

TCP Stealth

Port Knocking Advanced

Dr. Udo Seidel

Digital Evangelist & Chief Architect
useidel(at)amadeus(dot)com



Agenda

- Who & why?
- A bit of history
- The idea
- The implementation
- Show
- And?



Who & Why

Me :-)

- Teacher of mathematics and physics
- PhD in experimental physics
- Started with Linux in 1996
- Linux/UNIX trainer
- s+c: 2002-2005
- @Amadeus:
 - Linux Admin, Strategy, Automation
 - Architecture & Technical Governance



Why not?

- Security by obscurity
- Port knocking daemon – SPOF
- IP address spoofing
- ... and more



Why?

- Brute force login attacks
- Technical evolution
- Snowden disclosures
 - Computer power
 - Country-wide port-scans



A Bit of History

Looking back and around

- Not new
- Several implementations
- Normally 3 parties
- 'Knocking' differences
- E.g. <http://www.portknocking.org/>

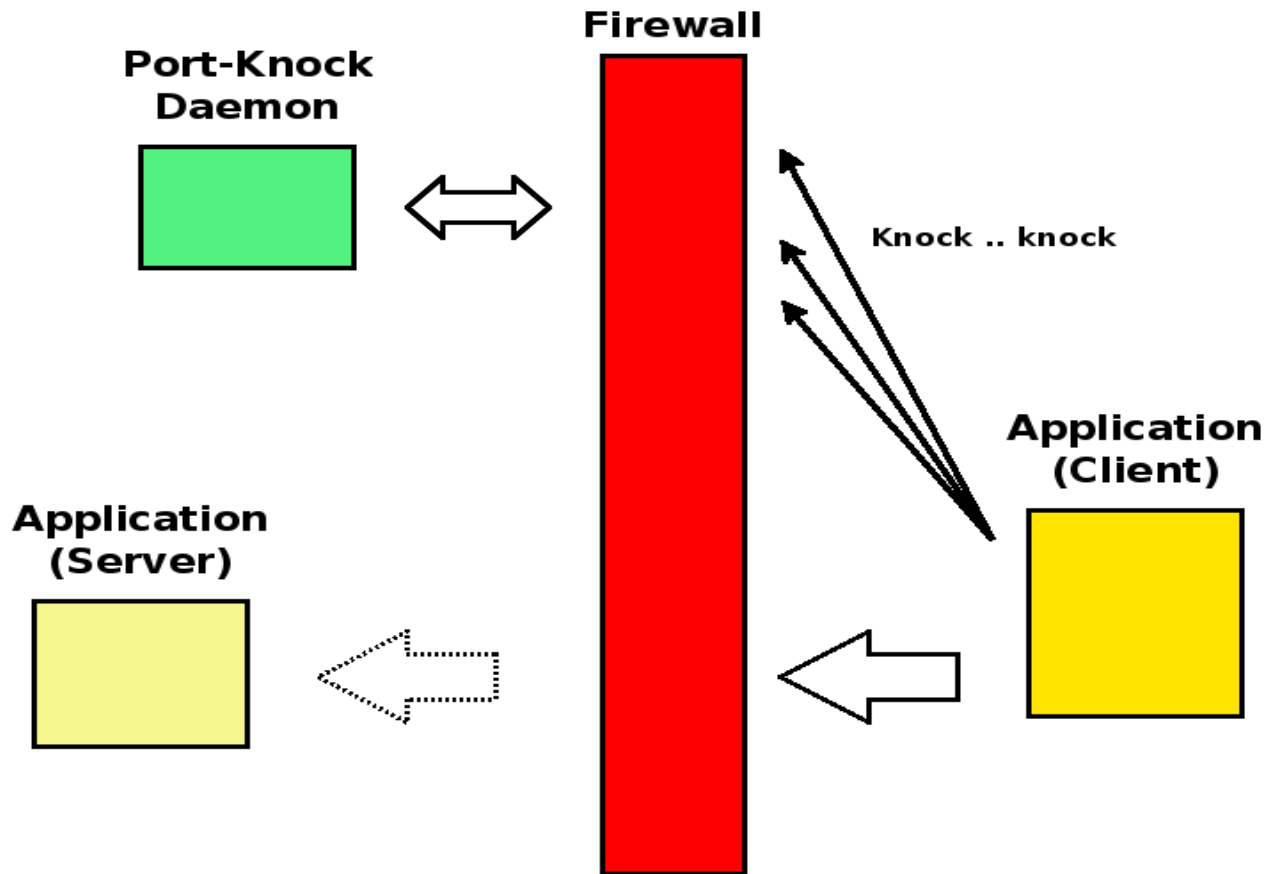


Traditional Port Knocking Procedure

- I. Connection attempt by client to server => blocked by firewall
- II. Knock sequence performed by client => verified by Port Knocking daemon
- III. Port-Knocking daemon instructs firewall
- IV. 2nd Connection attempt by client => granted by firewall



Traditional Port Knocking Setup



Further downsides

- Additional network packets
- Man-in-the-Middle attacks
- Protocol-dependent port behaviour



The Idea

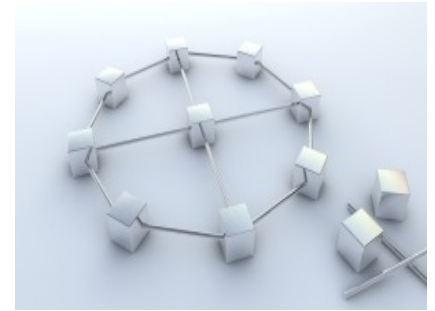
Project roots

- TU Munich
- Previously called 'Knock'
- Summer course 2013
- Master thesis 2014



TCP only!?!

- Well-used
- UDP limitations, e.g syslog, DNS
- Widely used for TLS/SSL

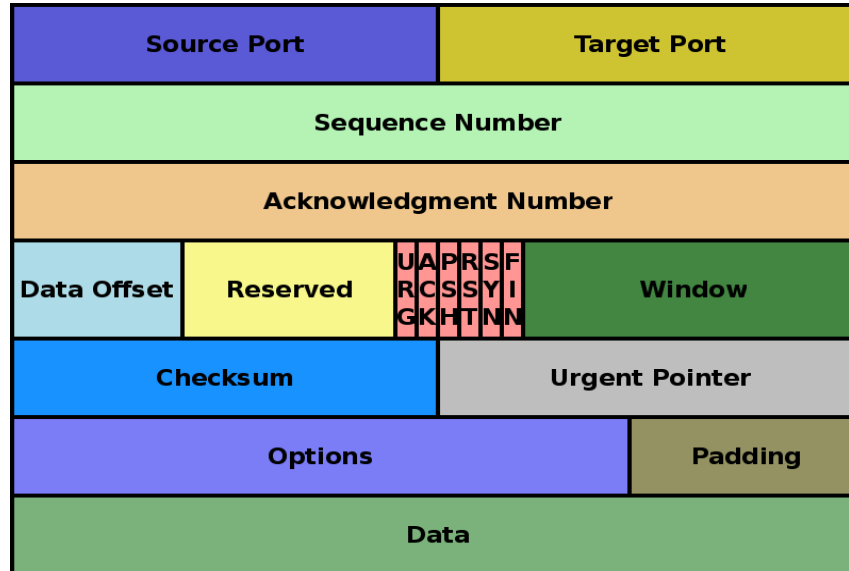


First Thoughts

- Avoid additional packets
- Hide information in protocol header
 - 'Steganography'
 - Limitations per design
 - RFC 793 (and successors)
- Shared Secret
- Not new => Silentknock (1997)



TCP Header – potential candidates



- Initial Sequence Number (ISN)
- Acknowledgement Number (ACK)?
- Timestamps?

ISN as a Carrier Medium

- 32 Bit
- Input data
 - Shared Secret
 - Target IP
 - Target Port
 - Time
- Generation via MD5



Implementation

About Sockets

- Change of TCP implementation
- Client AND Server
- Linux/BSD
 - Kernel
 - tcp_v4/6_connect()
 - tcp_v4/6_do_rcv()
 - Application
 - setsockopt()



Linux Kernel with TCP Stealth

```
.config - Linux/x86 3.18.0 Kernel Configuration
Networking support Networking options
Networking options
Arrow keys navigate the menu. <Enter> selects submenus ---> (or
empty submenus ----). Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*]
(-)
[*] TCP: advanced congestion control --->
[*] TCP: MD5 Signature Option support (RFC2385)
[*] TCP: Stealth TCP socket support
<*> The IPv6 protocol --->
[*] NetLabel subsystem support
(+)
<Select> < Exit > < Help > < Save > < Load >
```

Socket changes for closed source

- Helper library
 - *libknockify*
 - *LD_PRELOAD*
 - Overlay of system calls
 - `listen()`
 - `connect()`
- A bit flaky
- Not recommended => 'just a hack'



Stealth Socket Example

```
$cat tcp_stealth_server.c
..
#define TCP_STEALTH 26
..
char secret[64] = 'ThisismagicID!';
..
if (setsockopt(sock, IPPROTO_TCP, TCP_STEALTH, secret, sizeof(secret))){
    printf("setsockopt() failed, %s\n", strerror(errno));
    return 1;
}
..
$
```

Man in the Middle .. Still possible!?!

- Duty of application .. yes .. but ..
- Small but real attack vector
- Data integrity check!!



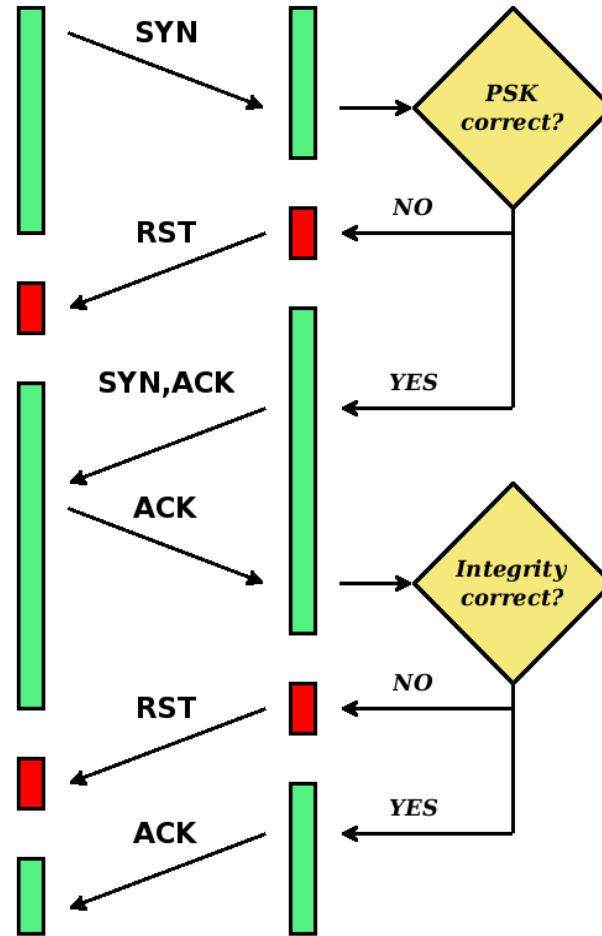
TCP Stealth – Data Integrity Check I

- Input data
 - PSK
 - Additional information
- MD5^{*)}
- Split and XOR => 16 Bit
- 2nd part of ISN :-)



*) Debateable!

TCP Stealth – Data Integrity Check II



TCP Stealth – Retransmissions

- Traditional TCP => new timestamp
- TCP Stealth
 - Timestamp => ISN calculation
 - Use same/old timestamp



Show

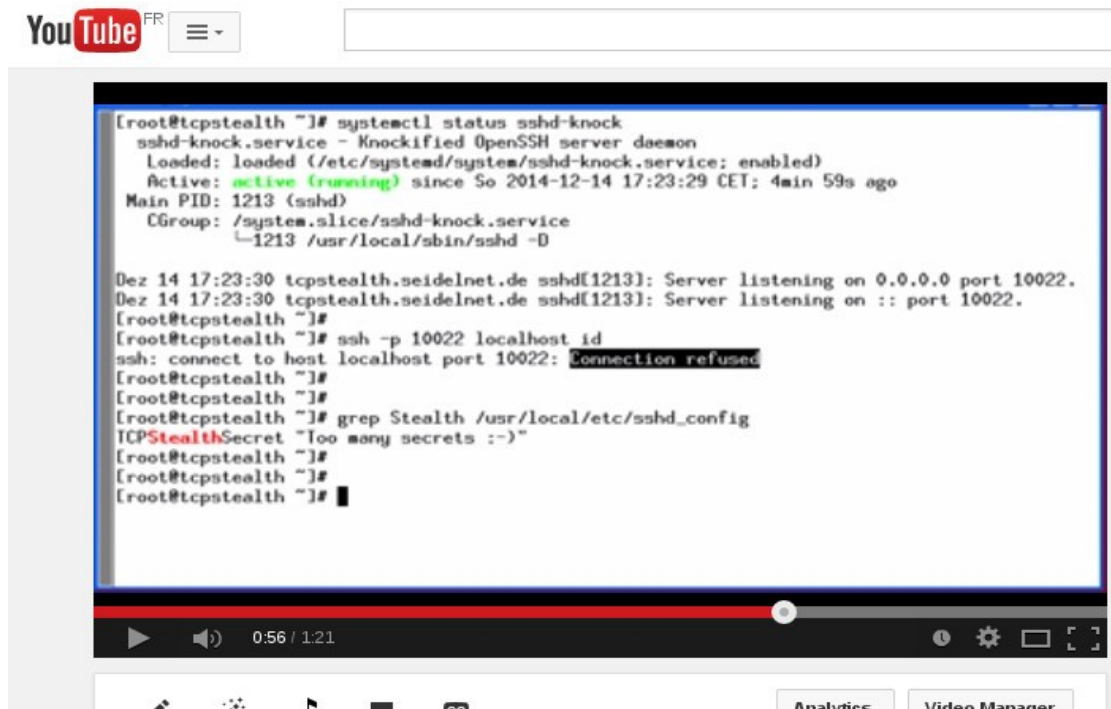
Example – The Scene

- Patched Linux kernel 3.19rc1
- Patched SSHD with TCP Stealth enabled
- Configured as service using port 10022

```
#  
# uname -r  
3.19.0-rc1  
#  
# cat /etc/systemd/system/sshd-knock.service  
[Unit]  
Description=Knockified OpenSSH server daemon  
After=network.target sshd-keygen.service  
Wants=sshd-keygen.service  
  
[Service]  
EnvironmentFile=/etc/sysconfig/ssh  
ExecStart=/usr/local/sbin/sshd -D $OPTIONS  
ExecReload=/bin/kill -HUP $MAINPID  
KillMode=process  
Restart=on-failure  
RestartSec=42s  
  
[Install]  
WantedBy=multi-user.target  
#  
#  
# █
```

Example – The Show

- <http://youtu.be/7CadOVTNxr4> :-)



And?

Bits and Pieces

- Not part of Vanilla kernel
 - First trial in DEC 2013
 - Last one in DEC 2014
- No plans by Linux distributors
 - True for Enterprise
 - True for Community :-(
 - Not true for Knoppix :-)
- IETF mailing list discussions
 - Not using ISN
 - Using Tsva/Tsecr
 - No real follow-up yet



Summary

- Promising project
- Good documentation
- Addresses known challenges
- Upstream integration outstanding ...
... and important if not crucial



References

- <http://gnunet.org/knock>
- http://gnunet.org/sites/default/files/ma_kirsch_2014_0.pdf
- <http://tools.ietf.org/html/draft-kirsch-ietf-tcp-stealth-01>
- <http://github.com/useidel/knock>



Check it out
Provide feedback
<http://gnunet.org/knock>

Thank you.



Unpublished Work of SUSE LLC. All Rights Reserved.

This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

