9 lib/net/ftp.rb: trusting PASV responses allow client abuse

Share:

TIMELINE

ighook submitted a report to Ruby. Apr 2nd (2 years ago)

When `net/ftp` performs a passive FTP transfer, it tries to using PASV. Passive mode is what `net/ftp` uses by default.
A server response to a PASV command includes the (IPv4) address and port number for the client to connect back to in order to perform the actual data transfer.

This is how the FTP protocol is designed to work.[^1]

A malicious server can use the PASV response to trick `net/ftp` into connecting back to a given IP address and port, and this way potentially make it extract information about services that are otherwise private and not disclosed, for example doing port scanning and service banner extractions.
If `net/ftp` operates on a URL provided by a user (with by all means is an unwise setup), a user can exploit that and pass in a URL to a malicious FTP server instance without needing any server breach to perform the attack.

Other FTP clients have in the past also had this flaw and have fixed it at different points in time:

- Chrome in 2009: https://github.com/chromium/chromium/commit/a1cea36673186829ab5d1d1408ac50ded3ca5850
- Curl in 2020 (CVE-2020-8284) : https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8284
- Firefox in 2007 (CVE-2007-1562): https://bugzilla.mozilla.org/show_bug.cgi?id=370559. In that bugzilla issue there's also a link to paper that describes exactly this and lists a few affected clients (link to archive.org since the original has vanished)
  https://web.archive.org/web/20070317052623/http://bindshell.net/papers/ftppasv/ftp-client-pasv-manipulation.pdf

[^1]: With one exception: EPSV. The correct behaviour is first try the EPSV command and if that is not supported, fall back to using PASV.

Impact

This behavior is by design (unless `EPSV ALL` is sent) but it could still lead to security issues depending on the context.
I encountered this issue within a web application with a server-side request forgery (SSRF) issue (but this issue applies to any form of SSRF with `net/ftp` as the request processor). In that context, one can get the following additional capabilities:

- Reliable tcp port scanning (this is not normally possible by just providing a random ip:port to `net/ftp`)
- Network service banner extraction (we setup the data channel on the target ip:port and extract for example an ssh banner: `SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8` without any errors)
- Potential bypass of ip/port restrictions, e.g. the server might be filtering internal IPs or allowing only specific ports (but still allowing FTP)

PoC

I used the following simple code:

Code 162 Bytes — Wrap lines Copy Download

```
require 'net/ftp'
ftp = Net::FTP.new
ftp.connect(ARGV[0], ARGV[1])
ftp.login
#ftp.passive = true # by default
ftp.getbinaryfile('/whatever', 'whatever')
ftp.close
```

And the custom ftp-server:

Code 1.63 KiB — Wrap lines Copy Download

```
[Parent] Got connection from 192.168.100.2:43520... Spawned process 31749 to handle connection
[PID 31749] SEND: 220 FTP PASV Demo Server v1.0
[PID 31749] RECV: USER anonymous
[PID 31749] SEND: 331 Please specify the password.
[PID 31749] RECV: PASS anonymous@
[PID 31749] SEND: 230 Login successful.
[PID 31749] RECV: TYPE I
[PID 31749] SEND: 200 Switching to Binary mode.
[PID 31749] RECV: PASV
 [PID 31750] Handling incoming request to PASV port
 >>> Sending 127.0.0.1:8123
 [PID 31750] SEND: 227 Entering Passive Mode (127,0,0,1,31,187)
 [PID 31750] Exiting
 ------------------------------ The Port is Open --------------------------------
 [PID 31749] RECV: RETR /whatever
 [PID 31749] SEND: 150 Opening BINARY mode data connection for /whatever (0 bytes).
 [PID 31749] SEND: 226 File send OK.
 [PID 31749] Exiting
 --------------------------------------------------------------------------------
 [Parent] Got connection from 192.168.100.2:43524... Spawned process 31787 to handle connection
 [PID 31787] SEND: 220 FTP PASV Demo Server v1.0
 [PID 31787] RECV: USER anonymous
 [PID 31787] SEND: 331 Please specify the password.
 [PID 31787] RECV: PASS anonymous@
 [PID 31787] SEND: 230 Login successful.
```

```
[PID 31788] Handling incoming request to PASV port
>>> 127.0.0.1:8080
[PID 31788] SEND: 227 Entering Passive Mode (127,0,0,1,31,144)
[PID 31788] Exiting
------------------------------ The Port is Closed ------------------------------
[PID 31787] RECV: ERROR: unmatched reply
[PID 31787] Exiting
--------------------------------------------------------------------------------
```

Mitigation

Currently, `net/ftp` can mitigate this flaw by disabling passive mode, which is enabled by default. But this is not the best solution to this problem, perhaps, as well as disabling passive mode by default.

For example, firefox just ignores the ip address that is sent from the server. But Curl provides the option which tell to not use the IP address the server suggests in its response to curl's PASV command when curl connects the data connection. Instead curl will re-use the same IP address it already uses for the control connection. The second seems more reasonable.

mame Ruby staff posted a comment. Apr 2nd (2 years ago)

Thank you for the report. @shugo, can you please check this ticket?

shugo Ruby staff posted a comment. Apr 5th (2 years ago)

Thanks for your report.

I'd like to fix net/ftp to ignore the IP address sent from the server by default, and to add a new option `use_pasv_ip` to use the IP address sent from the server. I think it's a rare case that `use_pasv_ip` is necessary.
Do you have any concerns?

nghook posted a comment. Apr 5th (2 years ago)

Nope. It seems reasonable.

hsbt Ruby staff changed the status to **Triaged**. Apr 5th (2 years ago)

shugo Ruby staff posted a comment. Apr 5th (2 years ago)

I've made the following patch.
Unfortunately new versions of Ruby have just been released yesterday, and the severity of this issue is low, so the fix will be included in the next releases of Ruby (maybe a few or several months later).

Code 8.42 KiB Wrap lines Copy Download

```
commit be549bd6e68877731ec77f163f712189894743f7
Author: Shugo Maeda <shugo@ruby-lang.org>
Date:   Mon Apr 5 17:03:25 2021 +0900

    Ignore IP addresses in PASV responses by default, and add new option use_pasv_ip

    This fixes CVE-2020-8284.
    Reported by Alexandr Savca.

diff --git a/lib/net/ftp.rb b/lib/net/ftp.rb
index dc498e0..eeca328 100644
--- a/lib/net/ftp.rb
+++ b/lib/net/ftp.rb
@@ -98,6 +98,10 @@ module Net
     # When +true+, the connection is in passive mode.  Default: +true+.
     attr_accessor :passive

+    # When +true+, use the IP address in PASV responses.  Otherwise, it uses
+    # the same IP address for the control connection.  Default: +false+.
+    attr_accessor :use_pasv_ip
+
     # When +true+, all traffic to and from the server is written
     # to +$stdout+.  Default: +false+.
     attr_accessor :debug_mode
@@ -206,6 +210,9 @@ module Net
     #                         handshake.
     #                         See Net::FTP#ssl_handshake_timeout for
     #                         details.  Default: +nil+.
+    # use_pasv_ip::  When +true+, use the IP address in PASV responses.
+    #                Otherwise, it uses the same IP address for the control
+    #                connection.  Default: +false+.
     # debug_mode::  When +true+, all traffic to and from the server is
     #               written to +$stdout+.  Default: +false+.
     #
@@ -266,6 +273,7 @@ module Net
       @open_timeout = options[:open_timeout]
       @ssl_handshake_timeout = options[:ssl_handshake_timeout]
       @read_timeout = options[:read_timeout] || 60
+      @use_pasv_ip = options[:use_pasv_ip] || false
       if host
```

```
          raise FTPReplyError, resp
        end
        if m = /\((?<host>\d+(?:,\d+){3}),(?<port>\d+,\d+)\)/.match(resp)
-         return parse_pasv_ipv4_host(m["host"]), parse_pasv_port(m["port"])
+         if @use_pasv_ip
+           host = parse_pasv_ipv4_host(m["host"])
+         else
+           host = @bare_sock.remote_address.ip_address
+         end
+         return host, parse_pasv_port(m["port"])
        else
          raise FTPProtoError, resp
        end
diff --git a/test/net/ftp/test_ftp.rb b/test/net/ftp/test_ftp.rb
index 14afef8..8ab5181 100644
--- a/test/net/ftp/test_ftp.rb
+++ b/test/net/ftp/test_ftp.rb
@@ -61,7 +61,7 @@ class FTPTest < Test::Unit::TestCase
   end

   def test_parse227
-    ftp = Net::FTP.new
+    ftp = Net::FTP.new(nil, use_pasv_ip: true)
     host, port = ftp.send(:parse227, "227 Entering Passive Mode (192,168,0,1,12,34)")
     assert_equal("192.168.0.1", host)
     assert_equal(3106, port)
@@ -80,6 +80,14 @@ class FTPTest < Test::Unit::TestCase
     assert_raise(Net::FTPProtoError) do
       ftp.send(:parse227, "227 ) foo bar (")
     end
+
+    ftp = Net::FTP.new
+    sock = OpenStruct.new
+    sock.remote_address = OpenStruct.new
+    sock.remote_address.ip_address = "10.0.0.1"
+    ftp.instance_variable_set(:@bare_sock, sock)
+    host, port = ftp.send(:parse227, "227 Entering Passive Mode (192,168,0,1,12,34)")
+    assert_equal("10.0.0.1", host)
   end

   def test_parse228
@@ -2509,10 +2517,155 @@ EOF
     end
   end

+  def test_ignore_pasv_ip
+    commands = []
+    binary_data = (0..0xff).map {|i| i.chr}.join * 4 * 3
+    server = create_ftp_server(nil, "127.0.0.1") { |sock|
+      sock.print("220 (test_ftp).\r\n")
+      commands.push(sock.gets)
+      sock.print("331 Please specify the password.\r\n")
+      commands.push(sock.gets)
+      sock.print("230 Login successful.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to Binary mode.\r\n")
+      line = sock.gets
+      commands.push(line)
+      data_server = TCPServer.new("127.0.0.1", 0)
+      port = data_server.local_address.ip_port
+      sock.printf("227 Entering Passive Mode (999,0,0,1,%s).\r\n",
+                  port.divmod(256).join(","))
+      commands.push(sock.gets)
+      sock.print("150 Opening BINARY mode data connection for foo (#{binary_data.size} bytes)\r\n")
+      conn = data_server.accept
+      binary_data.scan(/.{1,1024}/nm) do |s|
+        conn.print(s)
+      end
+      conn.shutdown(Socket::SHUT_WR)
+      conn.read
+      conn.close
+      data_server.close
+      sock.print("226 Transfer complete.\r\n")
+    }
+    begin
+      begin
+        ftp = Net::FTP.new
+        ftp.passive = true
```

```
+        assert_match(/\AUSER /, commands.shift)
+        assert_match(/\APASS /, commands.shift)
+        assert_equal("TYPE I\r\n", commands.shift)
+        buf = ftp.getbinaryfile("foo", nil)
+        assert_equal(binary_data, buf)
+        assert_equal(Encoding::ASCII_8BIT, buf.encoding)
+        assert_equal("PASV\r\n", commands.shift)
+        assert_equal("RETR foo\r\n", commands.shift)
+        assert_equal(nil, commands.shift)
+      ensure
+        ftp.close if ftp
+      end
+    ensure
+      server.close
+    end
+  end
+
+  def test_use_pasv_ip
+    commands = []
+    binary_data = (0..0xff).map {|i| i.chr}.join * 4 * 3
+    server = create_ftp_server(nil, "127.0.0.1") { |sock|
+      sock.print("220 (test_ftp).\r\n")
+      commands.push(sock.gets)
+      sock.print("331 Please specify the password.\r\n")
+      commands.push(sock.gets)
+      sock.print("230 Login successful.\r\n")
+      commands.push(sock.gets)
+      sock.print("200 Switching to Binary mode.\r\n")
+      line = sock.gets
+      commands.push(line)
+      data_server = TCPServer.new("127.0.0.1", 0)
+      port = data_server.local_address.ip_port
+      sock.printf("227 Entering Passive Mode (127,0,0,1,%s).\r\n",
+                  port.divmod(256).join(","))
+      commands.push(sock.gets)
+      sock.print("150 Opening BINARY mode data connection for foo (#{binary_data.size} bytes)\r\n")
+      conn = data_server.accept
+      binary_data.scan(/.{1,1024}/nm) do |s|
+        conn.print(s)
+      end
+      conn.shutdown(Socket::SHUT_WR)
+      conn.read
+      conn.close
+      data_server.close
+      sock.print("226 Transfer complete.\r\n")
+    }
+    begin
+      begin
+        ftp = Net::FTP.new
+        ftp.passive = true
+        ftp.use_pasv_ip = true
+        ftp.read_timeout *= 5 if defined?(RubyVM::MJIT) && RubyVM::MJIT.enabled? # for --jit-wait
+        ftp.connect("127.0.0.1", server.port)
+        ftp.login
+        assert_match(/\AUSER /, commands.shift)
+        assert_match(/\APASS /, commands.shift)
+        assert_equal("TYPE I\r\n", commands.shift)
+        buf = ftp.getbinaryfile("foo", nil)
+        assert_equal(binary_data, buf)
+        assert_equal(Encoding::ASCII_8BIT, buf.encoding)
+        assert_equal("PASV\r\n", commands.shift)
+        assert_equal("RETR foo\r\n", commands.shift)
+        assert_equal(nil, commands.shift)
+      ensure
+        ftp.close if ftp
+      end
+    ensure
+      server.close
+    end
+  end
+
+  def test_use_pasv_invalid_ip
+    commands = []
+    binary_data = (0..0xff).map {|i| i.chr}.join * 4 * 3
+    server = create_ftp_server(nil, "127.0.0.1") { |sock|
+      sock.print("220 (test_ftp).\r\n")
+      commands.push(sock.gets)
+      sock.print("331 Please specify the password.\r\n")
```

```
206 +      sock.print("200 Switching to Binary mode.\r\n")
207 +      line = sock.gets
208 +      commands.push(line)
209 +      sock.print("227 Entering Passive Mode (999,0,0,1,48,57).\r\n")
210 +      commands.push(sock.gets)
211 +    }
212 +    begin
213 +      begin
214 +        ftp = Net::FTP.new
215 +        ftp.passive = true
216 +        ftp.use_pasv_ip = true
217 +        ftp.read_timeout *= 5 if defined?(RubyVM::MJIT) && RubyVM::MJIT.enabled? # for --jit-wait
218 +        ftp.connect("127.0.0.1", server.port)
219 +        ftp.login
220 +        assert_match(/\AUSER /, commands.shift)
221 +        assert_match(/\APASS /, commands.shift)
222 +        assert_equal("TYPE I\r\n", commands.shift)
223 +        assert_raise(SocketError) do
224 +          ftp.getbinaryfile("foo", nil)
225 +        end
226 +      ensure
227 +        ftp.close if ftp
228 +      end
229 +    ensure
230 +      server.close
231 +    end
232 +  end
233 +
234    private
235
236 -  def create_ftp_server(sleep_time = nil)
237 -    server = TCPServer.new(SERVER_ADDR, 0)
238 +  def create_ftp_server(sleep_time = nil, addr = SERVER_ADDR)
239 +    server = TCPServer.new(addr, 0)
240      @thread = Thread.start do
241        if sleep_time
242          sleep(sleep_time)
```

nighook posted a comment. Apr 6th (2 years ago)

LGTM. I also checked the `EPSV/LPSV` message parsing just in case, and everything is fine there as well. Great.

P.S. `parse228/LPSV` is nevertheless vulnerable (when parsing `host/port`), but since it is not used, everything is fine. The main thing is not to forget to fix it if you suddenly decide to introduce support for `LPSV` :)

shugo Ruby staff posted a comment. Apr 6th (2 years ago)

LGTM. I also checked the `EPSV/LPSV` message parsing just in case, and everything is fine there as well. Great.

Thanks for your review.

> P.S. `parse228/LPSV` is nevertheless vulnerable (when parsing `host/port`), but since it is not used, everything is fine. The main thing is not to forget to fix it if you suddenly decide to introduce support for `LPSV` :)

Yes, parse228 is confusing.
I'd like to remove it after this fix is released.

mame Ruby staff posted a comment. Apr 17th (2 years ago)

This report was discussed in the previous developers' meeting, and we agreed that this qualifies as a security issue. I'll ask for CVE later.

shugo Ruby staff posted a comment. May 6th (2 years ago)

CVE-2021-31810 has been assigned:

> [Suggested description]
> An issue was discovered in Ruby through 2.6.7, 2.7.x through 2.7.3, and 3.x through 3.0.1.
> A malicious FTP server can use the PASV response to trick Net::FTP
> into connecting back to a given IP address and port. This
> potentially makes curl extract information about services that are
> otherwise private and not disclosed (e.g., the attacker can conduct port scans
> and service banner extractions).
>
> ---
>
> [VulnerabilityType Other]
> Incorrect Access Control
>
> ---
>
> [Vendor of Product]
> the Ruby community
>
> ---
>
> [Affected Product Code Base]
> Ruby - 3.0.1 or before
> Ruby - 2.7.3 or before
> Ruby - 2.6.7 or before
>
> ---

[Attack Type]
Remote

[Impact Information Disclosure]
true

[Attack Vectors]
To exploit vulnerability, a user must connect to a malicious FTP server.

[Reference]
https://hackerone.com/reports/1145454

[Has vendor confirmed or acknowledged the vulnerability?]
true

[Discoverer]
Alexandr Savca

Use CVE-2021-31810.

CVE Assignment Team
M/S M300, 202 Burlington Road, Bedford, MA 01730 USA
[A PGP key is available for encrypted communications at
https://cve.mitre.org/cve/request_id.html]

hsbt Ruby staff updated CVE reference to CVE-2021-31810. May 6th (2 years ago)


shugo Ruby staff closed the report and changed the status to ○ **Resolved**. Jul 7th (about 1 year ago)

We have released new versions of Ruby and have published the vulnerability.

https://www.ruby-lang.org/en/news/2021/07/07/trusting-pasv-responses-in-net-ftp/

Thank you.

shugo Ruby staff requested to disclose this report. Jul 7th (about 1 year ago)

The Internet Bug Bounty rewarded sighook with a **$500** bounty. Jul 7th (about 1 year ago)

sighook agreed to disclose this report. Jul 8th (about 1 year ago)

This report has been disclosed. Jul 8th (about 1 year ago)