Global Information Assurance Certification Paper

Port Knocking: An Overview of Concepts, Issues and Implementations

Ben Maddock
Submitted for SANS GIAC GSEC Practical on 23rd September 2004
Assignment version: 1.4c

Abstract

Port-knocking is a stealthy method of information transmission across computer networks. It has been a source of interest of late thanks largely to an article by Martin Krzywinski[1].

This paper seeks to define port-knocking, examine why it might be useful, answer some objections and highlight some of the benefits. An overview of features in currently available implementations will illustrate some of the diversity this approach offers. Finally areas for future exploration will be offered and a conclusion will be drawn on the usefulness of port-knocking.

What is Port Knocking?

Martin Krzywinski, who is credited with much of the recent interest in this method of covert information sending, offers a fairly narrow definition on his port knocking site as follows: “Port knocking is a method of establishing a connection to a networked computer that has no open ports”[2]

However a contrasting example of port knocking could be as follows. IP packets are sent from a client to a predetermined sequence of closed ports on a firewall-protected host. A script on the host monitors the firewall logs and, when it recognizes the secret sequence, triggers some user-defined action on the host.

Thus, perhaps a broader definition would be more appropriate: “a method for delivery of information via closed ports on a networked computer”.

For those who haven’t come across port-knocking before a small concrete example of a problem and solution may be of use.

Assume Client A needs to provide information to Host B. However it is not convenient for Host B to have open ports. How can Client A communicate with Host B?

A solution using port-knocking might begin with certain ports being arbitrarily defined as representing certain values. For example ports 501-526 represent letters of the alphabet with port 501 representing ‘a’, 502 ‘b’ up to 526 ‘z’. Port ‘500’ represents begin/end for the message to be sent. In order to send the

test message ‘foobar’, Client A simply sends packets to Host B ports 500, 506, 515, 515, 502, 501, 518 and 500 respectively.
For the purpose of this example (and most simple port knocking implementations) Host B must be logging every attempt to access any port in the port range 500-526 through its firewall rules. A simple script which is monitoring the firewall log on Host B notes a packet arriving aimed at port 500 from Client A and then watches for further attempted connections from Client A in the relevant port range. After port 500 it should detect 506, 515, 515, 502, 501, 518 and 500. The script then decodes the message using the agreed protocol and reverses the encryption to reveal the message: “begin, f o o b a r, end”.

There are several things to note in this example. Once an attempt is received on port 500, the script must watch for all subsequent attempts from that IP. An obvious (and potentially large) limitation is that there is really no way to ensure the packets arrive in order[a] or even at all. Time delays for each subsequent packet would help to alleviate this (but would impact the speed of overall information transmission), as would the concept of beginning and ending the message with a special port/packet.

Unfortunately there is no way ensure the reliability of this method of transmission in the same way as an end to end TCP connection, where path, packet order and error checking are all available. Fundamentally the message, whatever it is, is still being transmitted in ‘clear-text’ which makes eavesdropping trivial.

So what is the Point?

That’s all very interesting but if someone needs to send a host a message like ‘foobar’ there are many less convoluted and more reliable ways to do it.
As mentioned above, the usefulness of port-knocking really relies in the ability to communicate with a server which has no open ports/services.

The most obvious application where this is desirable and the one Krzywinski and most recent port-knocking advocates put forward is getting the ‘silent’ server to make a service temporarily available.

Suppose a user has a server at home which is essentially used ‘standalone’ and is normally only accessed via the console.

In this scenario there may be little need for services to be run allowing remote access. If a user wishes to have the ability to login occasionally from a remote location (say their place of work), it is possible to configure the server to respond to a port-knocking sequence in order to temporarily allow remote access through ssh.

[a] Short discussion later in the paper about a way around this using altered sequence numbers in TCP packets. (The Future of Port Knocking 4th point)

In order for this to happen a ‘secret’ knock must be predefined (on host and the client) for port knocking to work. For example ports 500, 498, 500, 502 and 560 must be pinged (with ICMP packets) in sequence to activate the listening script on the host. Again, a script runs in the background monitoring the logs for attempts to access these closed ports. Once it detects a particular IP address attempting to access these ports in sequence (within say 10 seconds) it is configured to enable ssh on port 22 for 3 minutes.

So the user at work can ping their remote host with this combination of pings/knocks 500, 498,500,502,560 and then has 3 minutes to connect to ssh:22 on the remote host. After 3 minutes the ssh service is disabled again, but by then the user has established their connection and the ssh service is unavailable to anyone else who attempts to connect on port 22.

One thing to note is that it is generally a bad idea to use sequential or consecutively increasing port numbers as there is an immediate and real risk of a port scan ‘discovering’ the knock sequence.

Isn’t This All Just Security Through Obscurity?

A key point of debate surrounding port knocking is whether it is just another form of security through obscurity. To examine this properly a definition of security through obscurity is required.

Security through Obscurity in the computer security context is generally defined as securing something by ‘hiding’ or not making known its design.[3] As one Slashdot poster put it, it relies on people not knowing HOW a given security method works.[4] [5]

Jay Beale provides the following example in his excellent security through obscurity paper. A Web Server for an organisation which holds secure data and perhaps runs the web service on a non-standard port and/or uses long URL’s for the content is an example of security through obscurity in the generally understood IT context.[6]

It does not offer any discrete access control of who gets access to the document (the long URL is the lone ‘password’). Nor is there any way to track users who may be attempting to find the document by ‘guessing’ the URL. Most robust security methods have provision for at least these features.

It is important to note that security through obscurity is not universally viewed as being bad in and of itself, the problem occurs when an organisation or individual uses obscurity as the *only* method to secure access to resources or data without sensible access control provisions in place (such as passwords). A layered approach (defense in depth) is obviously much better than a single level of control placed on a resource or group of resources for reasons described widely in current computer security literature[7].

However a password can also be viewed as just an example of security through obscurity, a token which a user knows and an attacker doesn't which is used to gain access resources[8]. As mentioned above security through obscurity is defined slightly differently in an IT context, but it is clear that most forms of security or access control at some point use a secret or token that is unknown to the potential attacker.
The difference (it is argued) is that in these cases the security protocol is known. For example anyone can read on the Internet how the DES encryption algorithm works, anyone with enough skill as a programmer can implement it. The difference is that if an attacker knows everything except the secret being used, they will still have difficulty attacking something secured using this encryption method with anything other than brute-force guessing or getting hold of a valid secret. There is no 'backdoor' by which someone with more knowledge than the user of this type of encryption can gain access to the encrypted resources. The fundamental difference of security through obscurity is that it potentially has such backdoors.

In this context then, port knocking serves to hide the services on the host until the correct combination of ports are pinged, especially if there is no logging of failed attempts or discrete access control.

Krzywinski argues[9] that by encrypting the knock, monitoring the logs and providing fine-grained access control (more than just the one arbitrary knock sequence for all users) port knocking succeeds in rising above the obscurity alone tag. A Slashdot user in the initial debate about port knocking proposed the knock is really analogous to a key.[10] These are both quite persuasive, but the debate has by no means been settled.[11]

Both sides of the debate agree that port-knocking on its own is not a valid security measure. However, advocates of port knocking argue that if used with other security options it is of some use in making it more difficult for attackers to discover the services initially before proceeding to attack them, defense in depth. Just as a house (that is otherwise secure) is better off with a security grill door as well as a deadlocked wooden door, so is a host which implements port knocking in front of services which in turn require their own separate authentication.

The Argument Against

It is worth reviewing and responding to the common arguments against port knocking currently put forward.

Port knocking is really just an extra password (in clear text[12]).

This is true for a simple knock. If someone is sniffing traffic heading to a host the knock sequence will be clearly visible from the packets being sent, just as a telnet password is clearly visible to anyone watching the TCP stream of a telnet session.

The response to this is to use encrypted and perhaps rolling sequences or one-time key's to generate valid knocks which your server understands.

Port Knocking engenders a false sense of security.

Similar to other security through obscurity techniques (or any security implementation for that matter), someone who uses port-knocking in tandem with ssh for example may not be as vigilant about patching ssh vulnerabilities because they believe the service is 'invisible' and therefore safe from casual attack.

There is certainly some merit in this, in the same way that users on a large private network behind a firewall (from the Internet) may not feel the need to patch as regularly because they're protected from the Internet. Unfortunately this ignores the fact that it is just as easy to get a virus or be attacked from someone 'inside' the private network. This is not directly a function of port knocking however. In the current computer security climate all administrators must be actively monitoring and making decisions about who and what is trying to connect to hosts they administer, not rely on passive blocks they may put in place.

Port knocking just adds complexity to either the firewall logs that need to be kept or the network stack to watch for port-knocks without really adding a comparable security benefit.

Some complexity is added, particularly for long sequential knocks and there may be a time delay to activate and connect to the service required. However, port knocking is comparable to challenge/response authentication combined with a password and many users have accepted the impost this requires. The question then becomes whether it adds enough extra 'security' to make up for the extra inconvenience? That probably depends more on the implementation than anything else. Some of the better implementations provide a significant degree of security but are still unwieldy for an average user. However there is no reason a decent plug-in to ssh or a VPN[13] for example could not be developed to make port knocking for an ssh connection relatively seamless from a user perspective.

Port knocking is vulnerable to DOS attacks/replay attacks/man-in-the-middle.

The concept of port knocking is indeed vulnerable to these attacks and they must be addressed through additional countermeasures. Further, there exists the possibility in simpler implementations of an attacker exploiting an indiscriminate knock daemon (which doesn't link knocks and IP) by randomly pinging ports, thus interfering with the knock sequence and denying access to a legitimate user.

Replay attacks are also a problem for simple static knocks, but can be mitigated by use of a one-time knock or by encrypting client information (IP) into the knock sequence.

Man-in-the-middle attacks are similar and may be effective even when employing encrypted or one-time knocks. The attacker doesn't need to know how a sequence is encrypted, they just need to capture a legitimate knock attempt and replay it to the host. The best defense against this again comes from Krzywinski's implementation which is to encode the IP address of the client into the knock before encrypting it.

Users accessing from public networks may be unable to knock the non-standard ports because they are firewalled in and are only allowed to use known ports.

This is certainly a problem, a solution (although convoluted) could be to trigger an external machine to provide the knock (a 3-way knock). Alternately packet-knocking on known allowed ports (encrypting content in packets sent to an allowed port rather than in port sequence).

The Argument For

Port knocking also has several features to recommend it.

It provides a covert channel for transmitting data.

As discussed elsewhere the data transmitted can be anything from a simple message, to an IP to a complete command to be executed. At no time is the host required to have an open port.

It allows remote access to a server with no open ports.

If using a knock to activate ssh for example, a legitimate user can still gain access to a host which otherwise does not betray its existence in any way.

It offers protection in depth from viruses/worms and script kiddies.

A worm/virus or other automated attack against a vulnerable service will be less likely to access the service if it is only activated when needed. A firewall configured to normally DENY/DROP everything will give an attacker nothing[b], so that anyone who tries to port scan a host will get nothing at all, the host is a 'black hole'.

[b] DENY/DROP as opposed to REJECT will not respond to any packet sent to a host:port combination which they are applied. REJECT will not allow access, but will respond that access is denied.

Potentially allows time to fix vulnerabilities in the services it protects.

Similarly, by adding a layer in front of services, if vulnerability is discovered in those services they are not immediately open to exploitation; an attacker still has to discover the knock sequence required to access the service. This may buy some extra hours to test the patch and apply it to the service.

Forces an attacker to be less stealthy

In order to discover a knock an attacker may have to make many attempts, it is possible that this is obvious as a knock sequence is less directed than a directed port connection.

Even if an attacker manages to discover a valid knock sequence they still have to attack the protected service. By protecting the service, it would be expected that there would be less connection attempts and consequently the monitoring of connection attempts to the service could be increased. The only people who will get to the services are those who've got past the port-knock, immediately an attacker is more exposed to discovery because there will necessarily be less connection attempts to the service.

Example Implementations

Currently there are quite a few examples which implement port knocking ideas. While most of them are little more than proof-of-concept there are several interesting ideas that have been incorporated into them which are worth reviewing.

As mentioned elsewhere in this paper Krzywinski's[14] effort is the most sophisticated. It incorporates a method to encode the IP address of the knocking client into a knock sequence which is then encrypted and mapped back onto a range of valid host ports.

Further, a recent innovation[15] reduces the number of knocks significantly by increasing the range of valid ports. Values to be transmitted are converted to 8-bit binary numbers, concatenated and then split into N-bit numbers where 2^N ports are valid knock ports on the host[c]. The numbers are converted back to decimal, encrypted and encoded into a valid knock sequence. The host does the reverse.

Other interesting but less sophisticated innovations include salting the knock sequence with current time and a shared offset to produce a knock sequence

[c] Another nice feature is that the ports can be non-contiguous.

which (allowing for time drift) is known to the client and host[16]. Fwknop is an implementation which combines OS Fingerprinting[17] with port knocking to grant access only to certain flavours of client[18]. Doorman[19] uses a single UDP client on a certain port which must contain an encrypted packet which the host can check, and which indicates which port the host should open.

cd00r[20] and SaDoor[21] are both older implementations, cd00r having been written some time ago[d]. Both listen at the network stack level rather than watching the firewall logs, both are implemented to be hard to detect on the host. cd00r watches for SYN packets to be sent to particular ports in order. The sequence resets if a port outside the sequence receives a packet or if a packet arrives from a different client. However this can be changed to allow packets from different sources and hence allows someone using cd00r to 'fly below the IDS radar'[22]. Once it receives the correct knock sequence it executes a user defined command.

SAdoor is similar, although it allows different kinds of packets to be sent, it its default configuration the third packet contains an encrypted command in the payload to execute such as a change to the firewall[23].

Both the final examples illustrate that while the computer security community debates the effectiveness of port knocking, blackhats have already been using it for some time[24]. From a blackhat's perspective, applications such as opening a mail relay or activating a DDOS zombie by knocking from a remote host would be appealing possibilities[e], as would the ability to close up a compromised host from further attack[25] [26].

The Future of Port Knocking

Undoubtedly there are many more implementations and variations on the port-knocking theme.

Several interesting ideas spring to mind for which implementations or further discussion was unavailable.

1. A port knock on Host A activates and creates a hole in the firewall (if needed) to a service on Host B. This may be most useful when one is external to the workplace firewall and is really an implementation of a poor-mans VPN (as was described to me by a colleague). Presumably this would no be difficult to implement on a double-homed host which is denying external visitors, but needs to talk to the internal machines. It is similar in concept to redirections now common on home-routers, where access to a particular port on the router redirects traffic to a particular host: port on the inside. Even easier would be to combine this with port-triggering[27] which is now appearing on some routers

[d] cd00r has been around since at least 2000.

[e] While there were several allusions in documents read to black hat port knocking activity there was nothing particularly concrete that I could find.

where an internal host activates a service and the router automatically generates a rule allowing external access.

2. Another idea is hinted at in the discussion on cd00r. That is the concept of a back-channel. So rather than a host being port-knocked into allowing ssh access to the host for the client. The host could be triggered to initiate a connection back the client. Obviously this is more risky if an intruder stumbles on a valid knock, but in the cd00r context when it is not necessarily desirable for the host to be connected at all, it could be an option.

3. Further investigation of combinations of encrypting ports and payloads would strengthen port knocking as a security measure in a similar way that username/password combinations are stronger than one or the other. Although the usefulness for the added complexity may not be of much value.

4. When using TCP packets some artificial manipulation of sequence numbers may assist in packet arrival order problems and may allow for scope to cope with missing knocks as well. Again this could be combined with the concept of packet knocking in an environment which does not let out packets aimed at non-standard ports. Packets could be sent to a standard port with a particular range of pseudo-random sequence numbers, although these kinds of numbers have limitations[28].

5. As mentioned above[f], it is likely that port knocking is already in use to trigger zombie hosts which have been compromised and have had a listener installed.

Conclusions

While the ideas behind Port Knocking are not particularly new (cd00r is testament to that), it is useful that it has been brought to the attention of the security community recently, if for no other reason than a review of the important security concepts of Security through Obscurity and Defense in Depth.

While it is difficult to view port-knocking as being a viable addition to large high-traffic server there is definitely scope with the development of relatively simple clients and daemons for low-traffic servers or home-users to enjoy the benefits of remote access on a limited basis without being immediately vulnerable to any and every form of automated attack. It should be noted that the concerns raised in regards to complacency are valid and any robust implementation for port-knocking should have a feature to force the regular alteration of knocks.

[f] Last paragraph of Example Implementations

As a defense in depth information security tool for low traffic servers, port-knocking does have something to offer. The concept of covert communication is one that will become increasingly important in the hostile internet environment, and as can be seen by previous implementations such as cd00r, there is a lot of scope for how to implement and use such covert channels.

Overall for a relatively small investment in complexity port-knocking returns a useful security dividend that doesn't require advanced knowledge of cryptography or network programming for an average user maintaining a couple of hobbyist servers. Even for a host with more secure requirements, such as an IDS system which by nature probably doesn't need to be broadcasting public services, port-knocking is an added layer of security which makes an attackers task more difficult. It offers a unique advantage in that if vulnerability does occur in a service being protecting with port-knocking, an administrator probably has some grace time to patch it, because the service is not widely available to anyone using a portscanner. Port knocking is an interesting concept and as implementations mature further should result in wider use and broader application within computer security.

Links to some Current Implementations:

Bash version - http://www.opennet.ru/base/sec/port_knocking.txt.html
Cd00r - http://www.phenoelit.de/stuff/cd00rdescr.html
Combo.c - http://www.e-normous.com/nerd/
Doorman - http://doorman.sourceforge.net/
Fwknop - http://www.cipherdyne.org/fwknop/
JPortknock - http://www.gregoire.org/code/jportknock/
knockd - http://www.zeroflux.org/knock/
Krzywinski's implementation - http://www.portknocking.org/view/download/
Pasmal - http://pasmal.casino770.com/
SAdoor - http://cmn.listprojects.darklab.org/

List of References

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

[2] Krzywinski, Martin "Port Knocking Summary" URL: http://www.portknocking.org/view/about/summary (22 Sept 2004)

[3] Slashdot – "PK Discussion on Slashdot" URL: http://slashdot.org/comments.pl?sid=95670&cid=8194532 (22 Sept 2004)

[4] Slashdot - "PK Discussion on Slashdot" URL: http://slashdot.org/comments.pl?sid=95670&cid=8199147 (22 Sept 2004)

[5] Miller, Robin, “Security through Obsolescence”, URL: http://www.theregister.co.uk/2002/06/06/security_through_obsolescence/ (22 Sept 2004)

[6] Beale, Jay – ‘"Security Through Obscurity" Ain't What They Think It Is’ URL: http://www.bastille-linux.org/jay/obscurity-revisited.html (22 Sept 2004)

[7] Ogren, Eric, **“**Using a layered security approach to achieve network integrity**”**,URL: http://www.computerworld.com/printthis/2004/0,4814,89861,00.html (22 Sept 2004)

[8] Slashdot, “PK Discussion on Slashdot” URL: http://slashdot.org/comments.pl?sid=95670&cid=8192658 (22 Sept 2004)

[9] Krzywinski, Martin, ‘Is port knocking an obscurity hack?’ URL: http://www.portknocking.org/view/about/obscurity (22 Sept 2004)

[10] Slashdot, “PK Discussion on Slashdot”, URL: http://slashdot.org/comments.pl?sid=95670&cid=8192452 (22 Sept 2004)

[11] Relatively recent debates are found here:
Slashdot, “PK Discussion on Slashdot” URL: http://slashdot.org/articles/04/02/05/1834228.shtml?tid=126&tid=172 (22 Sept 2004)Slashdot, “PK + OS Fingerprinting discussion on Slashdot” URL: http://it.slashdot.org/it/04/08/01/0436204.shtml (22 Sept 2004)
Debian-Security “idea for improving security” URL: http://lists.debian.org/debian-security/2003/05/msg00059.html (22 Sept 2004) - Discussion on Debian Security of a simplified version in May 03.

[12] Narayanan, Arvind, “A critique of port knocking”, URL: http://software.newsforge.com/software/04/08/02/1954253.shtml (22 Sept 2004)

[13] Schneier, Bruce, “Port Knocking”, URL: http://www.schneier.com/crypto-gram-0403.html#5 (22 Sept 2004)

[14] Krzywinski , Martin, “Port Knocking: Features”, URL: http://www.portknocking.org/view/about/features (22 Sept 2004)

[15] Krzywinski, Martin, “Port Knocking: Knock Length”, URL: http://www.portknocking.org/view/details/knocklength (22 Sept 2004)

[16] Unknown, “Bash Implementation of Port Knocking”, URL: http://www.opennet.ru/base/sec/port_knocking.txt.html (22 Sept 2004)

[17] Trowbridge, Chris, “An Overview of Remote Operating System Finger Printing” URL: http://www.sans.org/rr/papers/42/1231.pdf (22 Sept 2004)

[18] Rash, Michael, “fwknop”, URL: http://www.cipherdyne.org/fwknop/ (22 Sept 2004)

[19] Ward, Bruce, “Doorman”, URL: http://doorman.sourceforge.net/ (22 Sept 2004)

[20] FX, “Cd00r” http://www.phenoelit.de/stuff/cd00rdescr.html (22 Sept 2004)

[21] CMN, “SaDoor”, URL: http://cmn.listprojects.darklab.org/ (22 Sept 2004)

[22] FX, “cd00r”, URL: http://www.phenoelit.de/stuff/cd00r.c (22 Sept 2004)

[23] Nakjang, Nawapong, “A Practical Approach of Stealthy Remote Administration” URL: http://www.linuxsecurity.com/feature_stories/feature_story-149.html (22 Sept 2004)

[24] Whitehouse, Walker and Yamamoto, Mike, “Knock Knock” URL: http://cyberdefensemag.com/april2004/citech.php (22 Sept 2004)

[25] Bradley, Tony, “Port Knocking: Good Guys and Bad Guys Are Using This Method To Open Ports”, URL: http://netsecurity.about.com/cs/generalsecurity/a/aa032004.htm (22 Sept 2004)

[26] “Port knocking New trend for Firewall Administrators”, URL:http://www.tla.ch/TLA/NEWS/2004sec/20040224PortKnocking.htm (22 Sept 2004)

[27] Bill, “INFO: Port Triggering – What is it?”, URL: http://www.dslreports.com/forum/remark,1020195;root=equip,16;mode=flat (22 Sept 2004)
Krzywinski, Martin, “Port Triggering”, URL: http://www.portknocking.org/view/about/port_triggering (22 Sept 2004)

[28] Eastlake, D (et al), “Randomness recommendations for security” URL: ftp://ftp.rfc-editor.org/in-notes/rfc1750.txt (22 Sept 2004)