SYSTEM UPDATES: ATTACK AND DEFENSE

Sofiane Mohamed Talmat

sofiane.talmat@ioactive.com

This is a **HIGH LEVEL FIRST DRAFT** version that is provided to EKOPARTY reviewers more details addition and rewriting will be done

More examples and technical details about each vulnerability and CVE listed with illustrations will be added in each category of vulnerabilities for the final version, same as for the LESSON LEARNED and CONCLUSION in order to avoid system updates vulnerabilities

***Abstract*—From device firmware to full complex operating systems, system updates are critical to maintain an up to date version of the running software, providing security patches and fixes for vulnerabilities, however many update and upgrade systems contain vulnerabilities that could make things go wrong.**

In this paper we will not only dissect in details some existing system update vulnerabilities, we will also deep dive into common vulnerability concepts discovered during this research and previous work, we will describe different attack scenarios and approaches and how this could lead to the whole system subversion.

We will also talk about both common design and technical mistakes and best practices on how to design secure system updates and upgrade for both devices and software.

***Index Terms*—Update system, software, mobile, Device, firmware, vulnerability, privilege escalation.**

INTRODUCTION

System updates is a critical process that runs at a higher privilege level than that of a regular user.

During a pentest, the penetration tester could need to escalate his privileges and exploiting system updates may be one of the ways to make things happen, as in many cases the updates are scheduled as a privileged user or the simple user can start the process by running a specific binary that will use one or more techniques in order to escalate its privileges in a specific way or hand over the update to a privileged service in order to finish the update.

In both cases the attacker being able to exploit an update process will be able to elevate his privileges and run code as the privileged service processing the update.

In this paper we will describe with examples common mistakes within system updates and key areas to look at during a pentest and how to approach system updates in order to conduct a successful penetration test by escalating user privileges to administrative ones.

We will go point per point and describe general vulnerabilities categories providing the reader with detailed examples on previous disclosed vulnerabilities and real life examples faced during engagements that led to privilege escalation.

VULNERABILITIES

1- Race conditions

Many update systems rely on downloading remote files either by:

- Downloading an executable update file
- Downloading a configuration file containing update locations.

In some cases this could lead to race conditions in cases where the destination folder is writeable by the user.

For example in the Lenovo Race condition vulnerability (CVE-2015-2234) a binary is downloaded to a world writeable folder "c:\programdata\lenovo\SystemUpdate\newclient" then run as Administrator.

An attacker could exploit this execution flaw by writing a small program that overwrites the newly downloaded executable at a critical time between the saving of the file and it's execution and fool the system update process to run the attacker's binary under Administrative privileges.

There are a lot of other situations where a configuration file (XML for example) is downloaded containing information and paths of update packages that that could be ZIP, CAB or other executable in order to be downloaded and extracted or run for updates purpose, in some cases those configuration files are downloaded in a world writeable folders, in many cases they are under user's temp folder, so in case an attacker is able to rewrite those configuration files, he can fool the update process that there are new packages needed to be installed and point the update

process to download and run malicious packages with high privileges.

2- Weakness in communication process

In order for a user to manually run system updates in a locked down environment, there is a scenario in which a communication between the binary run by the user and a pre-installed higher privileged service will take part in order finish the update process. Taking the example of the Lenovo vulnerability (CVE-2015-2219), when a user starts a manual system update, a binary is downloaded from the remote system, and since the user has no administrative privileges, the update process will initiate a pipe communication to a service running SUService.exe as SYSTEM user. The update process will ask the service to finish the system update by running the downloaded binary with higher privileges.
By exploiting any weakness within the communication, an attacker can fool the service and ask it to run a malicious binary with higher privileges.

In the case of Lenovo vulnerability (CVE-2015-2219), the SUService service is listening on a named pipe and waiting for client communication, however the software contains two major vulnerabilities. The first one is that the high privileged service relies on a hardcoded key that should be provided by the client through the named pipe in order to validate the authenticity of the caller prior to processing the command. The Service also receives a command line containing the path to the binary that should be executed as well as other arguments and parameters. One such argument is /securitycode which should contain the security code that is used to validate the authenticity of the caller, because this value is hardcoded in the software binaries, an attacker could extract it and build his own program that will just connect to the defined named pipe and send a command line to the service providing as argument "/securitycode xxxxxxxxxx". This ends up fooling the Service that it's being called by a legitimate update binary.

3- Use of relative path

Generally in a locked down environment,only Administrator has write access to the software folder in order to protect the integrity of the update system from low level users. However in many cases it is possible to copy the software folder to a new writeable location from where you can manually run the update process.

In many cases the binaries use a relative path looking for executables, DLLs, configuration files and so on.
By copying the folder to the desktop for example, it is possible to fool the application into loading modified files.

In many cases I was able during different assessments to use this technique to perform a dll hijack and force the binary to bypass specific restrictions or execute code within the hijacked DLL with higher privileges.

4- Weak or lack of Digital Signature validation

In the previous vulnerabilities, things are much more easy to exploit when there is no proper digital signature validation. It will be easy to patch the binaries in order to bypass digital signature validation.
For example in both Lenovo vulnerabilities concerning Race Conditions and weak communication categories, the downloaded binary provided to SUService should be signed by Lenovo. However, the other vulnerability about bad signature validation (CVE-2015-2233) makes things exploitable as the validation is done by just verifying the subject of the digital signature and not the validity of the signature.
So by creating a self signed certificate with the same subject as the valid Lenovo Signature and self sign the binary, the check will be bypassed and the service will run the binary assuming that it's actually signed by Lenovo.

5- Weak or lack of proper certificate validation

In general, the communication between the host and update server is done through secured protocols. However in some cases the certificate trust chain validation is not done properly and an attacker can redirect the hostname to a service with a self signed certificate that will fool the downloader into trusting the remote system and downloading and installing fake updates and binaries on the system.

In cases of bad certificate validation or ignoring certificate errors, a successful attack could be critical on the target system. In many engagements I used to use a self signed certificate within a web server that was able fool the client and install desired binaries on the remote system.
Another "Feature" from the developers that an attacker could exploit is the Failover process. I've been seeing during many engagements that developers do fail to a clear text HTTP connection in case the HTTPS one fails. A developer may think that in a critical update it is better to go back to a HTTP connection in case the HTTPS port is not reachable, however an attacker could exploit this by closing or dropping HTTPS connections from a network or router perspective forcing the application to go back to clear text HTTP connection that can be easily targeted by a MiTM attack.

6- Network attacks and clear text communications

A lot of software update system are still using clear text communication protocol and most of the cases I've found using clear text communication and mostly using FTP servers are hardware devices pulling firmware updates from remote FTP servers which will open door to different kind of network attacks including MiTM attacks .
One of the other examples of clear text protocols is a vulnerability that was reported to Fujitsu about their update Agents, as the agent uses a SOAP communication over clear text HTTP protocol. The account used for authentication is transferred encrypted

using a custom encryption algorithm, by reversing that simple algorithm it was possible to extract the windows domain and the login information from sniffing clear text HTTP communication, so anyone able to sniff a the clear text communication will be able to decrypt and extract high privileged windows domain accounts.

A nice and interesting tool called evilgrade written by Francisco Amato can be used for exploiting different network attack scenarios within different vulnerable software. The tool can be found under "https://github.com/infobyte/evilgrade"

7- File unpacking attacks

Many developers are still using custom code to unpack and unzip files, however they may fail in validating and parsing files within archives. So for example by extracting a zip file containing a filename with a directory traversal pattern, an attacker could corrupt the system by writing or overwriting files on the remote system.
It is actually pretty simple to create a corrupted zip file with a path traversal in it. This could lead to an escape from the restricted download folder to write files outside on the system.

During one of my previous engagements I was able to spot a similar case within a mobile application in which the zipped update file was downloaded within the mobile download folder and then extracted after validating the digital signature.
The first vulnerability consisted in a race condition where it was possible to change the ZIP file between the time of the validation of digital signature and the extraction. The second vulnerability was about the extraction process where in case of a successful race condition exploitation, it was possible to write arbitrary files on the mobile file system using a path traversal vulnerability within the modified zip file.

Another known attack within zip files is called ZipBomb that could lead to a DOS and consists of creating a specific zip file that is only some Kilobytes in size and when extracted it will lead to many petabytes. An example of such file is the famous 42.zip file.

8- Unattended installation log files and memory dump

Within the projects I conducted, I faced some locked down systems where unattended installation and update log files leaked both local and remote system credentials as secret keys.
It is actually common that update systems store sensitive information within log files or error files in case the update is facing a problem or a crash.

On the other side it is possible to disclose sensitive information by attaching to the user process or dump it's content from memory, many credentials or sensitive keys could be disclosed.

IMPACT

Since Update systems mainly goes through privileged users, exploiting those kind vulnerabilities could lead to an elevation of privileges on the system and this could be either locally or remotely.

Update systems is an interesting research area as it will reveal than many software are still suffering from lack in design or execution flow vulnerabilities as one single development error could lead to a critical finding as there is no place for Medium or Low risk once exploited.

LESSONS LEARNED AND BEST PRACTICES

<<under development, providing best practices in each of the above category and detailed technical approach on how to avoid the discussed vulnerabilities>>

REFERENCES AND FURTHER READINGS

- ……..
-