WITH LIBSSH, AUTHENTICATION IS OPTIONAL

October 18, 2018 BY  DANIEL GOLDBERG  OFRI ZIV 1 COMMENT

LibSSHhhh, You've Been Exploited: A New Vulnerability Allows Authentication Bypass

A critical vulnerability (CVE-2018-10933) was disclosed in libSSH, a library implementing the SSH2 protocol for clients and servers. The vulnerability allows an attacker to completely bypass the authentication step and connect to the server without providing any credentials, the worst possible flaw for a library implementing SSH. This library is in use by many small applications and embedded systems, among them GnuGK, a VOIP gateway service. This vulnerability was disclosed by Peter Winter-Smith of NCC Group, and patched versions already exist.

In this post we'll explain the scope of the vulnerability, explain the source, clarify how to exploit it and how to check internal appliances to see if they're vulnerable. GuardiCore Labs has previously done this for vulnerabilities such as Sambacry and Wannacry.

Who's Using libSSH

The vulnerable libSSH library can be used both as a client and a server SSH library and the vulnerability is only on the server side. The vulnerability was introduced in version 0.6, released in 2014, and survived until October 16th, 2018 whereupon it was fixed in versions 0.8.4 and 0.7.6, released earlier this week.

A vulnerable server is completely wide open and could easily be compromised by any attacker. The compromise's impact will depend on the permissions given to the SSH server and can provide the attacker full control over the vulnerable machine or a possible tunneling interface into internal networks.

It is important to note that this library is not in common use as an SSH server. Most SSH servers use OpenSSH, a popular open source library implementing the SSH protocol for servers. Shodan reports show fewer than 3,000 internet facing servers in the U.S. running a vulnerable version of the library.

The complicating factor is that there is no single popular package relying on this library, instead it seems that dozens of small applications use this server. The vulnerability is over four years old and with a long tail of software packages depending on it. Prior experience teaches us that vulnerable instances of this library will be around in networks for a very long time. Many of the applications running vulnerable versions of this library will never be patched, providing attackers an easy path for lateral movement.

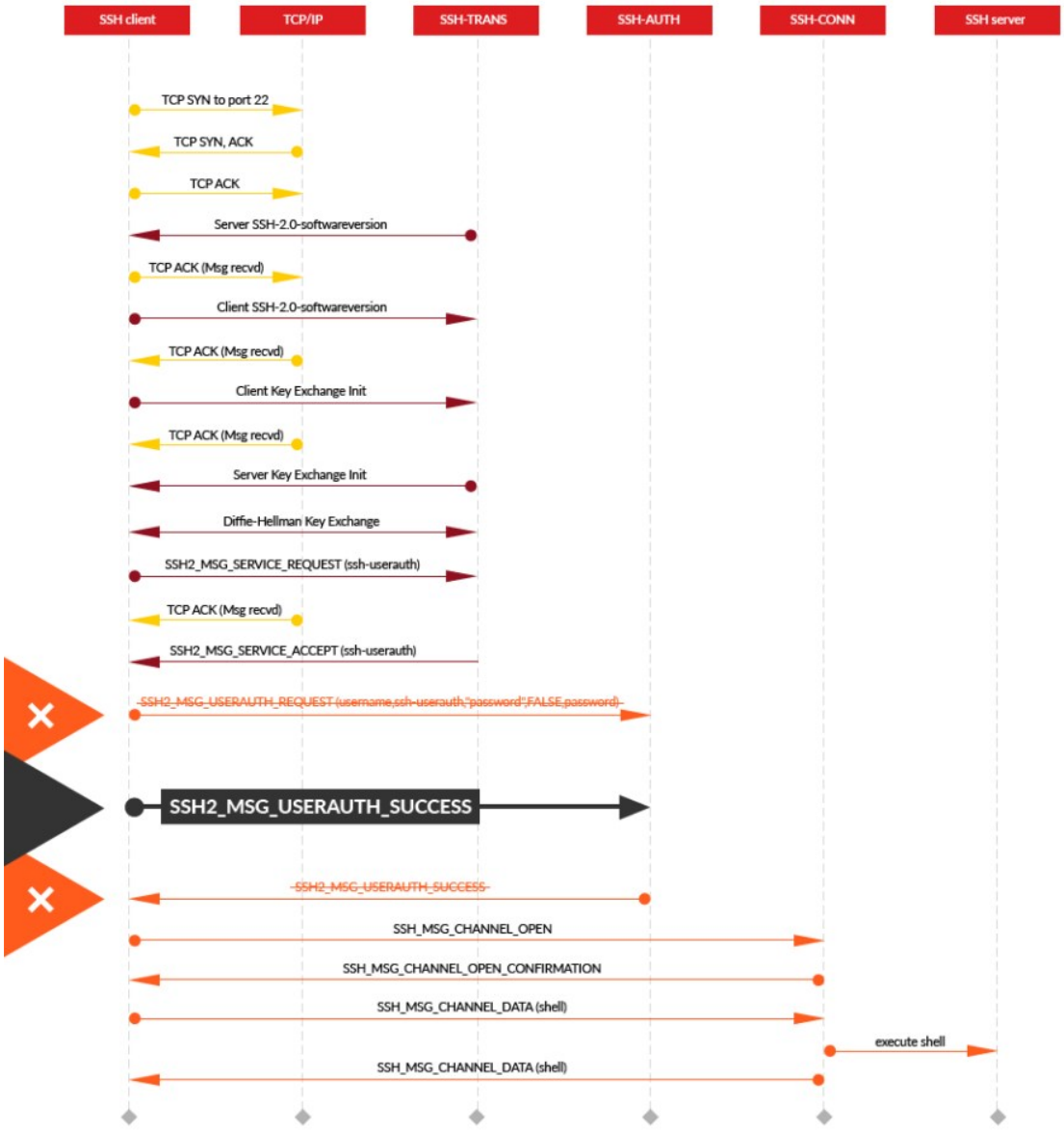
UPDATE: Per a recent security advisory from Cisco, there are a large number of Cisco services and appliances that are running a vulnerable version of libSSH. This means that a far greater number of servers are at risk than previously estimated.

What is the Vulnerability?

The vulnerability is simple to understand when looking at the protocol in diagram format. Below is the SSH connection

process, requiring a SSH2_MSG_USERAUTH_REQUEST message containing the client's username and authentication data, such as a password. The server replies with SSH2_MSG_USERAUTH_SUCCESS if the credentials are valid.

In case of libSSH, a user could just skip the authentication process and have his client send the SSH2_MSG_USERAUTH_SUCCESS and bypass all checks.



Peter Winter-Smith of NCC Group provided a clear explanation of where the vulnerability resides in the code itself.

> The issue is basically a bug in the libSSH library, not to be confused with the similarly named libSSH2 or OpenSSH projects (especially the latter) which results from the fact that the server uses the same state machine to authenticate clients and servers.
> The message dispatching code that processes messages either in client mode or server mode (it's the same function) doesn't make sure that the message type received is suitable for the mode it's running in. So, for example, the server will dispatch messages which are only intended by design for processing client side, even when running in server mode.

So when the client maliciously sends SSH2_MSG_USERAUTH_SUCCESS, the following code is triggered.

```
SSH_PACKET_CALLBACK(ssh_packet_userauth_success) {
  (void)packet;
  (void)type;
  (void)user;
//...
  session->auth.state = SSH_AUTH_STATE_SUCCESS;
```

```
  session->session_state = SSH_SESSION_STATE_AUTHENTICATED;
  session->flags |= SSH_SESSION_FLAG_AUTHENTICATED;

//..
/* Reset errors by previous authentication methods. */
    ssh_reset_error(session);
    session->auth.current_method = SSH_AUTH_METHOD_UNKNOWN;
  return SSH_PACKET_USED;
}
```

At this point, the servers session object is set to authenticated and when the client sends a SSH_MSG_CHANNEL_OPEN message, the following state verification in ssh_packet_channel_open is bypassed.

```
if (session->session_state != SSH_SESSION_STATE_AUTHENTICATED){
    ssh_set_error(session,SSH_FATAL, "Invalid state when receiving channel
open request (must be authenticated)");
    goto error;
  }
```

And the attacker successfully connects without providing any credentials and can easily open a shell.

Detection and Mitigation

This vulnerable library is likely to be found in closed box appliances,servers or embedded devices and will require effort to find vulnerable instances in your network.

You can check if a specific SSH service is vulnerable by running nmap with the -sV flag to check which SSH library is in use. Note that in many cases where the library is embedded into another application, the SSH service may run in a non-standard port rather than port 22 and you should refer to the products documentation.

A sample output for a vulnerable machine would look like this :

```
22/tcp open  ssh libSSH 0.7.2 (protocol 2.0)
```

If the version is earlier than 0.8.4 or 0.7.6 (different branches), then the application is vulnerable. If this is the case, you should check if an updated version with a newer libSSH package is available.

In cases where no update is available, you should tightly control access to the machine, as this vulnerability will bypass application level defenses. Possible mitigations include iptables or firewall rules that prevent access from unauthorized machines.

Speak “Friend,” and Enter

A complete authentication bypass in a library implementing one of the internet’s bedrock protocols is always worthy of notice. This time around the damage potential is limited, as libSSH is mostly used as a client library rather than a server. Even so, in networks running software embedding libSSH, such as a large number of Cisco appliances, this vulnerability allows attackers to just say SSH2_MSG_USERAUTH_SUCCESS and they’re in.

We encourage you to check if you have internal appliances using libSSH and to make sure to update these applications. This vulnerability should be a driver to make sure your organization tracks the long tail of software dependencies found in the network and build patching plans for every part of the network.

Tags: authentication bypass, Cisco, CVE-2018-10933, libSSH, vulnerability

1

COMMENT

redParrot

October 20, 2018 at 9:33 am

One of my favorites :)))

Reply

Leave a Comment

Want to join the discussion? Feel free to contribute!

Name *

Email *

POST COMMENT

LATEST POSTS

OPERATION PROWLI: MONETIZING 40,000 VICTIM MACHINES

AZURE PASSWORDS ARE STILL AT RISK; INFECTION MONKEY CAN HELP

<- Back to Guardicore Labs