CÖK - Cryptographic One-Time Knocking

David Worth - cesium@hexi-dump.org
http://www.hexi-dump.org

Have you seen me?

"Port Knocking" For Added Security

Posted by CmdrTaco on Thu Feb 05, '04 03:03 PM

from the thats-a-crazy-idea dept.

Jeff writes *"The process of Port Knocking is a way to allow only people who know the "secret knock" access to a certain port on a system. For example, if I wanted to connect via SSH to a server, I could build a backdoor on the server that does not directly listen on port 22 (or any port for that matter) until it detects connection attempts to closed ports 1026,1027,1029,1034,1026,1044 and 1035 in that sequence within 5 seconds, then listens on port 22 for a connection within 10 seconds. The web site explains it in some detail, and there is even an experimental perl implementation of it that is available for download. I can't think of any easy ways you could get around a system using this security method - let alone even know that a system is implementing it. Another article on port knocking is here."*

CÖK

BlackHat USA • EUROPE • ASIA USA 2004

Port-Knocking: Define, Defend, Attack

Define: Port-Knocking originally described a means of passing a shared secret from an arbitrary host to another, generally "secure", host. This shared secret was nothing more than a (short) sequence of `connect(2)` calls to a sequence of ports, at which point the firewall would be opened to the sending host.

i.e. `31335,31336,31337 -> Open Sesame, you're` *`SO`* `|337!`

Attack: This system is vulnerable to a trivial replay attack. Some port-knocking systems which use cryptography to protect themselves from this attack use their source IP as part of their encrypted payload to specify the host to which the firewall should be opened; for the port-knock to be successful in a NATed context, a WiFi hotspot for example, the external address, which acts as the source address of the NAT provider, must be opened, at which point **any** user of the hotspot has access to the same service without replaying, or the ability to replay later from the hotspot.

Defend: Applications of port-knocking include limiting access to important resources, using the port-knocking system as a gate-keeper, and those pesky replay attacks can be foiled via cryptographic techniques...

Cryptographic Techniques Employed in Port-Knocking

By using appropriate cryptographic techniques we can prevent replay attacks. Shared secrets are a bad idea, so we use a more appropriate system...

One candidate for such a system is One-Time-Passwords (OTP A.K.A. S/Key). OTP was designed for insecure transport media (rlogin/telnet actually). OTP's resilience to replays is based upon the cryptographic hash function one chooses to use (MD5 or SHA1 traditionally), and the *pre-image resistance* of that has function.

S/Key is defined in RFC 1760, and OTP in RFC 2289.

30 Second Introduction to One-Time-Password Schemes

To generate n one time passwords one simply iteratively computes the cryptographic hash function, $f(x)$, on the output of the previous step:

0	$p :=$ "password"
1	$f(p)$
2	$f(f(p))$
$\vdots$	$\vdots$
n	$f(\cdots f(p))$
$n+1$	$f(f(\cdots f(p)))$

The server begins by storing the $(n+1)^{th}$ iteration of the function. To authenticate, one provides the $N := n^{th}$ iteration, the server calculates $f(N)$, and compares it to the stored $(n+1)^{th}$ iteration. If they match the server authenticates the user, and stores the n^{th} iteration for the next round of authentication. In this way the system works backwards through the n passwords calculated initially.

CÖK

We can *STOP* Replay Attacks with OTP

Replay attacks fail by virtue of OTP's design; replaying a previous password means that extra iterations of the hash-function are effectively computed, and the comparison step auto-magically fails.

Detection of attempted replay attacks is also simple: collect valid one-time passwords in a hash, and when you receive an invalid password, check if it is in the hash. If it is, then someone is attempting to replay a previous password, and appropriate action can be taken against the attacker (i.e. block them entirely @ the firewall, `nmap(1)` them, DoS, 0-day, etc...)

CÖK

BlackHat
USA • EUROPE • ASIA
USA 2004

Welcome to CÖK Country - a Brief introduction to CÖK

CÖK is an implementation of an OTP-based port-knocking system, written in Java with JPCAP (a JNI wrapper to libpcap). The primary components of CÖK are:

COKd The daemon which does all the listening and tracking of knocks

COKtool The configuration tool which interacts with COKd via RMI

COKnocker The actual knocking tool (though there exist knock types which do not require COKnocker to generate them)

Currently three knock types are implemented, more can be at a later time:

OTP Knock One-Time-Password knock, sent via COKnocker in a UDP packet

DNS Knock Form of a one-time-password knock in which the knock is sent via a DNS lookup to a given listening host (most likely to avoid detection if the listening host is also a DNS server)

Port Sequence Knock Traditional, replayable, sequence of tcp connections, implemented for the sake of completeness.

CÖK

What Can CÖK Do?

Answer: *Anything!*

CÖK is not limited by some set of pre-defined commands it can run; in fact, it can run anything your system can run. Parsing of rules occurs as follows:

1. Certain pre-processor macros: `__SRC_IP__`,`__SRC_PORT__`,`__DEST_IP__`, and `__DEST_PORT__` are replaced with the source IP, source port, destination IP, and destination port respectively, and `__KNOCKDESC__` is replaced with a textual representation of the knock.
2. The rule is then checked for a leading execution macros, of which there are currently two: `__LOG__` and `__PRINT__` (deprecated), which log to the syslog server, and print to `STDOUT` respectively. The parameter to the execution macro is the text following the directive.
3. If no execution macro is specified, then the rule is assumed to be a command, which is then executed on the system.

CÖK

Where does CÖK Fit in the Context of Network Security?

Short Answer: It provides an additional layer of trust for important network resources.

Long Answer: Given the dynamic nature of modern network security, providing the least possible access to would-be attackers is a "best-practice" that should always be followed. Even with traditionally trusted tools such as `sshd`, there is a potential for 0-day exploits, and compromise. To prevent this attack vector from being used, one may drop all packets destined for a sensitive network device, and only open them to authorized individuals via a cryptographic one-time port-knock.

Since the original context in which port-knocking was discussed was in terms of firewall management, it is often forgotten that other applications exist. Since CÖK allows for arbitrary commands to be run on the server, it can be used as an authenticated remote application gateway to run appropriate tools remotely and log their calls.

CÖK

Demonstration

Here we go...

The Future of Port-Knocking

Covert Knocks DNS Knocks are one implementation of a covert knock, but only if the server is running a DNS server (`bind`, etc...). We can also induce appropriate One-Time-Knocks to be generated by using a web browser, if the DNS server for a given domain is the target for our knocks. For example, if we run the DNS server for `foo.org`, on which we want to run a restricted service, we may run COKd, and point our browser to [OTP].foo.org. The DNS server will fail to resolve [OTP].foo.org, but COKd will act appropriately by processing the DNS query.

OOB Knocks Using an out-of-bound protocol, such as SMS, to transport knocks is perfectly reasonable and raises the ante for a man-in-the middle. To execute a MITM attack against SMS one must recognize that knocking is occurring via the SMS network, and compromise it. This is not necessarily more difficult that compromising a router in the tcp/udp context, it just requires more information.

CÖK

Bibliography and Links

References

[1] Krzywinski, M. 2003. "Port Knocking: Network Authentication Across Closed Ports." SysAdmin Magazine 12: 12-17.

[2] www.portknocking.org - http://www.portknocking.org

[3] Menezes, Alfred J, et al. Handbook of Applied Cryptography - http://www.cacr.math.uwaterloo.ca/hac/

[4] Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C (Second Edition). New York: Wiley, 1996.

[5] RFC 2289 - A One-Time Password System

Useful Links:

jpcap - http://jpcap.sf.net
jotp: The Java OTP Calculator - http://www.cs.umd.edu/~harry/jotp
authpf - http://www.openbsd.org/faq/pf/authpf.html

Other Reading:

The openbsd-misc mailing list (http://monkey.org/cgi-bin/wilma/openbsd-misc) contains a thread entitled "Port Knocking on openBSD" and was begun on Thurs. Feb 05, 2004. This is good reading for background...