International Journal
Information Technology Applications (ITA)

Volume 14, Number 1, July 2025

CYBERSECURITY CHALLENGES ASSOCIATED WITH SOFTWARE UPDATES

Robert Valík

Abstract:

In the past, software distribution was one of the tougher problems for independent developers. With the rise and ubiquitousness of broadband internet connectivity, this problem has been solved as these developers have the ability to directly distribute software updates to their customers. As the focus is on fast-paced software delivery, they may lack practical knowledge in some other, more subtle aspects of software distribution, e.g. information security.

Our hypothesis is that there is a lack of security auditing before the final distribution. Using a controlled environment, we analyzed outgoing network traffic to identify the communicating during update sequence.

We used acquired data to identify a specific service that serves software update packages. After we identified the service, we tried to force the software to use our modified service, download a modified update package and execute it. As we were able to succeed, this outcome fully supports our above assumptions.

Keywords:

Software updates, security, vulnerability.

Introduction

In the last decades, the software development and distribution process has evolved considerably. In the past, before the abundance of broadband internet connectivity, the software distribution was handled by stone and mortar shops or postal services. One had to buy a physical medium containing the desired software. Delivery of regular patches was on the edge of unthinkable. The focus of the software development was to build and deliver a finalized working product. After the release the work would shift to building a newer version with a new functionality or a different product altogether.

After the emergence of the internet, new pathways for the software delivery became apparent. Potential clients may buy the software online, download it, install and use it nearly instantaneously [1]. If a problem with the software that warrants a new version arises, it is quite easy to download and apply the patch. Most modern applications contain a self-update mechanism that periodically checks if any update is available and install it.

As the updates on the top of new functionalities may contain fixes for newly discovered security issues, prompt installation of the software updates is considered a best practice [2]. But can we consider it to be safe in any situation? We will discuss some possible scenarios where it may have some undesirable effects.

This is an open access article under the CC BY-NC license
https://creativecommons.org/licenses/by-nc/4.0/

1 Hypothesis

The independent software studios or individual software developers have always existed, but the spread of the public / consumer internet access broadened the field for alternative software delivery methods and allowed the independent developers to thrive without the need to worry about the logistics of the distribution of physical installation media.

Our hypothesis is that as independent software developers focus mainly on software development and functional testing, there may be lack of attention to other parts of the software delivery life cycle. Security may be perceived as an unnecessary overhead and their quality assurance processes may have severely limited or entirely missing security auditing. As a result, many unpatched and overlooked security issues may exist. We also suspect that there may be a lack of understanding of the consequences of delivering insecure products on the creator's side.

In this work we will focus on one of the possible attack vectors on the update process.

2 Approach Description

Our intention is to show that some ways to influence a software update process with undesirable outcome for the software user do exist. To manipulate the process, we need to have some insight into it - how it works, where does it connect, which protocols the process uses. There are more ways to achieve this - direct reverse engineering or de-compilation of the software application or black-box testing [3]. We will apply the black-box approach.

The first step will be to passively capture the network traffic, identify relevant protocols, encryption and endpoints for transmission. The result may help us to identify some weak points in communication. There are a lot of tools available already, commercial and open source. We will use the open-source network analyzer - Wireshark.

After successful identification of the communication protocols used, we will try to identify and analyze the update payload, its structure and content.

After the analysis phase, if we gather enough information, we will proceed to the "attack" phase. In this phase we will try to create a modified update package, and we will try to force the application to connect to our specially constructed service endpoint, download the modified package and apply it.

Before we start, we have to select software to analyze. One class of software products that depend on rapid updates are - computer games developed by both large and independent game companies which rely on the recent "early access" paradigm [4]. Releasing a game as "early access" is in the essence - a commercial release of an unfinished product. Potential client pays for the promise that he will get a finished product in the future and while using the unfinished product, and that he may be a part of the development (you pay to be a beta tester of a game, and you can keep your license after the product is finished).

3 Analysis

After careful consideration as the first software to analyze, we have chosen a factory building game developed by an independent game studio. The game has been at the time of analysis under heavy development but quite stable and already available as early access. As we were unable to acquire the studio's consent at this moment, we will not disclose more information about the game and will refer to it as The Game.

3.1 Update Sequence Analysis

As the (Fig.1) shows, the update mechanism first uses DNS to resolve two addresses - updater.thegame.net and dl.thegame.net. After this step it uses a TLS encapsulated communication channel to connect to these two servers. After trying to connect to these addresses, it is apparent that the first is an authentication service and the second is an address of a content delivery network that caches and provides a worldwide accelerated access to the update packages.

The Transport Layer Security (TLS) [5], if implemented correctly, provides robust security over while communicating over other networks. It relies on cryptographic algorithms and if a party provides a cryptographic certificate, the other party can verify the counterpart's identity.

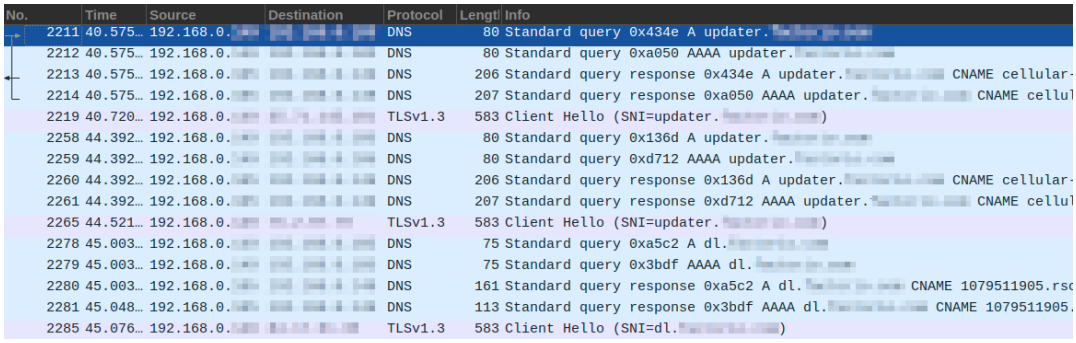

Fig.1. Network traffic captured using Wireshark - DNS queries and TLS connections to the update server are easily visible.

After the package is downloaded, it is applied to the game, the game is restarted automatically, and the player may enjoy it. The whole update flow is visualized on the sequence diagram on (Fig.2).

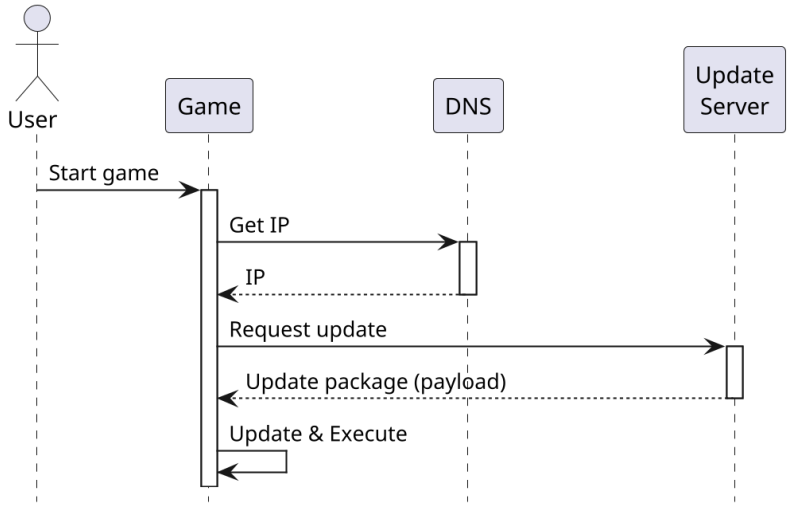

Fig.2. Communication flow during the update process.

To better understand the communication between the game and the update servers, we need to see what is inside of it, and how it works. The easiest way to do it is to prepare a Manipulator-in-the-Middle [6] attack environment to intercept the communication between The Game and the legitimate update server as shown on the (Fig.3).

We will redirect the DNS requests to a DNS server under our control. As the game depends on the operating system to do address resolution, it has no direct authority about it. Our DNS will reply with an IP that redirects the game to our fake update server. This web/proxy server that will transparently forward the requests to their desired destination, logs the whole communication and keeps the transmitted data fragments.

The game successfully connects to our server. The connection to our update server is secured by TLS, but it has no valid certificate as no certificate authority would provide us with a valid certificate without any proof that we are the rightful owners of the domain. (Un)fortunately, the validity of the certificate is never verified - and this is the first security problem, that in principle nullifies the confidentiality of the resulting connection.

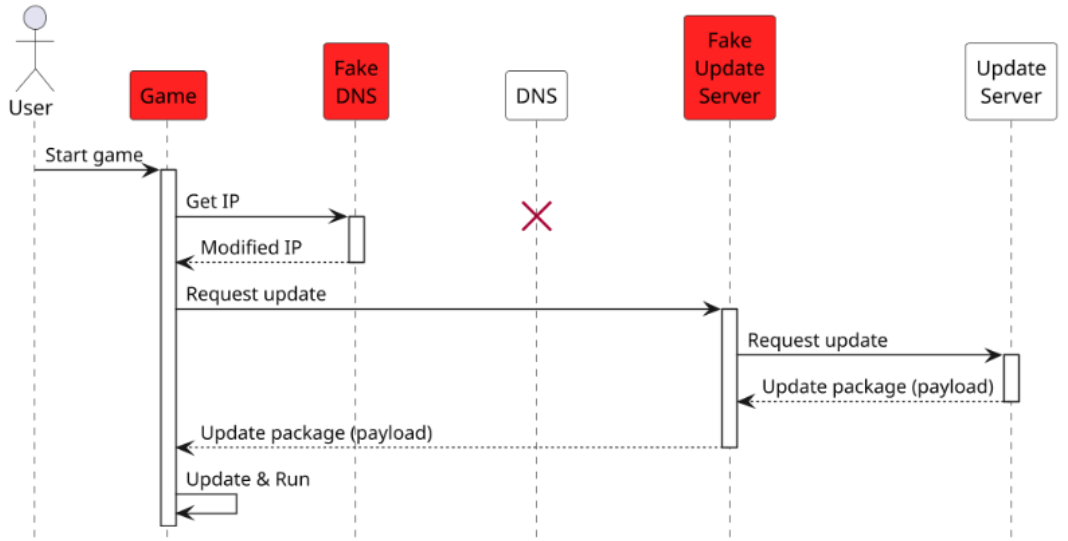

Fig.3. Network traffic during the analysis phase.

As the first step the game checks for available update packages, then downloads an update package for the current version. At this moment we have enough information about the update process and may proceed with the next step - inspection of the update payload.

3.2 Payload Structure

The captured payload is a single ZIP file containing differential data. The ZIP file contains a JSON file describing metadata about the package and all file operations required to update The Game to the new version as shown in (Fig.4). It also contains all the files that need to be updated. One of these files is an executable file thegame.exe that is used to start The Game (Fig.5.).

```
{
  "type": "update",
  "package": "core_expansion-win64",
  "version":
  {
  },
  "apiVersion": 1,
  "membership": "transport-belt-repair-man",
  "author": "",
  "contact": "",
  "www": "",
  "description": " update package (core_expansion-win64 2.0.27 → 2.0.28)",
  "files":
  [

    {
      "file": "__PATH__executable__/ ",
      "action": "differs",
      "old_crc": 3885279022,
      "crc": 2396609988
    },

    {
      "file": "__PATH__executable__/ .pdb",
```


Fig.4. Description of the update package in JSON format.

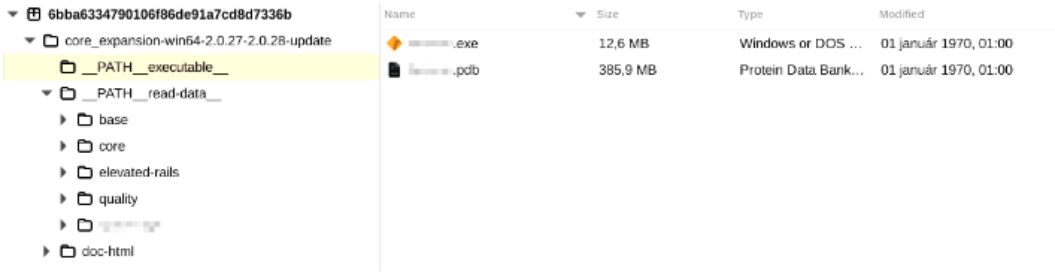

Fig.5. File list of the update ZIP archive.

We expected to find some kind of code signature [7] but we were unable to find it anywhere in the ZIP file or the contained JSON. Only Cyclic Redundancy Check sums are provided for some files. Without this information The Game will be unable to check the validity of the patch. This implies that the game developer has complete trust in the underlying infrastructure or delegates communication security to the often-non-IT-professional, user.

4 Attack Phase

After gathering this information, we may proceed with the "attack" phase. In this phase we will re-use the prepared network environment from the analysis phase. The web server will behave differently and will not resend the update requests but will serve the modified payload instead.

4.1 Payload Modification

We must prepare a modified payload that will pass the internal checks of The Game. Fortunately for us, as we discovered earlier, they are missing. The payload will contain only our specially prepared non-malicious executable file that will display an error message and a simple JSON file describing the installation steps (Fig.6.):

1. delete the existing executable
2. replace the executable with our version


```
"www": "http://[redacted]",
"description": "[redacted] update package (core_expansion-win64 2.0.27 → 2.0.28)",
"files":
[

  {
    "file": "__PATH__executable__/[redacted].exe",
    "action": "removed"
  },

  {
    "file": "__PATH__executable__/[redacted].exe",
    "action": "added"
  }

]
}
```


Fig.6. JSON description of the modified update package.

4.2 Modified Update Process

After the game requests an update, the modified payload is delivered by our server. And this is a second security problem. The implemented update process lacks any authenticity verification, and the payload is trusted without any kind of validation. After the update sequence is completed, the modified payload is executed without any additional user interaction and the target application is compromised. The whole update sequence is shown on (Fig.7).

We have tested this vulnerability on all the major PC operating systems (Windows, macOS and Linux) and succeeded. Even the macOS and Windows built-in security protection against executing an unknown program (Gatekeeper, resp. AppLocker) is not triggered. As the user is aware that an update is in progress, even a privilege escalation may be possible.

4.3 The Fix by the Developer

We have reported the issue to the developer, and it has been fixed a few days later. The update now checks the certificate validity and displays a technical warning that the certificate cannot be verified but older unpatched versions of The Game are still available on the developer's website.

While reporting the issue we stumbled upon an older bug report from 2013 that described the same issue, but the conclusion of the team somehow missed the point as "only authentication token may be disclosed, the bug will be fix in the future", but it finally ended in a "not a bug" basket without any further resolution.

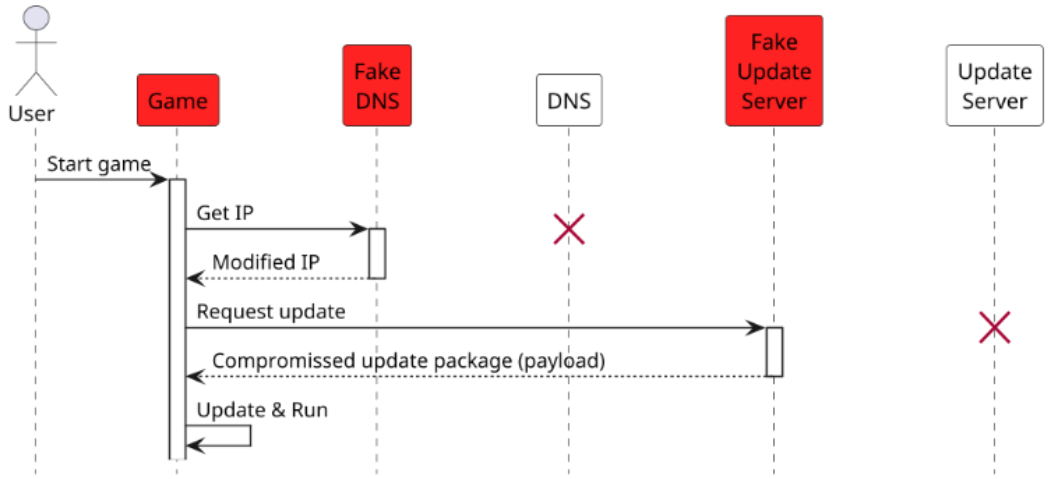

Fig.7. Sequence diagram of the update process with deployment of a modified update package.

Conclusion

The success of this attack confirms our hypothesis about the lack of security focus on the game developer's part, the old bug report even hints that event the consequences of insecure processes are misunderstood. While ignoring the security of the update process, the game may be used as a backdoor to the end-user's computer. As the game is already installed and trusted by the operating system, there is no warning about new executable being installed. If a malicious actor abuses this, the outcome may be quite harsh. User's data may be compromised, deleted, encrypted (ransomware) or a remote access toolkit may be installed.

Information security means balancing the three aspects known as a CIA triad [8] - confidentiality, integrity and availability. Our demonstrations shows what happens if some of these aspects are neglected. To mitigate this type of blunders the game studios should consider employee training in the field of information security and do a regular security audit by a professional third-party auditor.

As this attack relies on the control of the DNS server it is more probable to be deployed in public Wi-Fi scenarios or compromised networks. This implies that any software update should be executed only in a trusted network environment. Also this attack may not be a generic attack but more in the area of spear phishing.

Possible Next Steps

In this work we focused the on a game from the "indie" scene. But what about the A+ titles? How about security updates deployed by different digital distributions or "stores" (Steam, Epic, GOG, Origin, Battle.net, etc.)? And we need to remember that this problem may not be limited to games. This work consisted of a lot of manual steps. Is there a way to automatize it? These are the first areas to consider for further analysis that come to mind, but there may be more.

References

[1] Skelius Mortalis (2024). *Video Game Distribution: From Physical Media to Self-Publishing Breakthrough*. Retrieved online, May 8, 2025, https://www.1d3.com/blog/video-game-distribution-revolution

[2] Gallagher Security Team (2024). *Why software updates are important for security*. Retrieved online, May 8, 2025, https://security.gallagher.com/en/Blog/Why-software-updates-are-important-for-security

[3] Common Attack Pattern Enumeration and Classification (2018). *CAPEC-189: Black Box Reverse Engineering* Retrieved online, May 8, 2025, from https://capec.mitre.org/data/definitions/189.html

[4] Dale Beerling (2020). Alphafunding – *The new trend?* Retrieved online, May 8, 2025, https://www.indiegamemag.com/alphafunding-the-new-trend/

[5] Eric Rescorla, Internet Engineering Task Force (2018). *The Transport Layer Security (TLS) Protocol Version 1.3.* Retrieved online, May 8, 2025, from https://datatracker.ietf.org/doc/html/rfc8446

[6] The Open Worldwide Application Security Project (2025). *Manipulator-in-the-middle attack*. Retrieved online, May 8, 2025, https://owasp.org/www-community/attacks/Manipulator-in-the-middle_attack

[7] Public Key Infrastructure Consortium (2016). *Code Signing Whitepaper - What it Is, Best Practices, and Why its Important* Retrieved online, May 8, 2025, https://pkic.org/uploads/2016/12/CASC-Code-Signing.pdf

[8] Jeroen van der Ham (2021). *Toward a Better Understanding of "Cybersecurity"*. Retrieved online, May 8, 2025, https://dl.acm.org/doi/pdf/10.1145/3442445

Authors

Robert Valík
Pan-European University in Bratislava, Slovakia
xvalik@paneurouni.com
Student of Applied Informatics with over 20 years of experience in Information and Communication Technology in the areas of telecommunications and financial industry.