September 14, 2022

Breaking Bitbucket: Pre Auth Remote Command Execution (CVE-2022-36804)

Introduction

Often when performing application security research, we come across other researchers who have found critical vulnerabilities in software that can inspire us to dig deeper as well. This was the case when we read the blog post from William Bowling about his RCE finding in GitHub Enterprise.

After reading this blog post, we wondered whether or not this methodology to discovering command execution could be replicated on other source code management platforms.

We decided that a good target for this research would be Bitbucket Server, which is typically deployed on-premise and also obviously uses `git` for many operations within the software.

We found an argument injection vulnerability which ultimately allowed us to execute arbitrary commands through the `--exec` argument for `git`. This vulnerability was possible due to the way the underlying process creation library was processing null bytes.

All versions of Bitbucket Server and Datacenter released after 6.10.17 including 7.0.0 and newer are affected, this means that all instances that are running any versions between 7.0.0 and 8.3.0 inclusive are affected by this vulnerability.

This was fixed promptly by Atlassian and they issued CVE-2022-36804 as a result.

Methodology

In order to get an environment that we could perform our security research on, we setup a docker container which was running Bitbucket Server using a pre-prepared image from Docker Hub.

After having a local environment setup, we were able to use pspy to log all processes created, with the specific aim of looking at all sinks to the `git` command.

While we had `pspy` running, in parallel we performed a number of actions inside Bitbucket that we were hoping would trigger the `git` command in some way. We created a repo as a user to build a testing playground for future calls that may be calling out to `git`.

In order to find sinks where the `git` command was being executed alongside with user input, we started to replace everything in all requests involving the created repository with a canary string i.e. `PEWPEW` to determine whether or not it was ending up in the final executed command.

This was relatively easy with the help of `pspy` which was logging every command that was being executed by the process.

We performed testing for a short amount of time, and we were able to find an argument injection within a git subcommand, however it had no security impact (`/rest/api/latest/projects/~USER/repos/repo1/browse?at=--help`).

This was promising, however, we had not found anything of security impact just yet, and we put the project on hold until a later date.

When picking up the project again, we started performing the same methodology and we stumbled across the command execution bug when looking at the API endpoints for Bitbucket, specifically, the endpoint found at `/rest/api/latest/projects/PROJECTKEY/repos/REPO/archive`.

This API endpoint is responsible for streaming an archive of the repository's contents at a requested commit.

When looking at the API documentation for this API endpoint, we noticed that there was a parameter called `prefix` which we assumed mapped to the `git` parameter `--prefix=` in the `archive` subcommand. This seemed like an ideal candidate to test our argument injection exploitation.

In order to inject a new argument, we instinctively tried using null bytes, and to our surprise, when providing the input `padding%00--option%00padding`, an error appeared with the following `-option is not a option to git subcmd`. We realised that our argument injection had been successful due to the error message returned.

Exploitation

While it was amazing to have found argument injection, naturally, the question that arose was, now what? It's great that we can inject arguments, but are we able to escalate this to remote command execution?

It turns out that this was possible due to the functionalities present in the `git archive` subcommand, namely the argument `--exec` which is defined as the following inside Git's documentation:

```
--exec=<git-upload-archive>
Used with --remote to specify the path to the git-upload-archive on the rem
```

At first glance, this may not look as if it is executing an arbitrary command, as it is expecting a path to `git-upload-archive`, additionally, this argument also requires us to specify `--remote` which typicallly is a remote SSH repository.

We tested this command locally, extensively, and found that by executing the following command:

```
git archive --prefix xd --exec='echo pew#' --remote=file:///tmp/ -- blah
```

This would transform into the following:

```
execve('/bin/sh','-c','echo pew# /tmp')
```

Because of this behaviour described above, all that was left to do was to construct a payload for the archive API endpoint in Bitbucket. We could use our argument injection, and abuse Git's behaviour for the `--exec` and `--remote` flags inside the `archive` subcommand to ultimately achieve remote command execution, without even being authenticated to Bitbucket.

The final payload for this can be found below:

```
GET /rest/api/latest/projects/{projectKey}/repos/{repoSlug}/archive?prefix=
Host: bitbucket.demo
User-Agent: HACKZ
Content-Length: 3

xxd
```

While this vulnerability is exploitable pre-authentication, it is necessary for there to be a public repository within the Bitbucket Server instance and you must also know the `projectKey` and the `repoSlug` variables. Without this pre-condition, it is not possible to exploit this vulnerability without authentication.

But why does this work?

When performing security research, sometimes you find yourself in a position where a technique has led to a vulnerability, but it is extremely important to understand why that technique worked in the first place.

In this case, performing root cause analysis of this vulnerability was necessary so we could understand why the null bytes allowed us to inject new arguments to the `git archive` subcommand in the first place.

We started by reverse engineering Atlassian's patch, and we noticed that Atlassian had patched the bug by checking for null bytes in all indexes of the command argument passed to the class `com.zaxxer.nuprocess.NuProcessBuilder`. This was a major hint, as it suggested that this class may have been responsible for splitting up the command through null bytes.

After reading through the class `com.zaxxer.nuprocess.NuProcessBuilder`, we were able to confirm the original hypothesis as why `%00` worked.

It turns out that `com.zaxxer.nuprocess` does not use `ProcessBuilder` or `getRuntime().exec` but rather uses the native `Java_java_lang_ProcessImpl_forkAndExec` which requires a char array as the command argument.

Indexes inside the char arrays are separated by null bytes, and with the way the `prepareProcess` function was transforming the arguments, we were able to create new indexes in this char array by injecting null bytes.

To represent what is happening when we are providing user input with null bytes, you can see the flow below:

String Array to Char Array + VULN:

```
\x00 = {NULL}
["ONE","TWO","THREE","FOUR"] -> conv() (prepareProcess) -> "ONE{NULL}TWO{NU
["git","sub","--safe=xyz","--other"] -> conv() (prepareProcess) -> "git{NUL

exploited

["git","sub","--safe=xyz{NULL}--injected","--other"] -> conv() (prepareProc
```

You can read the vulnerable code, specifically the `prepareProcess` function, below:

```
private void prepareProcess(List < String > command, String[] environment,
  String[] cmdarray = command.toArray(new String[0]);

  // See https://github.com/JetBrains/jdk8u_jdk/blob/master/src/solaris/cla
  byte[][] args = new byte[cmdarray.length - 1][];
  int size = args.length; // For added NUL bytes
  for (int i = 0; i < args.length; i++) {
    args[i] = cmdarray[i + 1].getBytes();
    size += args[i].length;
  }
  byte[] argBlock = new byte[size];
  int i = 0;
  for (byte[] arg: args) {
    System.arraycopy(arg, 0, argBlock, i, arg.length);
    i += arg.length + 1;
    // No need to write NUL bytes explicitly
  }

  // See https://github.com/JetBrains/jdk8u_jdk/blob/master/src/solaris/cla
  byte[] envBlock = toEnvironmentBlock(environment);

  createPipes();
  try {
    // createPipes() returns the parent ends of the pipes, but forkAndExec
    int[] child_fds = {
      stdinWidow,
      stdoutWidow,
      stderrWidow
    };

    if (JVM_MAJOR_VERSION >= 10) {
```

```
        pid = com.zaxxer.nuprocess.internal.LibJava10.Java_java_lang_ProcessI
          JNIEnv.CURRENT,
          this,
          LaunchMechanism.VFORK.ordinal() + 1,
          toCString(System.getProperty("java.home") + "/lib/jspawnhelper"), /
          toCString(cmdarray[0]),
          argBlock, args.length,
          envBlock, environment.length,
          (cwd != null ? toCString(cwd.toString()) : null),
          child_fds,
          (byte) 0 /*redirectErrorStream*/ );
      } else {
        // See https://github.com/JetBrains/jdk8u_jdk/blob/master/src/solaris
        // Native source code: https://github.com/JetBrains/jdk8u_jdk/blob/ma
        pid = com.zaxxer.nuprocess.internal.LibJava8.Java_java_lang_UNIXProce
          JNIEnv.CURRENT,
          this,
          LaunchMechanism.VFORK.ordinal() + 1,
          toCString(System.getProperty("java.home") + "/lib/jspawnhelper"), /
          toCString(cmdarray[0]),
          argBlock, args.length,
          envBlock, environment.length,
          (cwd != null ? toCString(cwd.toString()) : null),
          child_fds,
          (byte) 0 /*redirectErrorStream*/ );
        }
      } finally {
        // If we call createPipes, even if launching the process then fails, we
        // the child side of the pipes are closed. The parent side will be clos
        closePipes();
      }
    }
```

The patch from Atlassian, which remediates this vulnerability can be found below:

```
private void ensureNoNullCharacters(final List<String> commands) {
      for (final String command : commands) {
            if (command.indexOf(0) >= 0) {
                throw new IllegalArgumentException("Commands may not contai
            }
       }
}
```

Vendor Response

The timeline for disclosure can be found below:

- **Jul 19th, 2022**: Disclosure of RCE to Atlassian through BugCrowd.
- **Jul 21st, 2022**: RCE is Triaged by BugCrowd.

- **Jul 21st, 2022**: Atlassian confirms and awards 6K USD for the RCE.
- **Aug 24th, 2022**: Atlassian creates CVE-2022-36804 and publishes an advisory.
- **Aug 24th, 2022**: Atlassian has provided a patched version to the public.
- **Sep 15th, 2022**: We let Atlassian team know that we will be publishing the vulnerability as per our co-ordinated disclosure process.

Remediation Advice

The remediation details provided from Atlassian's advisory are satisfactory and will ensure that this vulnerability cannot be exploited.

The advisory from Atlassian can be found here.

Conclusion

When performing security research, it is valuable to understand some of the methodologies that have led to finding critical vulnerabilities in other enterprise software.

By applying these methodologies across different software, you can often find similar critical vulnerabilities.

Due to being inspired by William Bowling's blog post and adopting a similar methodology, we were able to discover a critical pre-authentication remote command execution vulnerability in Bitbucket Server.

As always, customers of our Attack Surface Management platform were the first to know when this vulnerability affected them. We continue to perform original security research in an effort to inform our customers about zero-day vulnerabilities.

Written by:
Max Garrett

More Like This

Security Research

Insecurity through Censorship: Vulnerabilities Caused by The Great Firewall

Read on ASN Blog >

Security Research

Chaining Three Bugs to Access All Your ServiceNow Data

Read on ASN Blog >

Security Research

Why nested deserialization is harmful: Magento XXE (CVE-2024-34102)

Read on ASN Blog >

Security Research

Digging for SSRF in NextJS apps

Read on ASN Blog >

Security Research

Two Bytes is Plenty: FortiGate RCE with CVE-2024-21762

Read on ASN Blog >

Security Research

Continuing the Citrix Saga: CVE-2023-5914 & CVE-2023-6184

Read on ASN Blog >

Back to All »

Ready to get started?

Get on a call with our team and learn how Assetnote can change the way you secure your attack surface. We'll set you up with a trial instance so you can see the impact for yourself.

Request a Demo

Assetnote

Address:
Level 10, 12 Creek Street, Brisbane QLD, 4000

Contact:
contact@assetnote.io

Press Inquiries:
press@assetnote.io

Platform Features

Continuous Asset Discovery
Deep Asset Enrichment
Assetnote Exposure Engine
Expert Security Research
Collaborative Workflows
Customization

Use Cases

Continuous Asset Discovery and Inventory
Real-Time Exposure Monitoring
Attack Surface Reduction
Mergers & Acquisitions
Bug Bounty Readiness

Privacy Policy