3 Improper handling of untypical characters in domain names

Share:

TIMELINE

philippjeitner submitted a report to Node.js. Apr 28th (2 years ago)

Description

Missing input validation of host names returned by Domain Name Servers in node's `dns` library can lead to output of wrong hostnames (leading to Domain Hijacking) and injection vulnerabilities in applications using the library (leading to Remote Code Execution, XSS, Applications crashes, etc.).

Discoverer(s)/Credits

Philipp Jeitner, Fraunhofer SIT

References

Injection Attacks Reloaded: Tunnelling Malicious Payloads over DNS
https://www.usenix.org/conference/usenixsecurity21/presentation/jeitner
(Available starting from August 11, 2021)

Steps To Reproduce

Using the example application (`main.js`) which does dns lookups via node.

Code 1.75 KiB Wrap lines Copy Download

```
1  const dns = require('dns');
2
3  if (process.argv[2] == "-x") {
4   var host = process.argv[3];
5
6   dns.reverse(host, (err, result) => {
7
8       if (result){
9           for (var i = 0; i < result.length; i++)
10              {
11                  console.log("node".padEnd(8), "reverse".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "IN".padEnd(5), "PTR".padEnd(5), resul
12              }
13          } else {
14              console.log("node".padEnd(8), "reverse".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "-".padEnd(5), "ERROR".padEnd(5), err.errn
15          }
16      });
17
18  } else {
19      var host = process.argv[2];
20      dns.lookup(host, (err, result) => {
21          if (result) {
22              console.log("node".padEnd(8), "lookup".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "IN".padEnd(5), "A".padEnd(5), result);
23          } else {
24              console.log("node".padEnd(8), "lookup".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "-".padEnd(5), "ERROR".padEnd(5), err.errno
25          }
26      });
27
28      dns.resolve(host, (err, result) => {
29          if (result) {
30              for (var i = 0; i < result.length; i++) {
31                  console.log("node".padEnd(8), "resolve".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "IN".padEnd(5), "A".padEnd(5), result
32              }
33          } else {
34              console.log("node".padEnd(8), "resolve".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "-".padEnd(5), "ERROR".padEnd(5), err.errn
35          }
36      });
37
38      dns.resolveCname(host, (err, result) => {
39          if (result) {
40              for (var i = 0; i < result.length; i++) {
41                  console.log("node".padEnd(8), "resolveCname".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "IN".padEnd(5), "CNAME".padEnd(5)
42              }
43          } else {
44              console.log("node".padEnd(8), "resolveCname".padEnd(16), host.padEnd(30), "-".padEnd(80), "-".padEnd(10), "-".padEnd(5), "ERROR".padEnd(5), err
45          }
46
47      });
48
49  }
```

```
$ node main.js cnamezeroweb.test.xdi-attack.net

node     resolveCname     cnamezeroweb.test.xdi-attack.net - -  IN    CNAME zero.longtxtrecord.ml

$ node main.js cnamexss.test.xdi-attack.net

node     resolveCname     cnamexss.test.xdi-attack.net  - -  IN    CNAME <img/src=''/onerror='alert&#x28&#x22xss&#x22&#x29'>.a.cnamexss.test.xdi-attack.net
```

Compare with the output of a well-behaving stub resolver library (glibc) and/or dig:

Code 598 Bytes — Wrap lines Copy Download

```
$ dig dig cnamezeroweb.test.xdi-attack.net

cnamezeroweb.test.xdi-attack.net. 284 IN CNAME  zero.longtxtrecord.ml\000cnamezeroweb.test.xdi-attack.net.
zero.longtxtrecord.ml\000cnamezeroweb.test.xdi-attack.net. 284 IN A 1.2.3.4

$ dig cnamezeroweb.test.xdi-attack.net

cnamezeroweb.test.xdi-attack.net. 300 IN CNAME  zero.longtxtrecord.ml\000cnamezeroweb.test.xdi-attack.net.
zero.longtxtrecord.ml\000cnamezeroweb.test.xdi-attack.net. 299 IN A 1.2.3.4

$ getent hosts cnamezeroweb.test.xdi-attack.net
$ getent hosts cnamexss.test.xdi-attack.net

(no output, return code = 2 because name is filtered)
```

The first issue (cnamezeroweb) is a clear error in zero-byte handling and can potentially lead to DNS-cache injections in case an application implements a cache based on the library.

The second (cnamexss) shows that this can be used to tunnel all kinds of injection payloads, and we argue that applications do not typically expect other characters than [a-z0-9-.] in hostnames. We are aware of applications which can be exploited via this second attack vector (stub dns resovlers which does not filter special characters from hostnames) and argue that stub-resolver libraries should only allow hostnames containing [a-z0-9-.], as it is implemented by glibc's gethostbyname, etc. functions. See the Section 'More information' below on standardization of stub resolver functionality.

Note: One might argue that underscores (_) should also be allowed, since they are used for many application like DMARC, SRV, etc. Actually the underscore was chosen exactly because it is a character not allowed in "hostnames" and thus dmarc records (_dmarc.example.com) does not conflict with "normal" hostnames (See RFC8552, Section 1.1).

The same exploits also apply to reverse-dns records via node's `dns.reverse` function, and probably functions for other record types as well (not tested). You can test this by setting up a nameserver with the following records, in bind9 this requires disabling the `check-names` option in the configuration.

Code 179 Bytes — Wrap lines Copy Download

```
1.1.1.1.in-addr.arpa.   300     IN      PTR     t\000.example.com.
3.3.3.3.in-addr.arpa.   300     IN      PTR     <img/src=''/onerror='alert&#x28&#x22xss&#x22&#x29'>.example.com.
```

Then run `node main.js -x 1.1.1.1` and observe the misinterpreted/unfiltered result.

Note: I selected CWE-170 "Improper Null Termination" as a weakness, however this only applies to the first issue. You might want to consider this two seperate issues (zero-byte handling and missing filtering).

More information

The POSIX Standard for Information Technology defines interfaces for DNS lookups in systems standard C libraries. This Standard includes functions for forward lookups (gethostbyname, getaddrinfo) as well as backward-lookups (gethostbyaddr, getnameinfo). These funtions cannot only return IP addresses but can also contain host names of aliases (CNAME) of the requested host name in case of forward-lookups, or the primary host name of that ip address in the case of backward-lookups (PTR). The POSIX Standard defines the data format of these host names as a null-terminated C-String containing a "hostname" or "nodename", which are typically expected by developers and defined by RFC952 [2] and RFC1123 [3] to only contain alphanumeric characters (a-z,A-Z,0-9), hyphens ("-") and periods (".") to split labels. This creates a mismatch of allowed characters between "hostnames" and "domain names" as defined by the DNS standard [4] which defines "domain names" as a series of "text labels" which are textually represented by concatenating all "text labels" and joining them together with period signs. However, "text labels" can contain any octet value, even zero-bytes ("\x00") and period signs (".") and recursive DNS resolvers are required by the DNS standard to support any of these characters in DNS records, thus not implementing any sanitiy checks on domain names.

When DNS responses are parsed by the stub DNS resolver implemented by stub resolver library as part of the `gethostbyname()`, `getaddrinfo()`, `gethostbyaddr()` and `getnameinfo()` functions, these functions must therefore ensure that the returned, null-terminated C-Strings must be valid domain names as defined by the POSIX standard, else applications which use these values might include that information in contexts where malicious data can included inside the domain name and used for command injection attacks like Cross-Site-Scripting, SQL-injections, etc. Furthermore, if domain names contain text labels with periods (".") or zero-bytes ("\x00") and the stub resolver library does naively decode these domain names into strings, attackers can create malicious domain names which are misinterpreted by the naive decoding logic to look like different domain names than they actually are. When these misinterpreted domain names are than cached by applications using the stub resolver, this allows for domain hijacking by poisoning of the applications DNS cache which uses the vulnerable stub resolver library.

Note: node does not implement a stub resolver as standardized by POSIX, so the rules about allowed vs. non-allowed characters do not directly apply. However, we argue that developers do not know about the specifics of the "hostname" vs. "domain name" consideration, so any library which implements dns lookups should ideally behave in the same way to reduce vulnerabilities caused by developers switching from another language/stub resolver library.

Impact

Impact depends on the application triggering the DNS lookup, see description.

1 attachment:
F1281642: main.js

mcollina Node.js staff changed the status to ○ Triaged. Apr 29th (2 years ago)

Thanks for reporting. I have not set up bind9 to reproduce, but it seems a legitimate concern. Can you confirm all Node.js versions are affected?

We have reproduced the issue with node version v10.19.0 as it is packaged by ubuntu and the current version 10.0.0 as it can be downloaded from https://nodejs.org/en/.

> are you saying you plan to disclose this on August 11, 2021?

Yes, this is part of a paper on issues in dns stub libraries, and node is one of the tested implementations.

philippjeitner posted a comment. Apr 29th (2 years ago)

I have not set up bind9 to reproduce, but it seems a legitimate concern.

You can also just try to resolve one of the public domains operated by us (ie. cnamezeroweb.test.xdi-attack.net) these are available to any resolver in the world, the zone file is just included for completeness.

asnell Node.js staff posted a comment. Apr 29th (2 years ago)

just want to clarify the issue on this to make sure I understand...

The concern for the second issue is that because the `dns` module returns the results unfiltered/unmodified, applications may be at risk if they use the result directly without applying the necessary filtering and content checks?

On the first concern, I agree that it's an issue we should look at.

On the second, I'm not yet as convinced but we can look into it.

philippjeitner posted a comment. Apr 30th (2 years ago)

> I just want to clarify the issue on this to make sure I understand...
> The concern for the second issue is that because the dns module returns the results unfiltered/unmodified, applications may be at risk if they use the result directly without applying the necessary filtering and content checks?

Yes, exactly. We are discussing this issue with developers of multiple libraries, our opinion on why stub resolvers should filter special characters is the following:

> The definition of "hostname" in RFC952 and RFC1123 are still very relevant up to today. Without this restriction what a "hostname" can be, Internationalized Domain Names [RC5890] would be unnecessary (since domain names could carry binary UTF-8 codepoints directly) and any standard which builds up on DNS assumes the same restrictions: For a recent RFC which still contains this restricted definition of "hostname", we refer to RFC8552, Section 1.1 from march 2019 which states:
>
> > Because the DNS rules for a "host" (host name) do not allow use of
> > the underscore character, the underscored name is distinguishable
> > from all legal host names [RFC0952].
>
> The POSIX standard clearly mentions "hostnames" (renamed to "nodenames" in later versions), but fails to specify what a "hostname" or "nodename" is. In our view, the best reference therefore is the IETF definition of "hostname", which is derived from RFC952 in all later documents.
>
> Not filtering special characters in stub resolvers can create 2 issues:
>
> - Applications typically do not expect special characters inside hostnames. We conducted a study various applications and none of them sanitized hostnames before including them in potentially unsafe outputs, like HTML, shell output (ANSI escape codes) or newline-seperated pipes for inter-process-communication. Such behavior is ultimately caused by the application itself, but other that handling direct input from users, our results shows that application developers are not aware that values returned from stub resolvers essentially represent user input as well. This might be because they think that DNS does only support characters [a-z0-9-.] or because they trust the input because it seems to come from the operating system, not a remote peer. Anyway, we argue that fixing such behavior in all affected applications is unrealistic and therefore other ways of mitigating the issue should be taken.
> - Since there is no standard which defines how special characters should be handled by libraries, different libraries might handle special characters differently. For example, some libraries apply the same escaping as done in zone files (escaping characters like "@" with "\@"), but others only escape the bare minimum ("\000" and "."). This different behavior itself can create issues when an application mixes library implementations, ie. when an application acquires a hostname on one system, sends it to another system as a string and this system then tries to process it further. Furthermore, special characters included directly in domain names might overlap with IDNA-domains, which can create a further risk of misinterpretation which can again, lead to vulnerabilities.
>
> These risks can only be addressed by a standardized handling of special characters in domain names in all libraries, otherwise developers might develop their application on one platform and assume that the behavior on this platform applies to other platforms as well, thereby risking the creation of a vulnerability when porting the application. We argue that the best way of doing this would be to stick to the IETF rules, and therefore filter special characters from hostnames.

asnell Node.js staff posted a comment. May 25th (2 years ago)

On the mishandling of the zero-byte in the cname record, it appears that c-ares is to blame here. A fix is going to be quite a bit more complicated in that we would need to either fix upstream or implement alternative parsing logic.

asnell Node.js staff posted a comment. May 25th (2 years ago)

Ok, yeah, confirmed that the zero-byte handling issue is in c-ares. I will explore an upstream fix but it's not clear what we're going to be able to do here yet.

asnell Node.js staff posted a comment. Jun 1st (2 years ago)

There are two issues, each with separate resolutions.

1. For the handling of null characters in record names, the fix must be done upstream in c-ares. I have not yet identified the specific fix. If someone wants to help with this, it would be appreciated.
2. For the second issue on use of names that are returned, I think the most appropriate mitigation is a warning in the docs that applications need to verify and potentially filter the results.

philippjeitner posted a comment. Jun 2nd (2 years ago)

mcollina Node.js staff posted a comment. Jun 5th (2 years ago)

@philippjeitner go ahead and contact the c-ares developers, keep us in the loop!

philippjeitner posted a comment. Jun 11th (2 years ago)

I've contacted the c-ares developers and will notify about the outcome.

philippjeitner posted a comment. Jun 14th (2 years ago)

The issue (bad decoding of zero-bytes) has been fixed in c-ares commits 44c009b and 362f91d. I'm still in contact with the developer, i guess there will also be an advisory and new release at some point.

mcollina Node.js staff posted a comment. Jun 14th (2 years ago)

Awesome news! We'll integrate asap (as soon as a release is cut)
Could you please introduce me (matteo.collina@gmail.com) in the discussion with the C-Ares maintainer so that we could coordinate a date to announce those vulns.

richardlau Node.js staff joined this report as a participant. Jul 23rd (about 1 year ago)

danbev updated CVE reference to CVE-2021-22931. Jul 23rd (about 1 year ago)

danbev updated the severity to High (7.5). Jul 26th (about 1 year ago)

mcollina Node.js staff posted a comment. Aug 2nd (about 1 year ago)

The CVE of c-ares for this vuln is CVE-2021-3672.

mcollina Node.js staff posted a comment. Aug 10th (about 1 year ago)

c-ares has been released: https://c-ares.haxx.se/changelog.html#1_17_2

mhdawson Node.js staff posted a comment. Aug 10th (about 1 year ago)

@philippjeitner does this sound good for attribution:

Thank to you to Philipp Jeitner from Fraunhofer SIT for reporting this vulnerability.

mhdawson Node.js staff posted a comment. Aug 11th (about 1 year ago)

Published: https://nodejs.org/en/blog/vulnerability/aug-2021-security-releases/

mhdawson Node.js staff closed the report and changed the status to **Resolved**. Aug 11th (about 1 year ago)

mhdawson Node.js staff requested to disclose this report. Aug 11th (about 1 year ago)

The Internet Bug Bounty has decided that this report is not eligible for a bounty. Aug 25th (about 1 year ago)

This report has been disclosed. Sep 10th (about 1 year ago)