MikroTik RouterOS Security Vulnerability – L2TP Tunnel Unencrypted – CVE-2017-6297

l Ryan Milne | 24 Feb 2017 | v 0 Comments

m MikroTik CVE-2017-6297, MikroTik, Security

The L2TP Client in MikroTik RouterOS versions 6.83.3 and 6.37.4 does not enable IPsec encryption after a reboot. This allows eavesdroppers to view the transmitted data unencrypted. It also allows eavesdroppers to obtain L2TP client secrets and then establish tunnels to the L2TP servers, gaining unauthorised access to the networks they provide access to.

Vulnerability Summary

The L2TP Client in MikroTik RouterOS versions 6.83.3 and 6.37.4 does not enable IPsec encryption after a reboot. This allows eavesdroppers to view the transmitted data unencrypted. It also allows eavesdroppers to obtain L2TP client secrets and then establish tunnels to the L2TP servers, gaining unauthorised access to the networks they provide access to.

Product overview

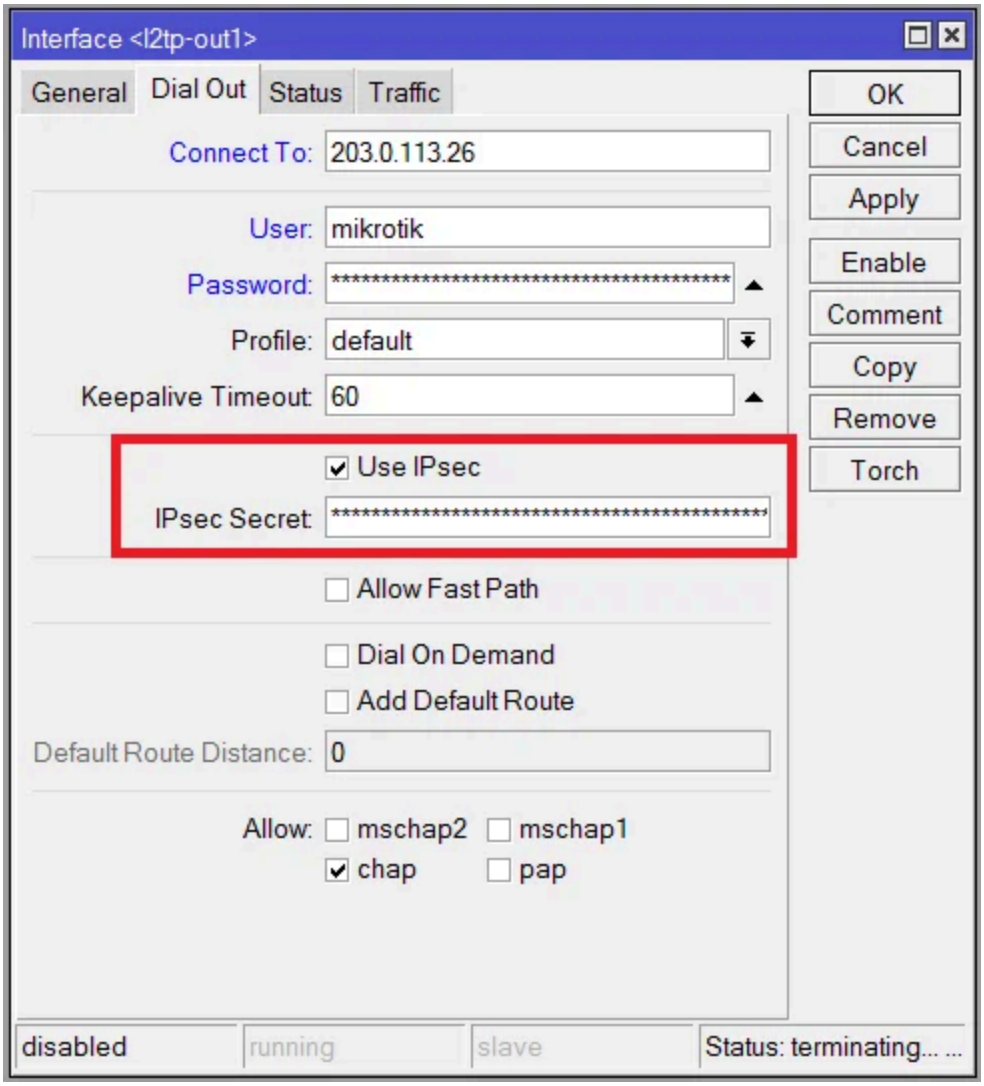
RouterOS is a versatile operating system, as you might guess primarily intended to run as a router. It is a modified version of Linux, where access to traditional GNU/Linux userland is not generally possible, and the device is instead maintained and configured through its own CLI, web interface, Windows GUI application (“WinBox”), and API. RouterOS primarily runs on MikroTik / RouterBOARD hardware; however you can also download it as an x86 virtual machine, branded **Cloud Hosted Router** or **CHR**.

RouterOS is used by both individuals and businesses. Hobbyists often run it on MikroTik hardware in homelabs as – compared to other routers priced for home use such as the DrayTek Vigor range – they are extremely powerful but also very affordable. Businesses use them as affordable and powerful alternatives to Cisco hardware.

RouterOS supports various VPN and tunnelling protocols, including PPTP, L2TP, SSTP, OpenVPN. As well as being able to operate as a server for these tunnels (as is standard for a router), it can also operate as a VPN client.

Vulnerability overview

When you configure an L2TP Client on RouterOS, you have the choice of whether or not to enable IPsec. If you enable it, you configure the L2TP Client with the IPsec Secret.



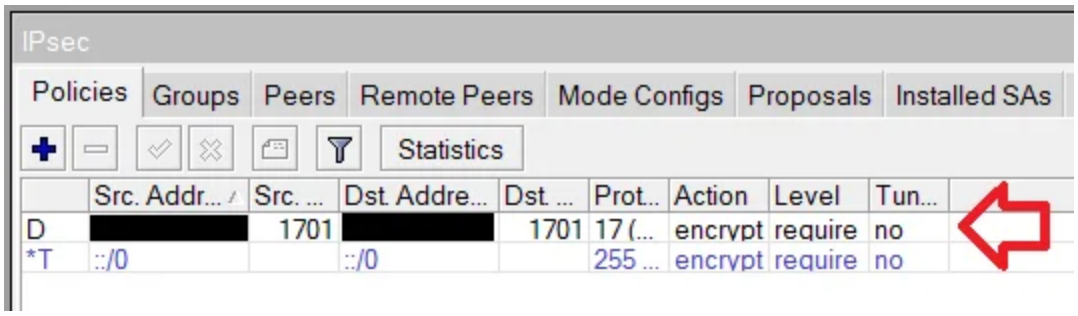
WinBox GUI showing an L2TP Client Interface with IPsec enabled and IPsec Secret specified.

In normal operation, when you enable an L2TP Client interface with IPsec enabled, RouterOS first attempts to establish an IPsec tunnel to the specified server. If the IPsec tunnel is successfully established, all L2TP traffic is transmitted through this tunnel – including the L2TP authentication and any subsequent traffic. If the IPsec tunnel is not successfully established, no L2TP connection is attempted. If you disable and re-enable the L2TP Client interface, this process repeats.

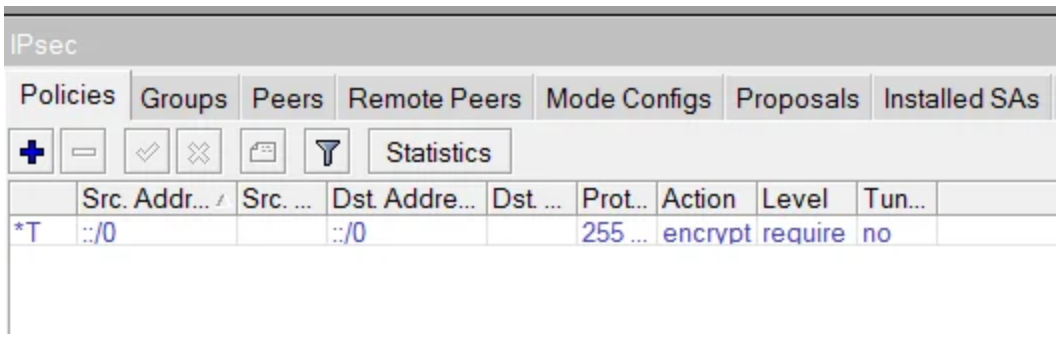
However, if you configure and enable an L2TP Client interface with IPsec enabled and then reboot the router, **RouterOS attempts the L2TP Client connection directly; not through IPsec**. If the L2TP server does not accept connections without IPsec, then the connection fails and the tunnel is never established until one disables and re-enables the interface. However **if the L2TP server also accepts connections without IPsec, an unencrypted L2TP tunnel is successfully established**.

There is no direct feedback to the user that the L2TP tunnel has been established without IPsec encryption. One can tell as follows:

- On RouterOS, viewing **IP > IPsec > Policies** in normal operation shows a dynamically created policy for the L2TP Client connection. When this bug is triggered, there is no such policy.

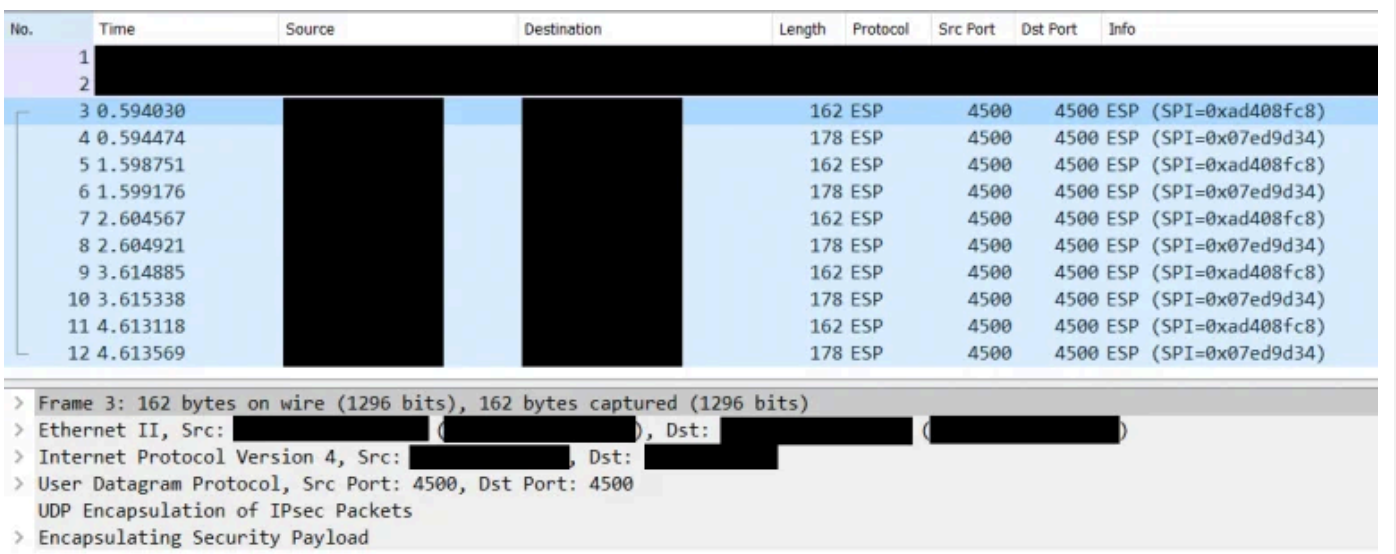


L2TP tunnel in normal operation. A dynamic IPsec policy is visible.

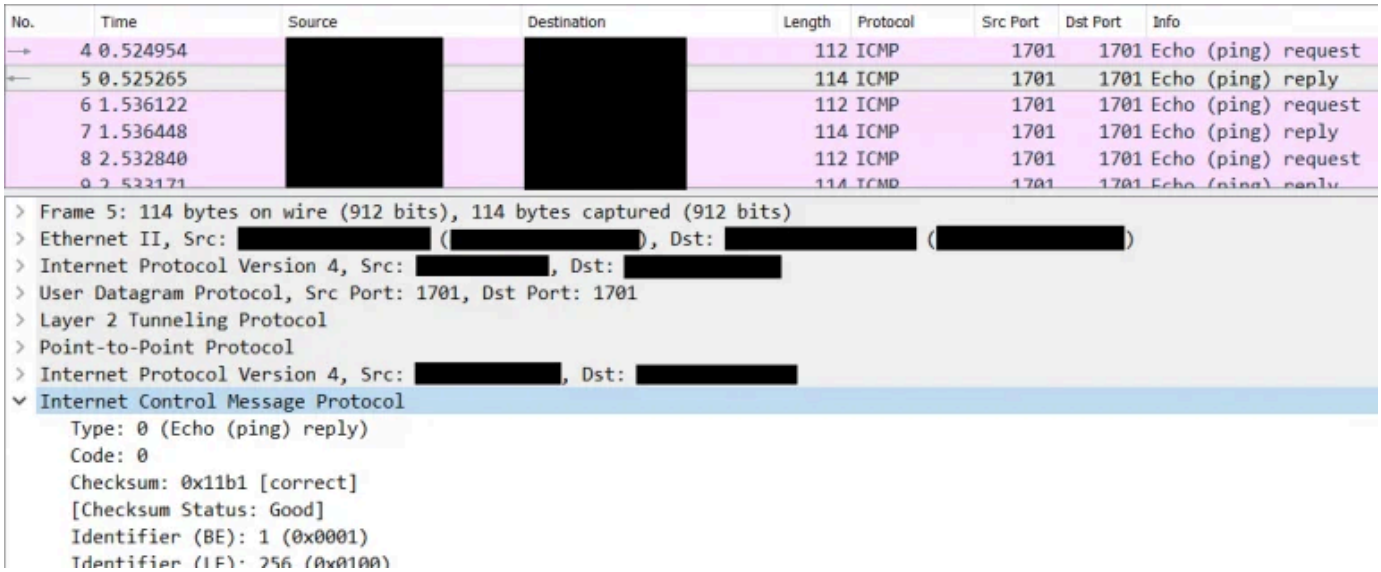


L2TP tunnel after bug triggered. The dynamic IPsec policy is missing.

- On RouterOS, viewing the log in normal operation shows entries relating to the establishing of the IPsec tunnel – for example “IPsec-SA established“. When this bug is triggered, these messages are not written to the log.
- Capturing the physical interface’s packets (for example with Wireshark, or using RouterOS’s Packet Sniffer), would show ESP traffic in normal operation, i.e. encrypted so you cannot see the contents of the inner tunnel’s packets. When this bug is triggered, the inner tunnel’s packets are not encrypted and can be read directly.



L2TP tunnel in normal operation. Traffic is encrypted by ESP (in this case over UDP 4500 due to NAT-Traversal.)



L2TP tunnel after bug triggered. Packets sent through the tunnel are visible outside, wrapped in the L2TP tunnel. In this case you can see ICMP ping packets in both directions.

Impact

Consider the following real-world scenario: You wish to connect your MikroTik router to an L2TP/IPsec server. The L2TP/IPsec server also accepts L2TP connections without IPsec. The reason why unencrypted L2TP connections are accepted isn't relevant, but for example it could either be a server configuration oversight, or it could be intentional.

- Oversight example: With a Linux server running libreswan for IPsec and xl2tpd for L2TP, you would normally use iptables to block L2TP traffic without an IPsec policy, and allow L2TP traffic encrypted by IPsec. Those without a decent understanding of iptables are unlikely to set up these rules correctly, or even at all. Even those with a good understanding can make innocent mistakes.
  See the example iptables INPUT chain below. Some may think that, after rule 6 accepts L2TP/IPsec traffic, rule 7 is sufficient to drop L2TP unencrypted traffic. However, if rule 6 has recently allowed an L2TP/IPsec connection, rule 4 can subsequently allow an unencrypted L2TP connection between the same IP addresses, as the connection tracking believes it's an existing connection that should be allowed through. Rule 2 is required prior to rule 4 in order to prevent this.

```
1  Chain INPUT (policy ACCEPT)
2  num  target     prot opt source               destination
3  1    ACCEPT     icmp --  anywhere             anywhere
4  2    DROP       udp  --  anywhere             anywhere             udp dpt:l2f p
5  3    DROP       all  --  anywhere             anywhere             ctstate INVAL
6  4    ACCEPT     all  --  anywhere             anywhere             ctstate RELAT
7  5    ACCEPT     udp  --  anywhere             anywhere             multiport dpo
8  6    ACCEPT     udp  --  anywhere             anywhere             udp dpt:l2f p
9  7    DROP       udp  --  anywhere             anywhere             udp dpt:l2f
10 8    DROP       all  --  anywhere             anywhere
```

- Intentional example: A server that is used for normal encrypted connections by road warriors, and also tunnels some other traffic between servers that is already satisfactorily encrypted and thus does not need to waste CPU on IPsec (e.g. SSH, HTTPS).

This bug has two main consequences.

First, all of the traffic between the RouterOS L2TP Client and the L2TP server is unencrypted (without the user being made clearly aware), and can be intercepted by eavesdroppers on the wider internet with the relevant access. Privacy is **the** core expectation of using a VPN, so this is gravely unacceptable behaviour.

By analogy, this bug is similar to if you tell your web browser to go to TLS-encrypted **https**://www.example.com, but it instead goes to unencrypted **http**://www.example.com. Now, these days many websites and web servers have measures to ensure that HTTPS is used and HTTP connections never take place. It's good that webmasters are doing this "for the greater good", but that doesn't somehow make it the server's responsibility – if the client is told to use encryption, it should use encryption. At bare minimum, it should warn the user if an encrypted session is not established and an unencrypted one is going to be used instead. Similarly, while it is certainly a best practice for an L2TP/IPsec server to prevent unencrypted L2TP connections, it's not their responsibility to the client to do so.

Second, the L2TP authentication process is also unencrypted, which can allow an eavesdropper to compromise the L2TP server and/or network behind it.

In more detail, RouterOS can perform PPP authentication with PAP, MS-CHAP (v1 or v2), or CHAP.

- PAP – The L2TP password is transmitted in the clear, and so is unencrypted. If the PPP authentication process is captured by an eavesdropper, they can easily obtain the L2TP password.
- MS-CHAP (v1 or v2) – These mechanisms have known vulnerabilities. If the PPP authentication process is captured by an eavesdropper, they will be able to crack the password with not much effort.
- CHAP – As far as I know, CHAP does not have any vulnerabilities as such. It does use MD5 as its cryptographic hash function however, which is quicker to crack than other hash algorithms and has known weaknesses. So if the PPP authentication process is captured by an eavesdropper, depending on the quality of the password, they may be able to crack it.

With PAP and MS-CHAP, the password is as good as given to the eavesdropper. With CHAP, it depends on the strength of the password. Overall, an eavesdropper has a much easier time obtaining the L2TP password, which will allow them to impersonate the user to the L2TP server and establish an L2TP tunnel. Depending on the configuration of the L2TP server, this may give them access to internal networks intended to be protected from the outside world.

Proof of concept

The simplest proof of concept involves using two RouterOS instances – one as the L2TP server and one as the L2TP client. The RouterOS L2TP server has to have IPsec either enabled or disabled – it cannot be configured to accept both encrypted and encrypted L2TP connections at the same time – so it is configured with IPsec disabled. An L2TP client with IPsec enabled should thus fail to connect to this server, however after a reboot it does not.

1. On the server instance:
    - Enable the L2TP server.
    - Configure an IP pool of addresses to give out to connecting clients.

- Configure the ppp profile to use this IP pool.
- Configure a PPP username and password.

For example, on the command line:

```
/ip pool
add name=L2TP ranges=192.168.123.10-192.168.123.20
/ppp profile
set *FFFFFFFE local-address=192.168.123.1 remote-address=L2TP
/interface l2tp-server server
set enabled=yes
/ppp secret
add name=ppp1 password=1234
```

2. On the client instance, configure the L2TP client as follows:
   - Configure the IP of the server instance.
   - Configure the PPP username and password.
   - Enable IPsec and set an IPsec secret.

   For example, on the command line, assuming the server instance has IP address 192.168.100.101:

```
/interface l2tp-client
add connect-to=192.168.100.101 disabled=no ipsec-secret=5678 name=l2tp-out1 \
    password=1234 use-ipsec=yes user=ppp1
```

3. On the client instance, when you enable the L2TP client interface, the tunnel fails to establish. If you review the logs, you can see that it attempted to establish an IPsec session. If you view **IP > IPsec > Policies**, you can see that a dynamic IPsec policy has been created to ensure L2TP traffic is encrypted between the client and server. This is working as expected, as the L2TP server is not configured to accept IPsec connections.
4. On the client instance, if you disable the client interface, you can see that the IPsec policy is deleted. If you reenable the client interface, it is recreated, and the tunnel still fails to establish. This is working as expected.
5. If you reboot the client instance operating system (for example with /system reboot), when the router starts back up, it will successfully establish the L2TP session. The logs will show no IPsec session. Under **IP > IPsec > Policies**, no policy has been created. If you capture the traffic from the physical interface and send some traffic through the tunnel, you will be able to see the unencrypted packets encapsulated within L2TP without IPsec encapsulation.

A different proof-of-concept more similar to real-world scenarios can be created using a Linux server using libreswan and xl2tpd, and **not** including iptables rules on the INPUT chain to drop port 1701 traffic that does not have an IPsec policy. (This script is very useful for creating this configuration – remove any DROP rules for port 1701 after running it). Such a server will accept L2TP connections either with or without IPsec. Configuring the RouterOS L2TP Client correctly with IPsec enabled will establish L2TP/IPsec connections successfully in normal operation, as well as establish unencrypted L2TP connections without IPsec after a reboot.

Vulnerable / tested versions

RouterOS maintains two stable branches – “Current” and “Bugfix”. This vulnerability has been demonstrated on both the latest Current (v6.83.3) and the latest Bugfix (v6.37.4) at the time of writing. It is likely present in older versions as

well.

Solutions / Workarounds

MikroTik Support have confirmed to me by email that this will be fixed in an upcoming version.

There are a few workarounds and mitigations:

1. Create a firewall rule on the outbound chain that blocks unencrypted L2TP traffic.
   (Note: This is a good practice for L2TP/IPsec regardless of this vulnerability.)

   This will block outbound L2TP traffic (including authentication) unless it is encrypted by IPsec. When the server is rebooted, the L2TP tunnel will fail to establish until the L2TP client interface is manually disabled and re-enabled, but it will not perform the L2TP handshake or tunnel any packets over the open internet.

   For example:

   ```
   /ip firewall filter add action=reject chain=output comment="Outbound - Block L2TP
       icmp-admin-prohibited
   ```

2. Create a static IPsec policy identical to the dynamic policy created in normal operation when the L2TP Client Interface is enabled. This policy will force the L2TP traffic to always be encrypted by IPsec, in lieu of the dynamic policy that is not created after a reboot.
   This workaround is difficult when dynamic IP addresses are involved and requires scripting to ensure the policy is kept up-to-date with the correct IP addresses.

3. If possible / applicable, use a different type of tunnel to L2TP/IPsec.
   - IPIP/IPsec tunnels are unaffected by this issue.
   - GRE/IPsec tunnels have not been tested, but judging from the similarities between IPIP and GRE configuration and differences from L2TP configuration in the WinBox GUI, it seems likely that GRE is also unaffected.
   - OpenVPN and SSTP do not use IPsec for encryption, and so are unaffected.
     (Note: PPTP also does not use IPsec. However PPTP has unrelated known weaknesses, and thus it is strongly recommended to avoid using it.)

Additionally, if the user also has control over the L2TP server, it would be a good practice to attempt to prevent unencrypted connections on the L2TP server as well, unless there is a need for unencrypted L2TP connections.

Vendor contact

MikroTik do not appear to have a specific method for reporting security vulnerabilities, so I contacted them through their support email address on their website. Overall, contact with MikroTik has had both positives and negatives.

On the positive side, support were very responsive and very polite. They agreed that the vulnerability can be publicly disclosed immediately so as to warn their customers, and they certainly didn't make any threats in relation to the

disclosure. They didn't ask for any proof of ownership of a RouterOS licence or RouterBOARD product, which is normally a requirement for support according to their website.

On the negative side, MikroTik do not seem to have the appropriate procedures in place to handle vulnerability disclosure. In particular there is no real way to securely communicate with them. They did not offer GPG keys. It seems very likely that the HTTPS form on their website simply generates an email that is sent to them, in all likelihood also unencrypted, and it's a one-off form rather than a messaging portal – the conversation continued over email. Furthermore, they requested the supout.rif file that RouterOS generates to assist in support issues, which I have discovered includes sensitive information such as IPsec secrets in plain text. They also seemed to put the onus on me to fully prove the bug – this is understandable if they have a bug bounty program, but as far as they know they do not, so they were relying on my goodwill.

- 2017-02-17: Emailed support notifying of a security vulnerability, and asking for a secure method of contact.
- 2017-02-17: Support replied inviting to send the vulnerability over email.
- 2017-02-17: Replied expressing concern over providing details through unencrypted email.
- 2017-02-17: Support replied saying it's up to me how to deliver the information, but suggested submitting through the mikrotik.com website's HTTPS form.
- 2017-02-17: Submitted vulnerability details by HTTPS form.
- 2017-02-17: Received email response from support indicating resolution on the MikroTik Wiki.
- 2017-02-17: Replied informing them that this mainly applied to L2TP servers, and that this was still a security bug nonetheless.
- 2017-02-22: Support replied asking for supout.rif files for further diagnostic.
- 2017-02-22: Replied providing supout.rif files, and also a step-by-step for reproducing on CHR.
- 2017-02-23: Support replied confirming they could reproduce the bug and would fix it in a future version of RouterOS.
- 2017-02-23: Replied asking for an agreeable time before publicly disclosing the vulnerability.
- 2017-02-23: Support replied saying it can be disclosed now. "It is better that other users know that such security problem may happen."

Responsible Disclosure

The technical ability of RouterOS's "end-users" is generally quite high, so they are easily able to apply one of the workarounds / mitigations above, almost as easily as they would apply RouterOS updates. Given this, and the fact that RouterOS does not update automatically, there is little security benefit to delaying disclosure until MikroTik release the update with the bug fixed. Disclosing this vulnerability now has the greater security benefit of allowing RouterOS L2TP Client users to put in place suitable mitigations prior to MikroTik fixing the bug.

← Previous post Next post →