used to get information on another user but can be used for executing commands from the reader's computer. Buffer Overflows, as mentioned before, can do this however you may only have database access. The *EXEC* command comes in handy for this use. An example of executing shell commands from the database would be *EXEC xp_cmdshell 'dir';* which will create a listing of files on the computer. Knowing this, you can assume that it is now possible to use other tools on that computer. For example creating a new user on the system and connecting to it via Remote Desktop Connection. If the computer running the database is using Linux, it is also possible to execute the *wget*

command in order to download a file from an external file server.

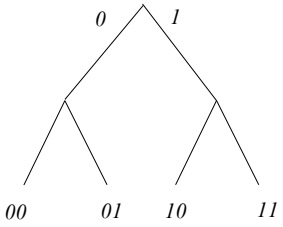
8. Countermeasures against attackers

1. Data Locking – Data Locking works by the tag not sending it's real data until the reader provides the correct hash associated with the tag. Simply, the tag first sends out a hash that is detected by the reader. The reader obtains the hash and compares it to a database of hashes and keys. A key is then sent back from the reader to the tag and if it the associated key is correct, the tag sends it's real data out. This works because anybody scanning for RFID tag data will only receive the tag's hash and when the tag does not receive a response back containing a correct key, the data is not sent out [2].



Above: Data Locking Diagram

2. Singulation (Selective Blocker) – Singulation protocols are used to combat the RFID collisions mentioned earlier that can cause Denial of Service problems. A common protocol for singulation is the Silent Tree Walking Protocol [3]. Instead of the database being stored in tables and rows, it is stored as a tree of 1s and 0s. The tag's data is read bit by bit by the RFID reader and the computer system behind it traverses the tree. When multiple tags are within the range, the RFID reader queries all the tags about the next bit in the pattern. For example, there may be two tags that start with 0. The reader may ask which tag starts with 01 and one of the tags may respond. A problem with this method is that the RFID reader can leak a considerable amount of information on the tag, however a researcher at MIT has produced a modified form of the silent tree walking protocol that solves this problem.



Above: Tree-Walking Diagram

3. Encryption – As mentioned before, encryption is the most obvious way to secure data between the tag and the reader. This can be done well with an asymmetric or symmetric key algorithm. For example, the tag sends the reader it's unique ID. The reader then sends a random number to the tag, encrypted with the tag's unique encryption key. The tag then decrypts it with the other key (asymmetric) or the same key (symmetric) and then sends it back to the reader. The reader then checks that the decrypted number is the original number and allows the tag's ID to be authenticated. The reason this works is because if an attacker were to intercept the tag's ID, they would not be able to use it without obtaining the tag's private key and the only way for an attacker to do that would be to physically obtain the tag and disassemble it in order to read the area of memory the private key is stored in.

9. Conclusion

In this paper we have shown that Radio-Frequency Identification can be a larger security risk than most corporations and individuals realize. Even the methods of protection such as Data Locking can still allow direct access to the database, meaning that SQL Injection and remote command execution is definitely still a possibility. Not only can these vulnerabilities be a problem for the systems behind the RFID readers, but also a risk for the users who have their private information stored within that database. The most important point in this paper is that RFID exploitation has been shown to be a broad subject, and one that we have only just touched upon. Hopefully, in time, RFID vulnerabilities will become an obsolete problem for users and companies.

10. References

[1] MohammadHassan Habibi, Mahmud Gardeshi, Mahdi R. Alaghband, "Practical Attacks on a RFID Authentication Protocol Conforming to EPC C-1 G-2 Standard", *International Journal of UbiComp (IJU)*, January 2011, pp. 1-3.

[2] Vaibhaw Dixit, Harsh K. Verma, Akhil K. Singh, "Comparison of various Security Protocols in RFID", *Internation Journal of Computer Applications (0975 - 8887)*, June 2011, pp. 3-4.

[3] Smart Border Alliance, "RFID Security and Private White Paper", pp. 8