

tsvwg  
Internet-Draft  
Intended status: Standards Track  
Expires: May 12, 2007

M. Larsen  
Ericsson  
F. Gont  
UTN/FRH  
November 8, 2006

Port Randomization  
draft-larsen-tsvwg-port-randomization-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on May 12, 2007.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

Recently, awareness has been raised about a number of "blind" attacks that can be performed against the Transmission Control Protocol (TCP) and similar protocols. The consequences of these attacks range from throughput-reduction to broken connections or corrupted data. These attacks rely on the attacker's ability to guess or know the four-tuple (Source Address, Destination Address, Source port, Destination Port) that identifies the transport protocol instance to be attacked. This document describes a simple and efficient method for random selection of the client port number, such that the possibility of an attacker guessing the exact value is reduced. While this is not a replacement for cryptographic methods, the described port number randomization algorithms provide improved security/obfuscation with very little effort and without any key management overhead.

Table of Contents

- 1. Introduction . . . . . 3
- 2. Randomizing Ports . . . . . 4
  - 2.1. Ephemeral Port Range . . . . . 4
  - 2.2. Choosing the Port . . . . . 4
  - 2.3. Secret Key . . . . . 7
  - 2.4. Choosing Algorithm . . . . . 7
- 3. Security Considerations . . . . . 9
- 4. Acknowledgements . . . . . 10
- 5. References . . . . . 11
  - 5.1. Normative References . . . . . 11
  - 5.2. Informative References . . . . . 11
- Appendix A. Changes from previous versions of the draft . . . . . 13
  - A.1. Changes from draft-larsen-tsvwg-port-randomisation-00 . . 13
- Authors' Addresses . . . . . 14
- Intellectual Property and Copyright Statements . . . . . 15

## 1. Introduction

Recently, awareness has been raised about a number of "blind" attacks that can be performed against the Transmission Control Protocol (TCP) [RFC0793] and similar protocols. The consequences of these attacks range from throughput-reduction to broken connections or corrupted data [I-D.ietf-tcpm-icmp-attacks] [I-D.ietf-tcpm-tcp-antispoof] [Watson].

All these attacks rely on the attacker's ability to guess or know the four-tuple (Source Address, Source port, Destination Address, Destination Port) that identifies the transport protocol instance to be attacked.

Services are usually located at fixed, 'well-known' ports [IANA] at the host supplying the service (the server). Client applications connecting to any such service will contact the server by specifying the server IP address and service port number. The IP address and port number of the client are normally left unspecified by the client application and thus chosen automatically by the client networking stack. Ports chosen automatically by the networking stack are known as ephemeral ports [Stevens].

While the server IP address and the well-known port and client IP address may be available to the attacker, the ephemeral port of the client is usually unknown and must be guessed.

This document describes a method for random selection of the client ephemeral port, thereby reducing the possibility of an off-path attacker guessing the exact value. This is not a replacement for cryptographic methods such as IPsec [RFC4301] or the TCP MD5 signature option [RFC2385]. However, the proposed algorithm provides improved obfuscation with very little effort and without any key management overhead.

The mechanism described is a local modification that may be incrementally deployed, and does not violate the specifications of any of the transport protocols that may benefit from it [RFC0793] [RFC0768] [RFC2960] [RFC4340].

Since the mechanism is an obfuscation technique, focus has been on a reasonable compromise between level of obfuscation and ease of implementation. Thus the algorithm must be computationally efficient, and not require substantial data structures.

## 2. Randomizing Ports

### 2.1. Ephemeral Port Range

The Internet Assigned Numbers Authority (IANA) assigns the unique parameters and values used in protocols developed by the Internet Engineering Task Force (IETF), including well-known ports [IANA]. IANA has traditionally reserved the following use of the 16-bit port range of TCP and UDP:

- o The Well Known Ports, 0 through 1023.
- o The Registered Ports, 1024 through 49151
- o The Dynamic and/or Private Ports, 49152 through 65535

The range for assigned ports managed by the IANA is 0-1023, with the remainder being registered by IANA but not assigned.

The ephemeral port range traditionally includes the 49152-65535 range, and should also include the 1024-49151 range. Since this range includes user-specific server ports this may not always be possible. However, transport protocols SHOULD use the largest possible range, since this improves the obfuscation provided by randomizing the ephemeral ports.

### 2.2. Choosing the Port

Transport protocols SHOULD randomize the ephemeral ports they use.

Choosing a random port can, if a suitable random source is available, be implemented as a simple random selection, i.e.:

```
port = min_ephemeral + random() % (max_ephemeral - min_ephemeral)
```

Figure 1

Several well-know operating systems use this approach.

However, since the resulting connection four-tuple must be unique, the chosen port may already be in use with identical IP addresses and server port, and thus the resulting four-tuple might not be unique. Consequently multiple ports may have to be tried and verified against all existing connections before a port can be chosen.

Although carefully chosen random sources and optimized four-tuple lookup mechanisms (e.g., optimized through hashing), will mitigate

the cost of this verification, some systems may still not want to incur this unknown search time.

Systems that may be specially susceptible to this kind of repeated four-tuple collisions are those that create many connections from a single local IP address to a single service (i.e. both IP addresses and server port are fixed). Gateways such as proxy servers are an example of such a system.

Finding ports that result in a unique four-tuple are handled by some operating systems by having a global 'next ephemeral port' variable that is equal to the previously chosen ephemeral port + 1, i.e. the selection process is:

```
next_ephemeral_port = 1024; /*initialization, could be random */  
  
do {  
    port = next_ephemeral_port;  
    if (next_ephemeral_port == max_ephemeral_port) {  
        next_ephemeral_port = min_ephemeral_port;  
    } else {  
        next_ephemeral_port++;  
    }  
} until (four-tuple is unique);
```

Figure 2

We will refer to this as 'Algorithm 1'. Note that the loop prevention mechanism has been left out for clarity.

This works well, since the number of connections (globally, across all four-tuples) that has a life-time longer than it takes to exhaust the total ephemeral port range is small, thus four-tuple collisions are rare.

However, this method has the drawback, that the 'next\_ephemeral\_port' variable and thus the ephemeral port range is shared between all connections and it is easy to predict the next ports chosen by the client. If an attacker operates an "innocent" server to which the client connects, it is easy to obtain a reference point for the current value of 'next\_ephemeral\_port'.

Ideally, we would like a 'next\_ephemeral\_port' value for each set of (local IP address, remote IP addresses, remote port). These should be initialized with random values within the ephemeral port range and would thus separate the ephemeral port ranges of the connections entirely. Since we do not want to store all these

'next\_ephemeral\_port' values, we propose an offset function  $F()$ , that can be computed from the local IP address, remote IP address, remote port and a secret key.  $F()$  will yield (practically) different values for each set of arguments, i.e.:

```
next_ephemeral_port = 1024; /*initialization, could be random */

offset = F(local_IP, remote_IP, remote_port, secret_key);
do {
    port = min_ephemeral +
           (next_ephemeral_port + offset)
           % (max_ephemeral - min_ephemeral);
    next_ephemeral_port++;
} until (four-tuple is unique);
```

Figure 3

We will refer to this as 'Algorithm 2'. Note that the loop prevention mechanism has been left out for clarity.

In other words, the function  $F()$  provides a per-connection fixed offset of the global ephemeral port range controlled by 'next\_ephemeral\_port'. Both the 'offset' and 'next\_ephemeral\_port' variables may take any value within the storage type range since we are restricting the resulting port similar to that shown in Figure 1. This allows us to simply increment the 'next\_ephemeral\_port' variable and rely on the unsigned integer to simply wrap-around.

The function  $F()$  should be a cryptographic hash function like MD5 [RFC1321]. The function should use both IP addresses, the remote port and a secret key value to compute the offset. The remote IP address is the primary separator and must be included in the offset calculation. The local IP address and remote port may in some cases be constant and not improve the connection separation, however, they should also be included in the offset calculation.

Cryptographic algorithms stronger than e.g. MD5 should not be necessary, given that port randomization is simply an obfuscation technique. The secret should be chosen as random as possible, see [RFC1750] for recommendations on choosing secrets.

Note that on multiuser systems, the function  $F()$  could include user specific information, thereby providing protection not only on a host to host basis, but on a user to service basis.

### 2.3. Secret Key

Every complex manipulation (like MD5) is no more secure than the input values, and in the case of ephemeral ports, the secret key. If an attacker is aware of which cryptographic hash function is being used by the victim (which we should expect), and the attacker can obtain enough material (e.g. ephemeral ports chosen by the victim), the attacker may simply search the entire secret key space to find matches.

To protect against this, the secret key should be of a reasonable length. Key-lengths of 32-bits should be adequate, since a 32-bit secret would result in approximately 65k possible secrets if the attacker is able to obtain a single ephemeral port (assuming a good hash function). If the attacker is able to obtain more ephemeral ports, key-lengths of 64-bits or more should be used.

Another possible mechanism for protecting the secret key is to change it after some time. If the host platform is capable of producing reasonable good random data, the secret key can be changed.

Changing the secret will cause abrupt shifts in the chosen ephemeral ports, and consequently collisions may occur. Thus the change in secret key should be done with consideration and could be performed whenever one of the following events occur:

- o Some predefined/random time has expired.
- o The secret has been used N times (i.e. we consider it insecure).
- o There are few active connections (i.e., possibility of collision is low).
- o There is little traffic (the performance overhead of collisions is tolerated).
- o There is enough random data available to change the secret key (pseudo-random changes should not be done).

### 2.4. Choosing Algorithm

Algorithm 1 has the advantage that it provides complete randomization, but may not scale well with many simultaneous connections. Algorithm 2 provides complete separation in local and remote IP addresses and remote port space, and only limited separation in other dimensions (See Section Section 2.3), however, this algorithm scales well.

Thus Algorithm 1 should be used when the cost of choosing an ephemeral port is not important, or when the ratio of used ports to available ports is low (for a given local IP address, remote IP address, and remote port). A switch to algorithm 2 should happen if the cost of choosing an ephemeral port is important and when the ratio between used ports and available ports increase.

Note that when the ratio between used ports and available ports increase, the obfuscation resulting from port randomization decreases and has no effect when the entire port space is in use.

The ratio at which to switch between algorithms depends on the cost of the four-tuple uniqueness test. Systems capable of handling many simultaneous connections normally have an efficient PCB-lookup. However, verifying a four-tuple for uniqueness requires a lookup against all existing connections, even unconnected (but bound). Additionally, for some protocols (e.g., TCP) options exist that allow reuse of port numbers, making the detection even more complex than a PCB-lookup. The cost of a four-tuple verification may easily be many times that of a single PCB lookup.

While the ratio is very implementation-dependent and calculating the exact ratio may be difficult without using additional resources, an appropriate ratio can be estimated and used for an algorithm switch. E.g. if the ephemeral port range contains  $N$  possible ports, the switch to algorithm 2 may happen when the total number of connections reaches  $N/2$ .

### 3. Security Considerations

Randomizing ports is no replacement for cryptographic mechanisms, such as IPsec [RFC4301].

An eavesdropper, which can monitor the packets that correspond to the connection to be attacked could learn the IP addresses and port numbers in use (and also sequence numbers etc.) and easily attack the connection. Randomizing ports does not provide any additional protection against this kind of attacks. In such situations, proper authentication mechanisms such as those described in [RFC4301] should be used.

If the local offset function  $F()$  results in identical offsets for different inputs, the port-offset mechanism proposed in this document has no or reduced effect.

If random numbers are used as the only source of the secret key, they must be chosen in accordance with the recommendations given in [RFC1750].

If all ports available in the ephemeral port range are in use, randomization provides no obfuscation.

If an attacker uses dynamically assigned IP addresses, the current ephemeral port offset (Algorithm 2) for a given four-tuple can be sampled and subsequently be used to attack an innocent peer reusing this address. However, this is only possible until a re-keying happens as described above. Also, since ephemeral ports are only used on the client side (e.g. the one initiating the connection), both the attacker and the new peer need to act as servers in the scenario just described. While servers using dynamic IP addresses exist, they are not very common and with an appropriate re-keying mechanism the effect of this attack is limited.

#### 4. Acknowledgements

The offset function was inspired by the mechanism proposed by Steven Bellovin in [RFC1948] for defending against TCP sequence number attacks.

## 5. References

### 5.1. Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [RFC1750] Eastlake, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.
- [RFC1948] Bellare, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.
- [RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

### 5.2. Informative References

- [Watson] Watson, P., "Slipping in the Window: TCP Reset attacks", december 2003.
- [IANA] "IANA Port Numbers",  
<<http://www.iana.org/assignments/port-numbers>>.
- [Stevens] Stevens, W., "Unix Network Programming, Volume 1: Networking APIs: Socket and XTI, Prentice Hall", 1998.
- [I-D.ietf-tcpm-tcp-antispoof] Touch, J., "Defending TCP Against Spoofing Attacks", draft-ietf-tcpm-tcp-antispoof-05 (work in progress), October 2006.

[I-D.ietf-tcpm-icmp-attacks]

Gont, F., "ICMP attacks against TCP",  
draft-ietf-tcpm-icmp-attacks-01 (work in progress),  
October 2006.

Appendix A. Changes from previous versions of the draft

A.1. Changes from draft-larsen-tsvwg-port-randomisation-00

- o Document resubmitted after original document by M. Larsen expired in 2004
- o References were included to current WG documents of the TCPM WG
- o The document was made more general, to apply to all transport protocols
- o Miscellaneous editorial changes

Authors' Addresses

Michael Vittrup Larsen  
Ericsson  
Skanderborgvej 232  
Aarhus DK-8260  
Denmark

Phone: +45 8938 5100  
Email: michael.vittrup.larsen@ericsson.com

Fernando Gont  
Universidad Tecnologica Nacional / Facultad Regional Haedo  
Evaristo Carriego 2644  
Haedo, Provincia de Buenos Aires 1706  
Argentina

Phone: +54 11 4650 8472  
Email: fernando@gont.com.ar

## Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

