NAT-Again: IRC NAT helper flaws

tl;dr: A Linux kernel bug allows unencrypted NAT'd IRC sessions to be abused to access resources behind NAT, or drop connections. Switch to TLS right now. Or read on.

NAT slipstreaming was an attack where a NAT device is tricked into forwarding requests into a network that is behind a NAT router. The original attack is largely mitigated and primarily targets web browsers.

This attack works against a standard IRC client. It is possible due to a bug in the Linux netfilter implementation, in particular nf_conntrack_irc. I have not checked other NAT implementations but similar attacks may be possible.

Interestingly the discoverer of NAT slipstreaming has an attack from 2010 called "NAT Pinning" that leveraged IRC, but the novel addition here is the fact the matching done on the message is not complete.

Background on IRC

IRC is a simple and very old text based protocol, each line sent or received from the server represents a message in the IRC protocol. In order to extend the protocol IRC clients support a protocol called Client to Client Protocol (CTCP). CTCP simply uses the ^A character ("\001") to mark the special messages that are not messages between users, but some form of control message.

On top of this protocol some clients support a protocol called DCC ("Direct Client-to-Client") that allows for connections between clients, by sending the other client the details of the IP and port to connect to. This protocol is what nf_conntrack_irc targets in order to change the IP sent to the other client and therefore continue to allow DCC connections in the presence of NAT.

The bugs

This attack leverages two bugs in nf_conntrack_irc.

First bug: nf_conntrack_irc does not completely match on the IRC protocol.

As mentioned above DCC messages are contained within a CTCP message, therefore to find a DCC message the string "\1DCC " should be matched. This string should be at the start of a message -- this is the bug, nf_conntrack_irc allows this string anywhere within the outbound TCP stream.

This code snippet from nf_conntrack_irc.c shows the main matching code, in particular it looks through the string (while loop) finding the first instance of "\1DCC " (with memcmp).

```
    /* strlen("\1DCC SENT t AAAAAAAA P\1\n")=24
     * 5+MINMATCHLEN+strlen("t AAAAAAAA P\1\n")=14 */
```

```
    while (data < data_limit - (19 + MINMATCHLEN)) {
        if (memcmp(data, "\1DCC ", 5)) {
            data++;
            continue;
        }
```

Second bug: The external IP address is not checked correctly.

Again, in nf_conntrack_irc.c the comment says it is checking for "the internal OR external IP". In fact this code is buggy and instead checks for the IP address of the IRC server.

```
    /* dcc_ip can be the internal OR external (NAT'ed) IP */
    tuple = &ct->tuplehash[dir].tuple;
    if (tuple->src.u3.ip != dcc_ip &&
        tuple->dst.u3.ip != dcc_ip) {
          net_warn_ratelimited("Forged DCC command from %pI4: %pI4:%u\n",
```

There is also a related third bug that allows for a DoS, if the port specified is 0, then the packet is dropped, eventually resulting the user being disconnected from IRC (because they cannot send any further packets on the connection).

The attack on IRC

Because of the first bug, we can trick a connected IRC client into opening a port forward of our liking by sending it a specially crafted CTCP PING message.

We can also take advantage of the second bug and only need to provide the address of the server the user is connected to (often available from /whois on IRC networks).

If 198.51.100.1 is the address of the IRC server the victim user is connected to, we can craft a message by encoding the IP as a number:

```
$ perl -MSocket -E'say unpack "N", inet_aton("198.51.100.1")'
3325256705
```

This then leads to constructing a raw IRC message like:

```
PRIVMSG ExampleUser :^APING ^ADCC CHAT x 3325256705 22^A
```

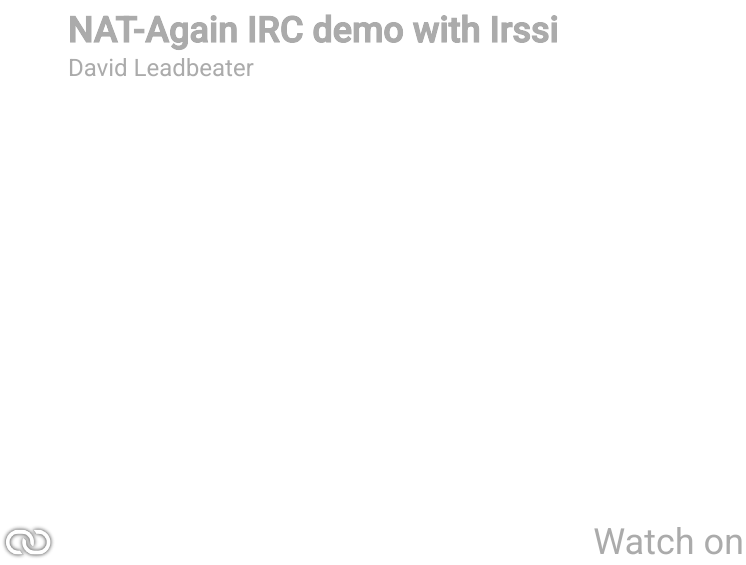
(where ^A is Ctrl-A, to type this on some clients needs Ctrl-V Ctrl-A, prefixing the raw command with /quote to send it from most clients.)

Once this command is run you should see a reply from the user's client, with the 3325256705 number replaced with their numeric IP address. Example:

```
CTCP PING reply from ExampleUser: ADCC CHAT x 3221225985 22
```

Connecting to this IP and port should lead to port 22 on their internal machine.

The above has been tested and works on Irssi and mIRC, other clients may not respond to the PING in quite the same way. Here is a demo video of the attack against Irssi:



Note because the response packet is rewritten it would be possible to use something like this to scan for vulnerable users (in particular, it's possible to send a CTCP PING message to a whole channel); as well as reveal an IP address of a user, that would otherwise be "cloaked" by the IRC server.

The third bug mentioned (sending port 0), combined with the reflection attack means it is possible to disconnect a user.

Mitigations

- **Use TLS for IRC**, this means nf_conntrack_irc cannot intercept DCC requests and DCC NAT traversal will not work without manual port forwarding. (Using TLS hopefully goes without saying these days anyway.)

- Since upstream kernel version 4.6 (commit 3bb398d925ec) NAT helpers have not been loaded by default and need a rule to enable them. However some consumer routers use an older kernel version, or change the default back and load helpers. (As a result of this vulnerability a change is being made to make the helpers require explicit configuration.)

- In the short term remove any iptables rules referencing `-m helper --helper irc` and unload nf_conntrack_irc. On MikroTik devices remove IRC from the service ports list (`/ip firewall/service-port/disable irc`)

- Apply the patch.

Recommendations

- Potentially entirely deprecate and remove nf_conntrack_irc, it's unclear it has much use anymore.
- Fix nf_conntrack_irc to match on "PRIVMSG" and " :" (case insensitively) before the DCC string. It is unlikely IP fragmentation is a concern as in the original NAT slipstreaming attacks because IRC is limited to 512 bytes per message, so a PING request cannot generate a fragmented response, however this could be investigated further.
    - Matching on "PRIVMSG" provides defence-in-depth, as a reply to a CTCP message is sent as a "NOTICE".
- Fix nf_conntrack_irc to match on the external NAT'd IP (or even drop it as it is unclear it has worked correctly and matching only the internal IP somewhat increases the difficulty of attacking this). Patches have been sent for these.
- IRC clients should implement a defence-in-depth of dropping "^A" inside CTCP PING responses. It is also possible for an IRC network to filter this, this vulnerability was pre-disclosed to Libera.Chat and they have implemented filtering.

This issue is tracked as CVE-2022-2663.

Timeline

2022-07-29: Issue discovered, contacted kernel security contacts.
2022-08-03: Notification to various Linux distros + MikroTik.
2022-08-26: Patches sent to netfilter-devel@
2022-08-29: Summary details sent to oss-security.
2022-09-01: These details released.
2022-09-01: MikroTik release testing RouterOS 7.6beta4 containing a fix for this issue.
2022-09-01: First patch is merged into the net.git tree
2022-09-07: Second patch is merged into the net.git tree
2022-09-15: Linux 5.19.9 and other stable kernels released on this date contain patch #1
2022-09-28: Linux 5.19.12 and other stable kernels released on this date contain patch #2

Thanks to G-Research Open Source for supporting some of my work on this. While this may not be a critical vulnerability alone, many open source developers use IRC so this could be used as part of a supply chain attack or similar.

29th August 2022

© David Leadbeater