
// Workshop In Information Security
Under supervision of Reuven Plevinsky @ Check Point
Aviv Yaniv / Tel-Aviv University
DD/MM/YY
LibSSH Auth Bypass
SSH
DLP
Firewall Architecture
First Stages
Whats ahead of us?

Whats ahead of us?

* Steps taken while building the FW
* Detailed explanation on the DLP module
* Share of research process and defense mechanism

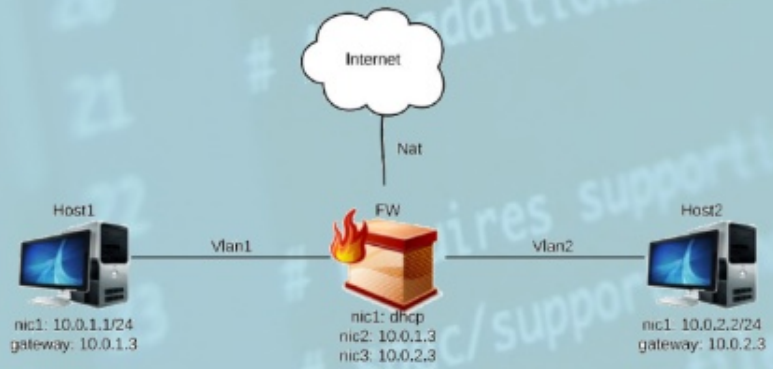
Feel free to interrupt and ask questions :)


// Workshop In Information Security
Under supervision of Reuven Plevinsky @ Check Point
Aviv Yaniv / Tel-Aviv University
DD/MM/YY
Whats ahead of us?
First Stages
Firewall Architecture
DLP
LibSSH Auth Bypass
SSH

First Stage

The first step I've implemented a simple FW
* Inner packets (from FW to FW) - Passed
* Outgoing packets (to any other host) - Blocked

V Learned to write device drivers
V Became familiar with Linux Netfilter & hooks
V Hands-on experience with Linux Kernel Programming



Second Stage

Third Stage

Linux Kernel

Second Stage

* Added user space interface
* Counting number of incoming & outgoing packets

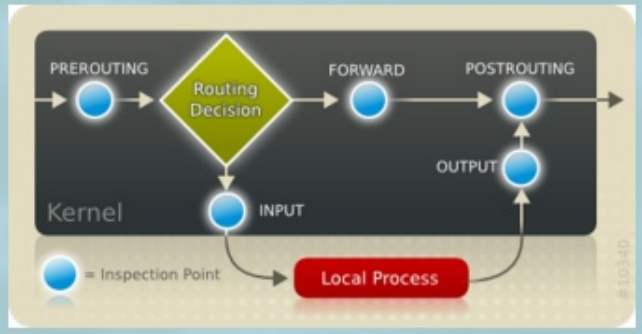
```
fw@fw:~$ ./a.out
Firewall Packets Summary:
Number of accepted packets: 182
Number of dropped packets: 96
Total number of packets: 278
```

Third Step

* Implementing stateless packet filtering
* Use of rules to decide packets verdict
* Logging of packets in a dynamic list (Linux Kernel klist)

When a packet reaches the NF_IP_FORWARD STAGE,
it's being examined against the predefined rules to decide it's verdict

V Understand real-life rules of packets
V More hands-on experience in device drivers & Linux Kernel
V Special use-case: XMAS packet (PSH & URG & FIN) - blocked



Linux Kernel

During the third stage, in order to maintain log dynamic list - used first online source implementation
* When ran on PC - worked fluently
* When ran on VM - created kernell panic
=> Used klist !


// Workshop In Information Security
Under supervision of Reuven Plevinsky @ Check Point
Aviv Yaniv / Tel-Aviv University
DD/MM/YY
Whats ahead of us?
First Stages
Firewall Architecture
DLP
LibSSH Auth Bypass
SSH

Fourth Sage

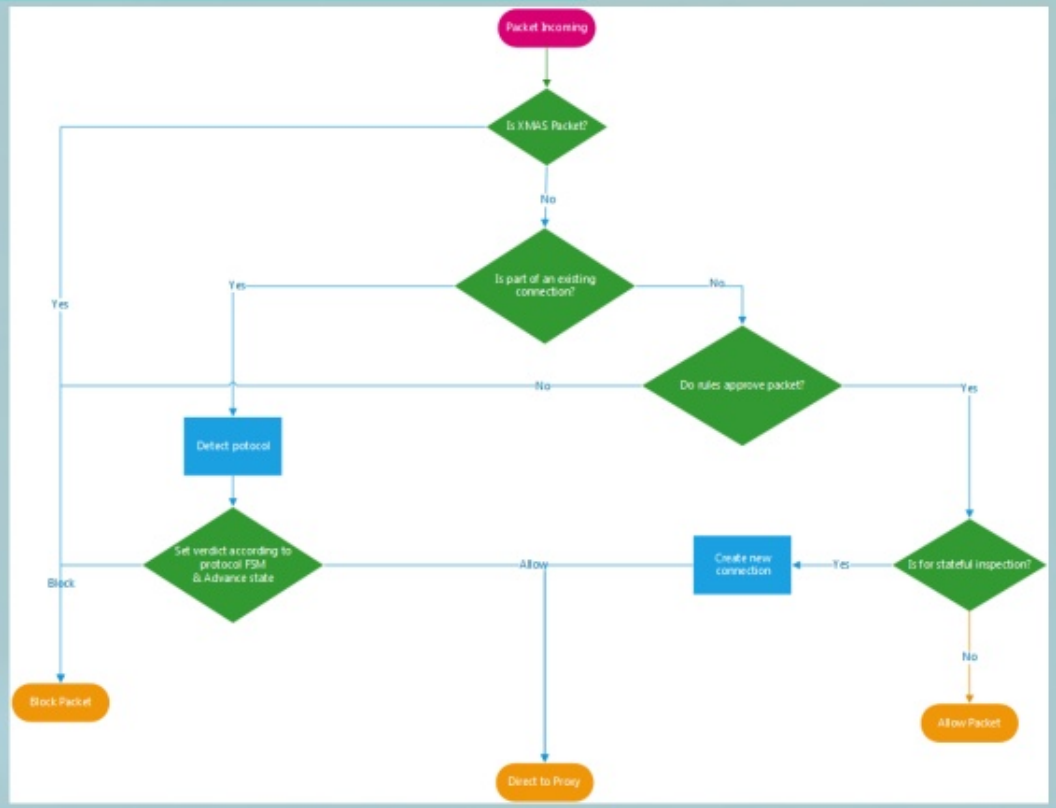
* Implementing stateful packet filtering
* Maintaining connection table
* Enabling FTP data connection
* Filtering specific file types - above known protocols

V Implemented proxy in application level for stateful inspection
V Become familiar and gain deep understandings of TCP & FTP

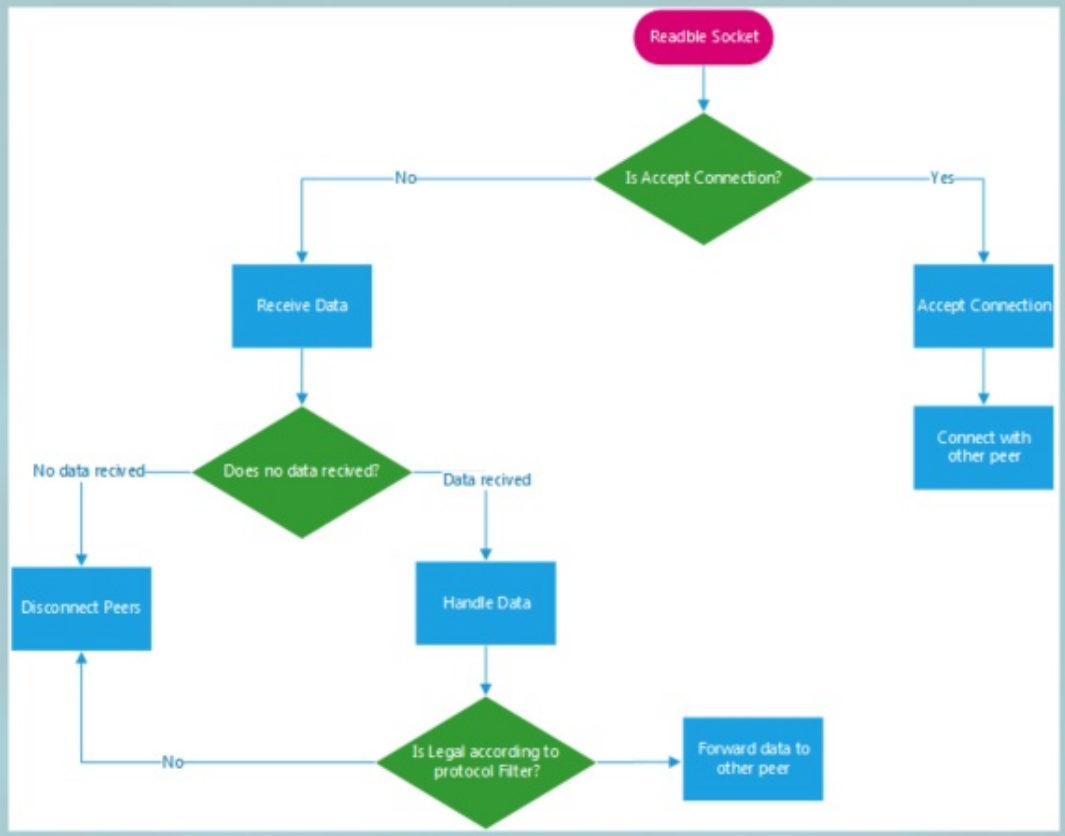
Handling incoming packet

Proxy

Handling incoming packet



Proxy


Readble Socket
Is Accept Connection?
No
Yes
Receive Data
Accept Connection
Connect with other peer
Does no data recived?
No data recived
Data recived
Disconnect Peers
Handle Data
Is Legal according to protocol Filter?
No
Forward data to other peer

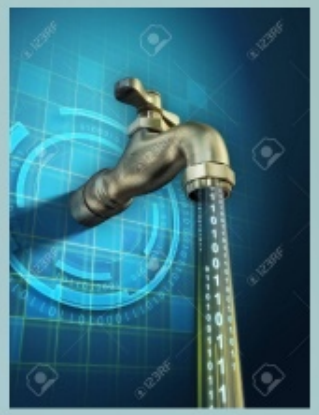

// Workshop In Information Security
Under supervision of Reuven Plevinsky @ Check Point
Aviv Yaniv / Tel-Aviv University
DD/MM/YY
Whats ahead of us?
First Stages
Firewall Architecture
DLP
LibSSH Auth Bypass
SSH

DLP

Goal:
* Protection against leakage of important code
* Code leakage is innocent, assuming no bad intentions

Keeping in mind:
V Must be fast (so users won't get upset)
V Avoiding false-positives
V Detecting code and blocking it - when necessary

Solution:
Protection of leakage of:
V C
V C++
V C#
V Java
V Python
code!



Protection

UML

What's Next?

Protection

1. Detecting is binary file: Scanning first [SIZE] bytes, counting printable VS. non-printable chars

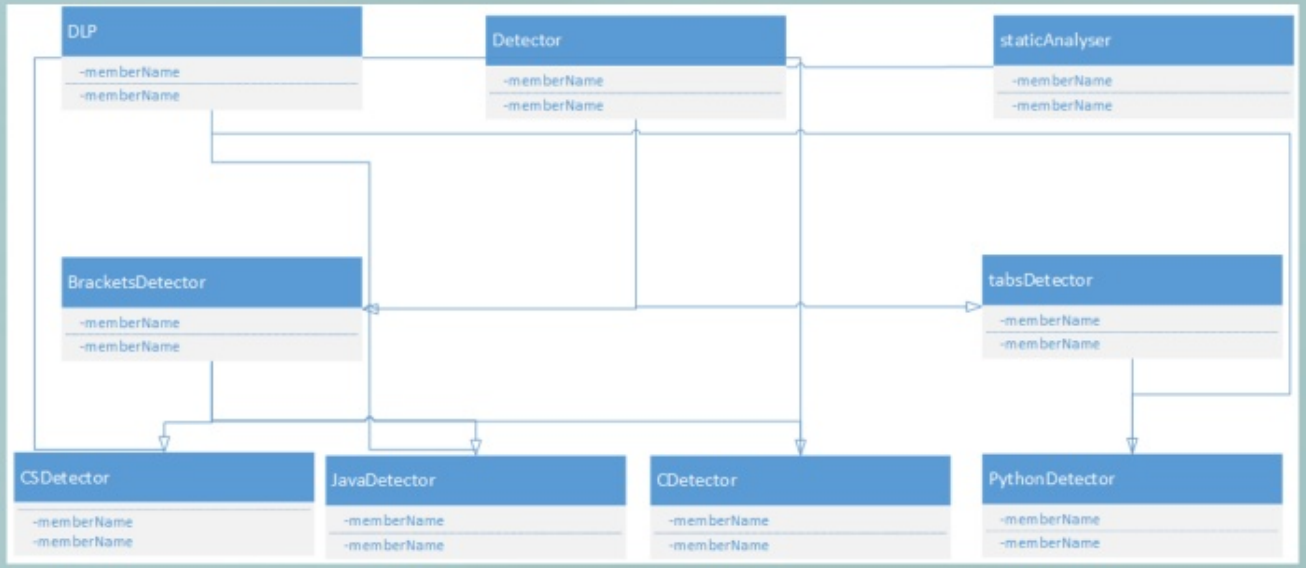
2. Each language has a detector, to find out if may be matching:
2.1. Stripping comments out
2.2. If both functions and logical structures (i.e. loops, conditions) have been recognized, mark as suspicious

3. Each suspecting detector runs over the code:
3.1. Separating it to code blocks.
3.2. Every block being scanned and ranked for having:
Data structures, Functions, Code line markers, Libraries, Actions (i.e. pointer access), tokens and reserved words.
3.3. Anomaly test: is documentation if way more functions than commands. Or small code, by small amount of commands. If either - being exonerated.

4. Maximal total rank among languages being compared to threshold - if exceeds, blocking.

UML

DLP module contains detectors for each language.
Detector inherits from brackets or tabs detector.
Detector uses static analysis on each block.



Data Leak Preventor

```
#### Main Section ####
class CDataLeakPreventor:
    # Configuration
    MINIMUM_COMMANDS_FOR_DETECTION  =       5
    DETECTION_THRESHOLD             =       100

    SANITY_CHUNK_SIZE               =       512

    PRINTABLE_THRESHOLD             =       0.7

    DETECTORS                       =       [ CDetectorC(), CDetectorCS(), CDetectorCPP(), CDetectorJava(), CDetectorPython() ]
    # DETECTORS                       =       [ CDetectorC() ]

    # Sanity
    def isBinaryFile(self, data):
        chunk = data
        if CDataLeakPreventor.SANITY_CHUNK_SIZE < len(chunk):
            chunk = data[:CDataLeakPreventor.SANITY_CHUNK_SIZE]
        printableCount = 0.0 + len([c for c in chunk if c in string.printable])
        return CDataLeakPreventor.PRINTABLE_THRESHOLD > (printableCount / len(chunk))

    # Methods Section #
    def detectCode(self, data):
        possibleLanguagesToStrippedData = {}
        # Statics analysing to detect language
        for detector in CDataLeakPreventor.DETECTORS:
            strippedComments = detector.stripComments(data)
            if DEBUG.DEBUG_WRITE_FILES:
                DEBUG.writeFileContent(DEBUG.STRIPPED_FILE_NAME_FRMT.format(detector.getName()), strippedComments)
            if detector.isMatching(strippedComments):
                possibleLanguagesToStrippedData[detector] = strippedComments
        # If no language detected based on static analysis
        if not possibleLanguagesToStrippedData:
            return None
        # Ranking language probabilities
        languagesProbabilitiesUnsorted = {}
        for d in possibleLanguagesToStrippedData:
            strippedData                        =   possibleLanguagesToStrippedData[d]
            languageRank                        =   d.getRank(strippedData, CDataLeakPreventor.MINIMUM_COMMANDS_FOR_DETECTION)
            languagesProbabilitiesUnsorted[d]   =   languageRank
        # Get maximal rank language
        maximalRankLanguage = max(languagesProbabilitiesUnsorted.iteritems(), key=operator.itemgetter(1))
        # If maximum rank dosen't exceed threshold - no language detected
        if CDataLeakPreventor.DETECTION_THRESHOLD > maximalRankLanguage[1]:
            return None
        # Returning name of detected language
        return maximalRankLanguage[0].getName()
```


// Workshop In Information Security
Under supervision of Reuven Plevinsky @ Check Point
Aviv Yaniv / Tel-Aviv University
DD/MM/YY
Whats ahead of us?
First Stages
Firewall Architecture
DLP
LibSSH Auth Bypass
SSH

First, a riddle...

Whats in common of Jeff Bezos & Verizon?

LibSSH bypass scanner

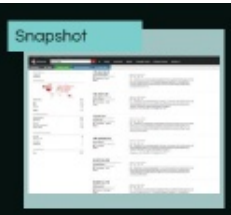
LibSSH bypass scanner

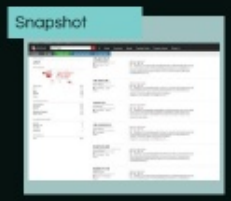
Digging Deeper

Sol

Fin

LibSSH Authentication Bypass





Before we continue...

1. First stage; scanner detects vulnerable servers according to their version

2. Second stage; attacking by sending the malicious message

Estimation: 3,000-4,000 min.
Lifetime: 2014 - 16.10.18

Terminology

SSH: Protocol secure remote login into Unix/Linux, in an encrypted manner

Libssh: multiplatform C library implementing the SSHv2 protocol on client and server side

LibSSH != LibSSH2 (Seems alike)
(Not common) LibSSH != OpenSSH (Common)
Putty as well is not vulnerable
GitHub is not vulnerable

Can Windows be vulnerable?

Terminology

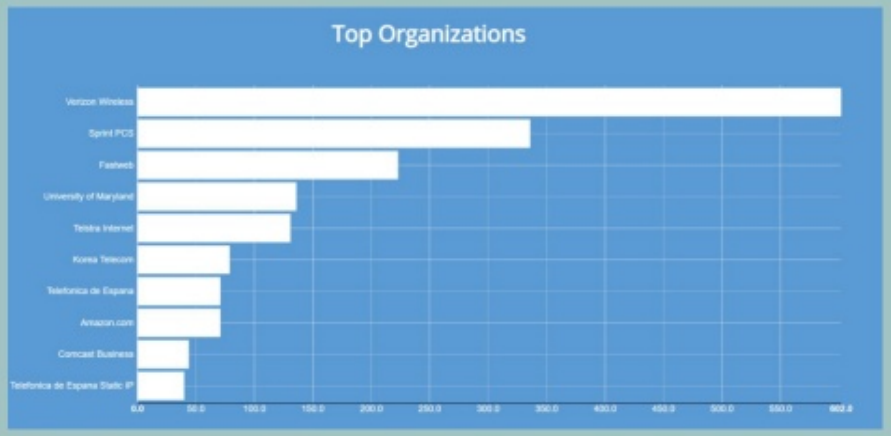
SSH: Protocol secure remote login into Unix/Linux, in an encrypted manner

Libssh: multiplatform C library implementing the SSHv2 protocol on client and server side

LibSSH != LibSSH2 (Seems alike)
(Not common) LibSSH != OpenSSH (Common)
Putty as well is not vulnerable
GitHub is not vulnerable

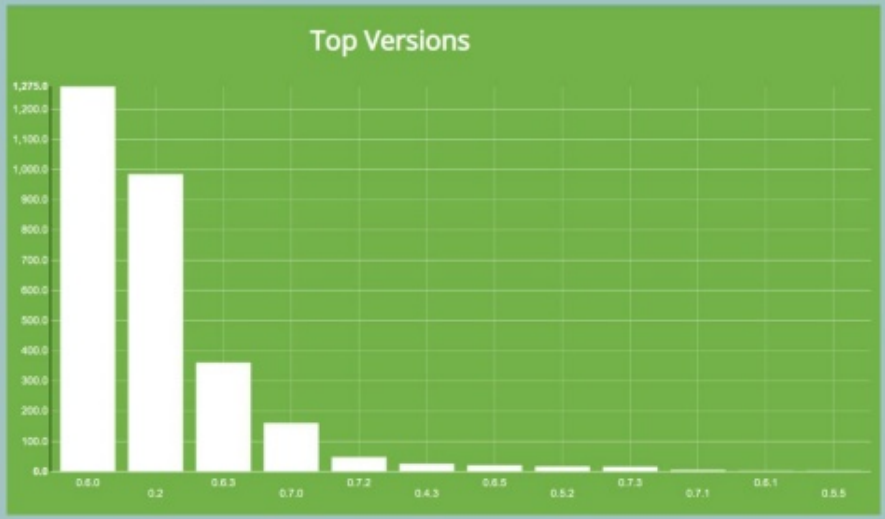
Can Windows be vulnerable?

Spread

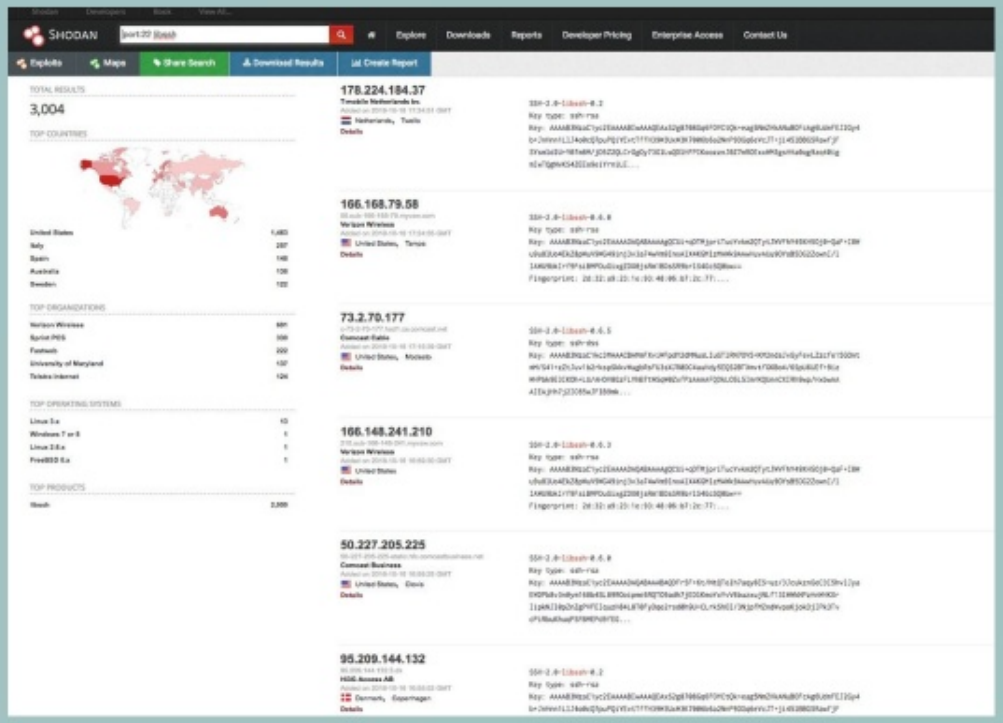

Top Organizations
Verizon Wireless
Sprint PCS
Fastweb
University of Maryland
Telstra Internet
Korea Telecom
Telefonica de Espana
Amazon.com
Comcast Business
Telefonica de Espana Static IP
0.0
50.0
100.0
150.0
200.0
250.0
300.0
350.0
400.0
450.0
500.0
550.0

Versions in use

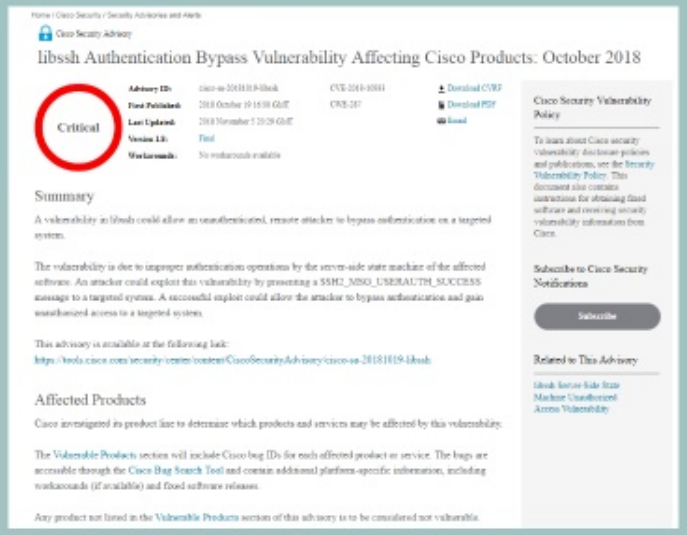
libssh versions 0.6 and above have an authentication bypass vulnerability in the server code.
libssh version 0.8.4 and libssh 0.7.6 have been released to address this issue.



Snapshot


SHODAN
Explore
Downloads
Reports
Developer Pricing
Enterprise Access
Contact Us
Exploits
Maps
Share Search
Download Results
Create Report
TOTAL RESULTS
3,004
TOP COUNTRIES
TOP ORGANIZATIONS
TOP OPERATING SYSTEMS
TOP PRODUCTS
178.224.184.37
166.168.79.58
73.2.70.177
166.148.241.210
50.227.205.225
95.209.144.132

Cisco



Home / Cisco Security / Security Advisories and Alerts

Cisco Security Advisory

libssh Authentication Bypass Vulnerability Affecting Cisco Products: October 2018

Critical

Advisory ID:	cisco-sa-20181019-libssh	CVE-2018-10933	Download CSRF
First Published:	2018 October 19 15:58 GMT	CWE-287	Download PDF
Last Updated:	2018 November 5 20:28 GMT		Email
Version 1.0:	Final		
Workarounds:	No workarounds available		

Summary

A vulnerability in libssh could allow an unauthenticated, remote attacker to bypass authentication on a targeted system.

The vulnerability is due to improper authentication operations by the server-side state machine of the affected software. An attacker could exploit this vulnerability by presenting a SSH2_MSG_USERAUTH_SUCCESS message to a targeted system. A successful exploit could allow the attacker to bypass authentication and gain unauthorized access to a targeted system.

This advisory is available at the following link:
https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20181019-libssh

Affected Products

Cisco investigated its product line to determine which products and services may be affected by this vulnerability.

The Vulnerable Products section will include Cisco bug IDs for each affected product or service. The bugs are accessible through the Cisco Bug Search Tool and contain additional platform-specific information, including workarounds (if available) and fixed software releases.

Any product not listed in the Vulnerable Products section of this advisory is to be considered not vulnerable.

Cisco Security Vulnerability Policy

To learn about Cisco security vulnerability disclosure policies and publications, see the Security Vulnerability Policy. This document also contains instructions for obtaining fixed software and receiving security vulnerability information from Cisco.

Subscribe to Cisco Security Notifications

Subscribe

Related to This Advisory

libssh Server-Side State Machine Unauthorized Access Vulnerability

Collaboration and Social Media

- Cisco Webex Meetings Server

Endpoint Clients and Client Software

- Cisco Jabber Guest

Network Application, Service, and Acceleration

- Cisco Adaptive Security Appliance (ASA) Software
- Cisco Cloud Services Platform 2100

Network and Content Security Devices

- Cisco ASA Next-Generation Firewall Services
- Cisco Content Security Management Appliance (SMA)
- Cisco Email Security Appliance (ESA)
- Cisco FirePOWER System
- Cisco Identity Services Engine (ISE)
- Cisco Web Security Appliance (WSA)

Network Management and Provisioning

- Cisco Elastic Services Controller (ESC)
- Cisco Enterprise Service Automation
- Cisco FindIT Network Manager
- Cisco NetFlow Generation Appliance
- Cisco Network Analysis Module
- Cisco Policy Suite
- Cisco Prime Access Registrar
- Cisco Prime Collaboration Provisioning
- Cisco Prime Infrastructure
- Cisco Prime Network Registrar Virtual Appliance
- Cisco Prime Network Registrar
- Cisco Prime Performance Manager
- Cisco WAN Automation Engine (WAE)

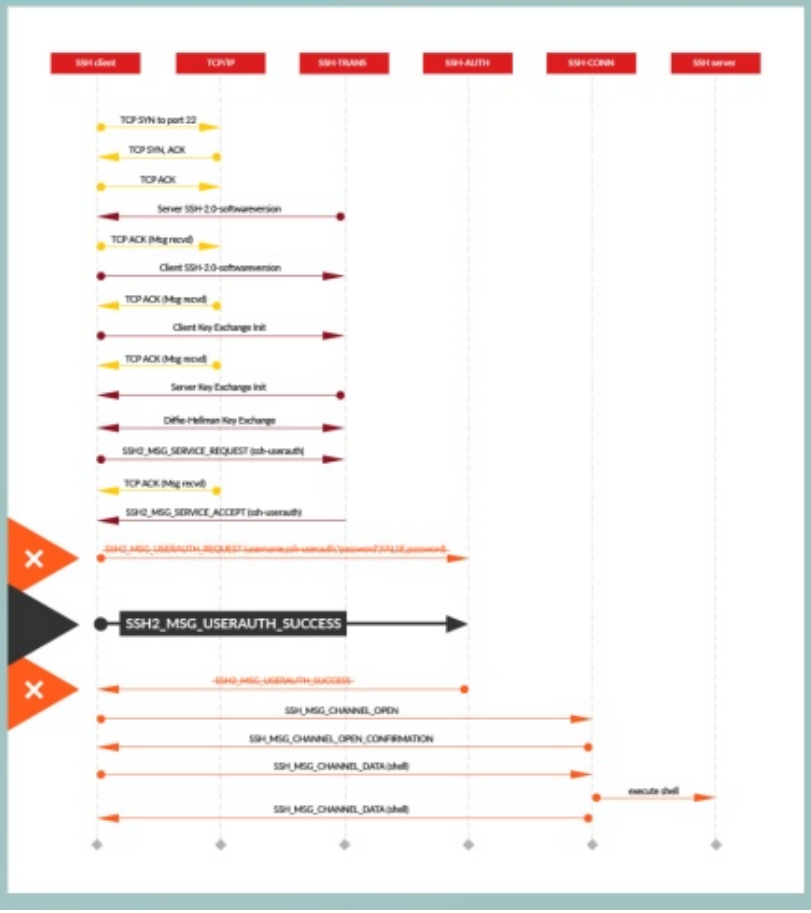
Routing and Switching - Enterprise and Service Provider

- Cisco Application Policy Infrastructure Controller (APIC)
- Cisco IOS XR Software for Cisco Network Convergence System 6000 Series Routers
- Cisco IOS XR Software
- Cisco Nexus 9000 Series Switches - Standalone, NX-OS mode
- Cisco Nexus 9000 Series Switches

Unified Computing

- Cisco UCS Director

LibSSH Auth Bypass



The Vulnerable Code

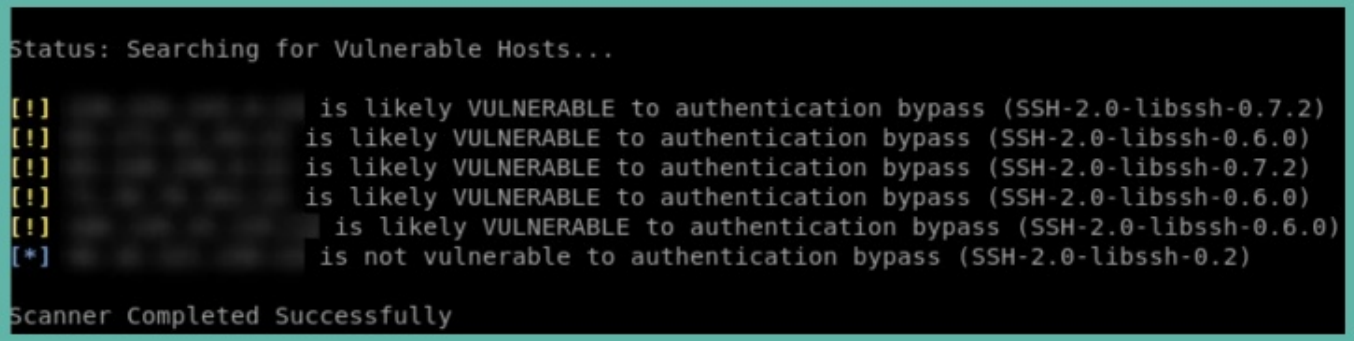
Scanner

The Vulnerable Code

```
1  SSH_PACKET_CALLBACK(ssh_packet_userauth_success) {
2    (void)packet;
3    (void)type;
4    (void)user;
5
6  //...
7    session->auth.state = SSH_AUTH_STATE_SUCCESS;
8    session->session_state = SSH_SESSION_STATE_AUTHENTICATED;
9    session->flags |= SSH_SESSION_FLAG_AUTHENTICATED;
10
11
12 //..
13 /* Reset errors by previous authentication methods. */
14     ssh_reset_error(session);
15     session->auth.current_method = SSH_AUTH_METHOD_UNKNOWN;
16   return SSH_PACKET_USED;
17 }
```

```
1  if (session->session_state != SSH_SESSION_STATE_AUTHENTICATED){
2      ssh_set_error(session,SSH_FATAL, "Invalid state when receiving channel open request (must be authenticated)");
3      goto error;
4    }
```

Scanner


Status: Searching for Vulnerable Hosts...
[!] [redacted] is likely VULNERABLE to authentication bypass (SSH-2.0-libssh-0.7.2)
[!] [redacted] is likely VULNERABLE to authentication bypass (SSH-2.0-libssh-0.6.0)
[!] [redacted] is likely VULNERABLE to authentication bypass (SSH-2.0-libssh-0.7.2)
[!] [redacted] is likely VULNERABLE to authentication bypass (SSH-2.0-libssh-0.6.0)
[!] [redacted] is likely VULNERABLE to authentication bypass (SSH-2.0-libssh-0.6.0)
[*] [redacted] is not vulnerable to authentication bypass (SSH-2.0-libssh-0.2)
Scanner Completed Successfully

Solution.... simple?!

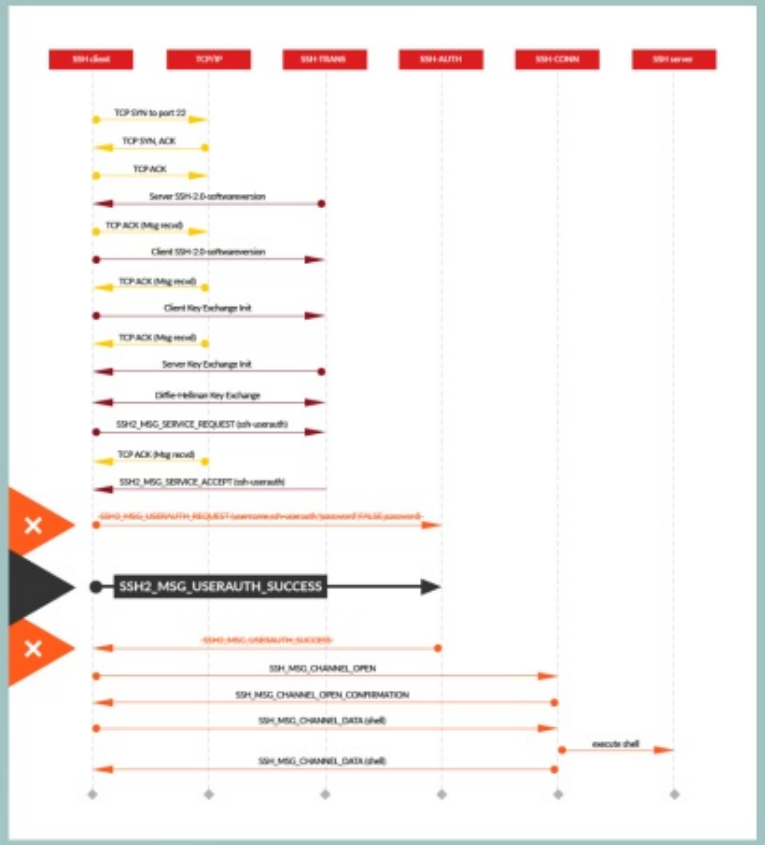
Analysis

Highlight

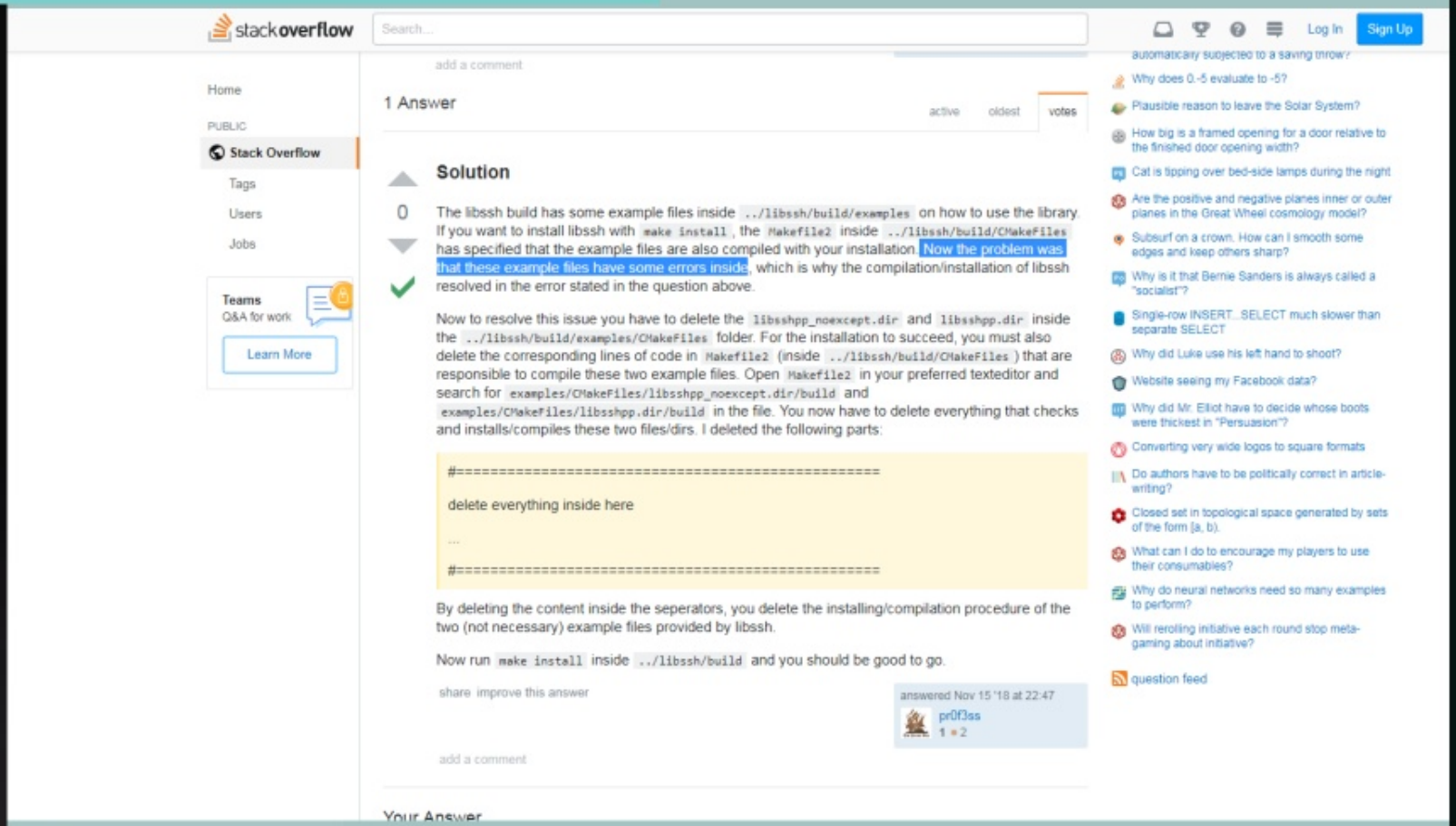
Highlight

Highlight

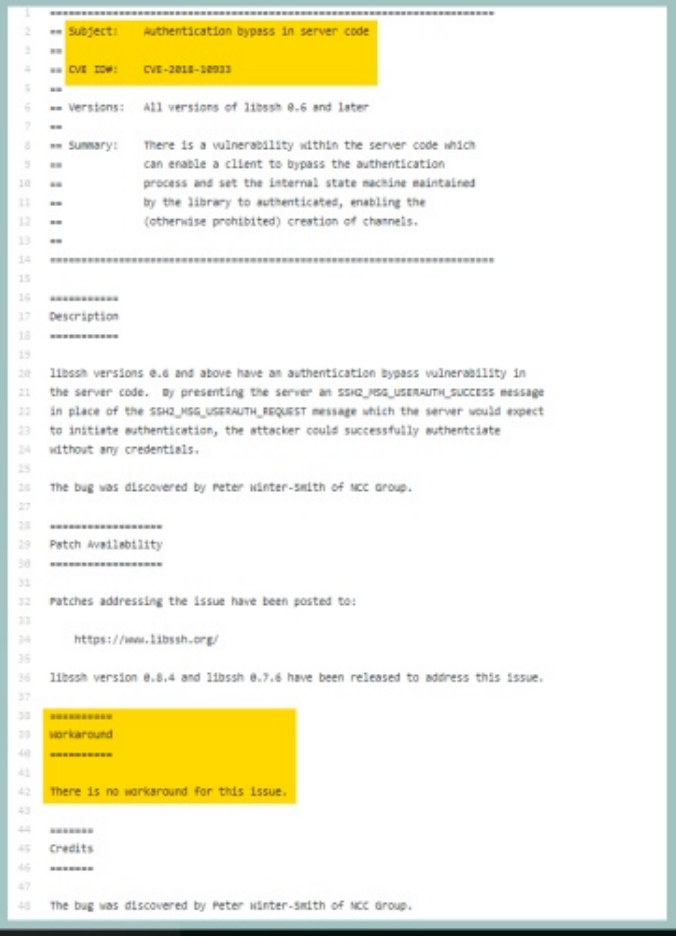
Not that fast!



Labor pains

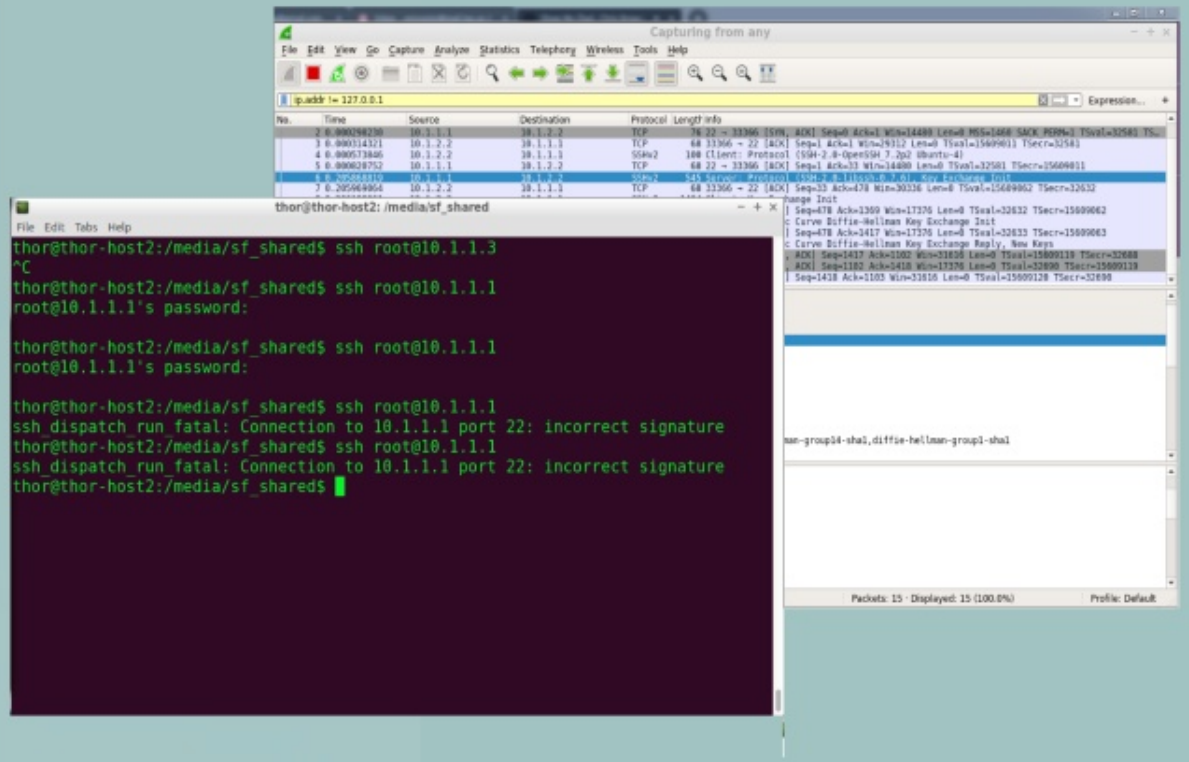

stackoverflow
Search...
Log In
Sign Up
Home
PUBLIC
Stack Overflow
Tags
Users
Jobs
Teams
Q&A for work
Learn More
add a comment
1 Answer
active
oldest
votes
Solution
0
The libssh build has some example files inside ../libssh/build/examples on how to use the library. If you want to install libssh with make install, the Makefile2 inside ../libssh/build/CMakeFiles has specified that the example files are also compiled with your installation. Now the problem was that these example files have some errors inside, which is why the compilation/installation of libssh resolved in the error stated in the question above.
Now to resolve this issue you have to delete the libsshpp_noexcept.dir and libsshpp.dir inside the ../libssh/build/examples/CMakeFiles folder. For the installation to succeed, you must also delete the corresponding lines of code in Makefile2 (inside ../libssh/build/CMakeFiles) that are responsible to compile these two example files. Open Makefile2 in your preferred texteditor and search for examples/CMakeFiles/libsshpp_noexcept.dir/build and examples/CMakeFiles/libsshpp.dir/build in the file. You now have to delete everything that checks and installs/compiles these two files/dirs. I deleted the following parts:
#==
delete everything inside here
...
#==
By deleting the content inside the seperators, you delete the installing/compilation procedure of the two (not necessary) example files provided by libssh.
Now run make install inside ../libssh/build and you should be good to go.
share improve this answer
answered Nov 15 '18 at 22:47
pr0f3ss
1 • 2
add a comment
Your Answer
Why does 0.-5 evaluate to -5?
Plausible reason to leave the Solar System?
How big is a framed opening for a door relative to the finished door opening width?
Cat is tipping over bed-side lamps during the night
Are the positive and negative planes inner or outer planes in the Great Wheel cosmology model?
Subsurf on a crown. How can I smooth some edges and keep others sharp?
Why is it that Bernie Sanders is always called a "socialist"?
Single-row INSERT...SELECT much slower than separate SELECT
Why did Luke use his left hand to shoot?
Website seeing my Facebook data?
Why did Mr. Elliot have to decide whose boots were thickest in "Persuasion"?
Converting very wide logos to square formats
Do authors have to be politically correct in article-writing?
Closed set in topological space generated by sets of the form [a, b).
What can I do to encourage my players to use their consumables?
Why do neural networks need so many examples to perform?
Will rerolling initiative each round stop meta-gaming about initiative?
question feed

CVE



```
================================================================================
== Subject:     Authentication bypass in server code
==
== CVE ID#:     CVE-2018-10933
==
== Versions:    All versions of libssh 0.6 and later
==
== Summary:     There is a vulnerability within the server code which
==              can enable a client to bypass the authentication
==              process and set the internal state machine maintained
==              by the library to authenticated, enabling the
==              (otherwise prohibited) creation of channels.
==
================================================================================


==========
Description
==========

libssh versions 0.6 and above have an authentication bypass vulnerability in
the server code.  By presenting the server an SSH2_MSG_USERAUTH_SUCCESS message
in place of the SSH2_MSG_USERAUTH_REQUEST message which the server would expect
to initiate authentication, the attacker could successfully authentciate
without any credentials.

The bug was discovered by Peter Winter-Smith of NCC Group.

================
Patch Availability
================

Patches addressing the issue have been posted to:

    https://www.libssh.org/

libssh version 0.8.4 and libssh 0.7.6 have been released to address this issue.

==========
Workaround
==========

There is no workaround for this issue.

=======
Credits
=======

The bug was discovered by Peter Winter-Smith of NCC Group.
```

Version renaming



Solution!

Implemented a transparent SSH proxy, integrated within the FW

Client believes he is talking with the server
Server thinks he talks with the client

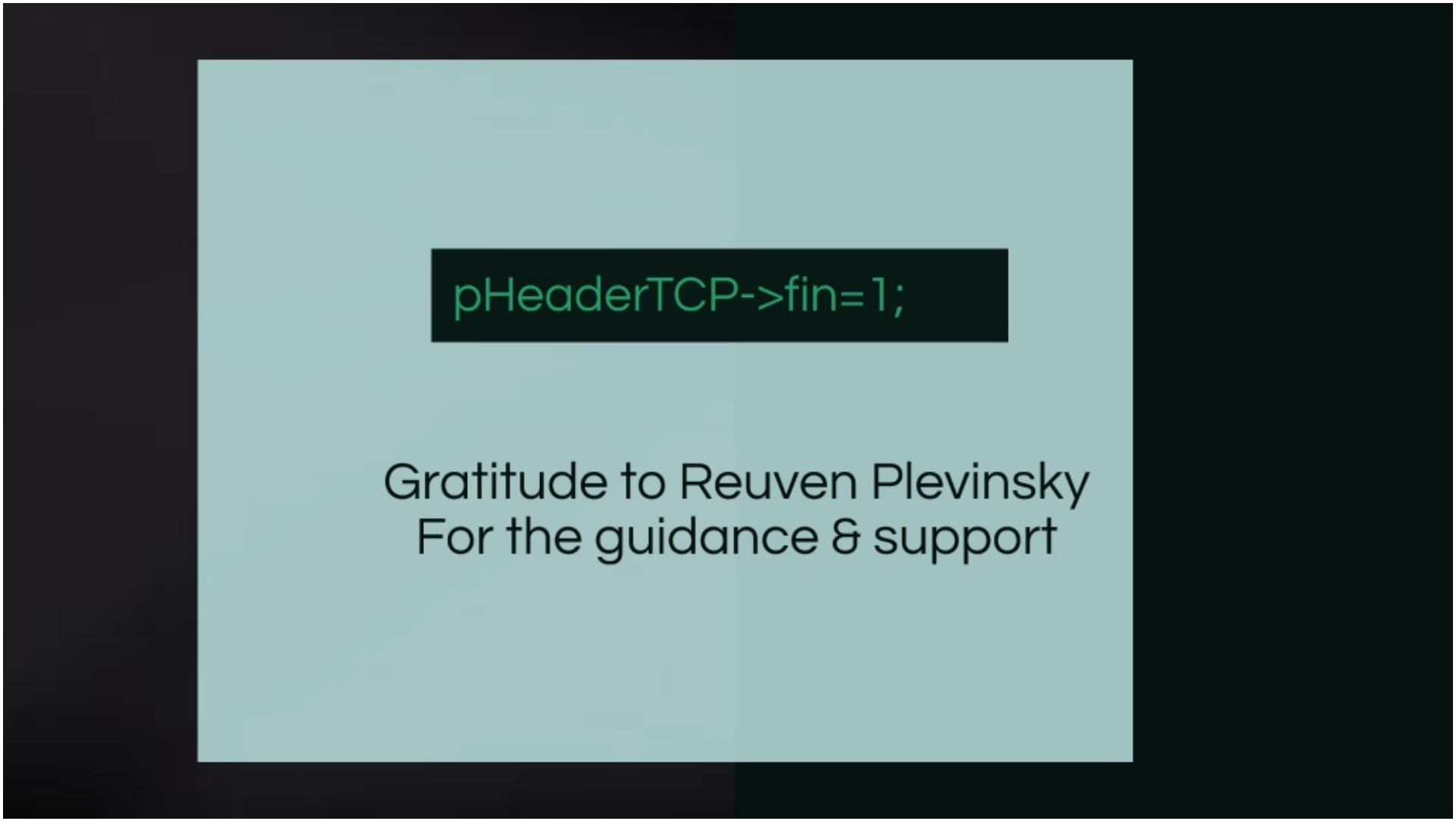
But...

We exchange encryption keys with both of them,
while listening carefully to all communication

MITM

V Authentication enforcement
V Full control over data passed
V Hindering attacker from scanning "real" server version
V Secured version in proxy

- COMPLETELY SECURED -


pHeaderTCP->fin=1;
Gratitude to Reuven Plevinsky
For the guidance & support


// Workshop In Information Security
Under supervision of Reuven Plevinsky @ Check Point
Aviv Yaniv / Tel-Aviv University
DD/MM/YY
Whats ahead of us?
First Stages
Firewall Architecture
DLP
LibSSH Auth Bypass
SSH