Results of a security assessment of the TCP and IP protocols and common implementation strategies

Fernando Gont

project carried out on behalf of

UK CPNI

DEEPSEC 2009 Conference
November 17-20, 2009. Vienna, Austria

About the speaker

- For the last few years I have worked on security assessment of communication protocols for the UK CPNI (Centre for the Protection of National Infrastructure).
- I'm also active at the IETF (Internet Engineering Task Force), where I have authored a few RFCs and several Internet-drafts (I-Ds).
- Whenever possible, I contribute code to open source operating systems (e.g., OpenBSD and FreeBSD).
- More information available at: http://www.gont.com.ar

Agenda

- Project overview
- Discussion of some specific issues
  - The “new” TCP Dos attacks (Sockstress)
  - Remote OS detection via TCP/IP stack fingerprinting
- Further work & Conclusions
- Questions and (hopefully) answers

Overview

(or “what we did, and why we did what we did”)

Problem Statement

- Many vulnerabilities have been found in a number of implementations of the TCP & IP protocols, and in the protocols themselves.
- Documentation of these issues has been spread among too many documents.
- Some of the proposed counter-measures negatively impact protocol interoperability (see e.g., Silbersack's presentation at BSDCan 2006).
- The efforts of the security community never resulted in changes in the corresponding IETF specifications, and sometimes not even in the protocol implementations.
- As a result, the same vulnerabilities have been re-hashed in new implementations of the protocols and even in "brand-new" protocols (e.g., IPv6 RHT0).

Project Overview

- During 2006-2008, CPNI – formerly NISCC – embarked itself in a project to fill this gap.
- The goal: producing a security roadmap for the TCP and IP protocols. The resulting documents are::
  - http://www.cpni.gov.uk/Docs/InternetProtocol.pdf
  - http://www.cpni.gov.uk/Docs/tn-03-09-security-assessment-TCP.pdf
- This set of documents would be updated in response to the feedback received from the comunity.
- Results would finally be taken to the IETF -- both documents have already been adopted by the IETF:
  - draft-ietf-opsec-ip-security (in opsec wg)
  - draft-ietf-tcpm-tcp-security (in tcpm wg)

Why work on TCP and IPv4 security?

- TCP and IP are the two most widely used protocols in the Internet – and will continue to be so for many years.
- Applications ranging from the web to BGP rely on them.
- While many security issues have been discussed in the past, many of them have never been addressed in actual implementations.
- The recent suggestion of relying on TCP for the DNS (as a result of DNSsec and IPv6) has put the the resiliency of TCP implementations back into question.
- Probably not a glamorous topic to work on…. but at the end of the day TCP and IP are two building blocks that we rely on.

Some areas of work

- Propose sanity checks to perform on header fields an options
- Identify TCP and IP options that currently have no legitimate purpose
- Propose algorithms for selecting header fields or options (e.g., IP ID, TCP ephemeral ports, TCP timestamps)
- Mitigate the exploitations of protocol mechanisms (e.g., Path-MTU Discovery, etc.)
- Improve standard protocol policies with known security implications (e.g., IP fragment reassembly, TCP congestion contorl, etc.)

The new (?) TCP DoS attacks

("the sky is falling.... but we cannot tell you why")

The Internet Plagued by Another Critical Design Flaw

A TCP stack design vulnerability could put Internet services everywhere at major DoS risk

Source: news.softpedia.com

The “new” TCP DoS attacks

- During 2008, the discovery of some (supposedly) new vulnerabilities received their share of press.
- They were “announced” by Outpost24, but no details were provided – thus resulting in speculation by the community.
- No counter-measures were proposed to vendors, either.
- While not publicly credited for our work, we provided advice to vendors on these issues.
- For the most part, the vulnerabilities are:
  - Connection-flooding attacks (Naphta and FIN-WAIT-2 flooding attacks)
  - Socket send buffer attacks (Netkill and closed windows)
  - TCP reassembly buffer attacks

Some insights on the recent TCP DoS vulnerabilities

(our view of these issues)

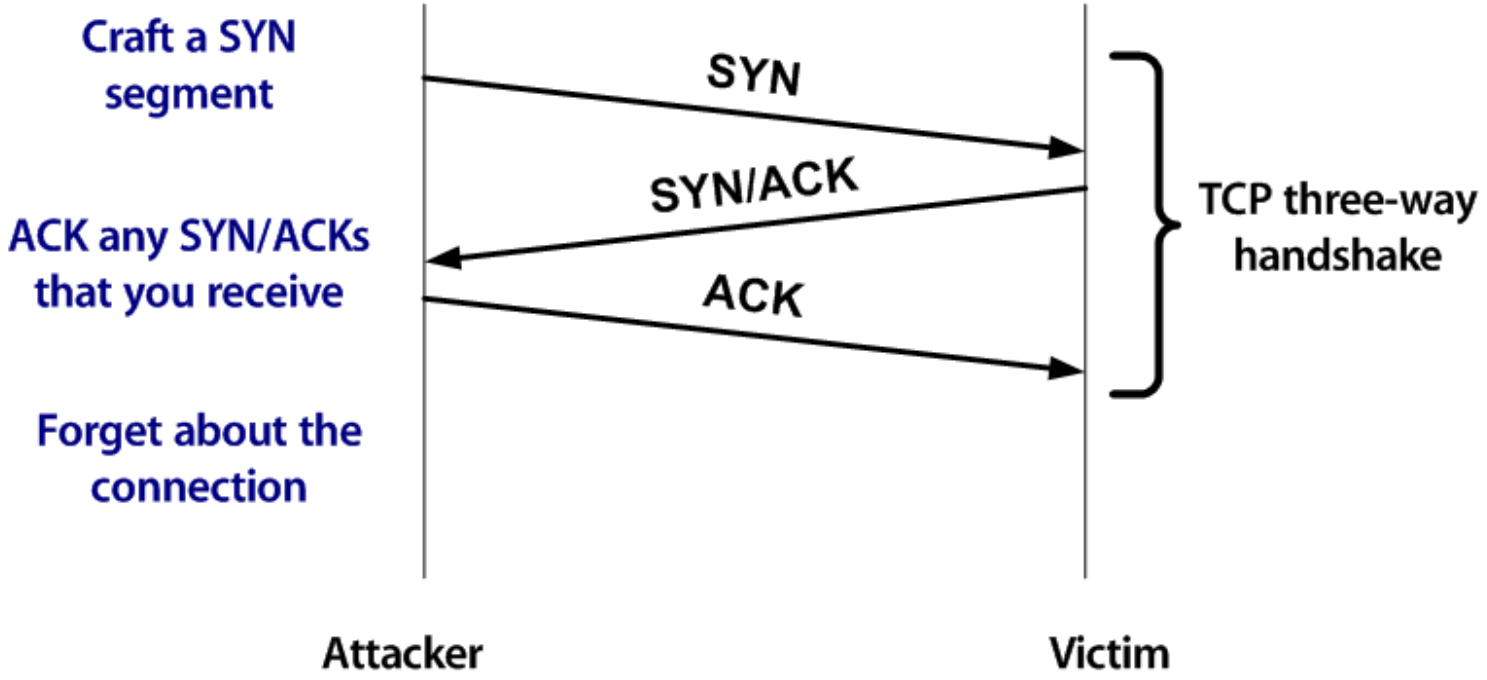
Connection-flooding attacks

(Naphta and FIN-WAIT-2)

Naphta (connection-flooding attack)

- TCP connections require end-points to keep state (in system memory) for the connections.
- Memory is a limited resource, and thus can be targeted for exhaustion: simply establish lots of connections with the target system.
- This attack vector was known as “Naphta” -- see CERT Advisory CA-2000-21.
- To avoid exhausting his own resources simply crafts the required packets to establish TCP connections with the target system, thus bypassing its kernel implementation of TCP.

Naptha attack (example)

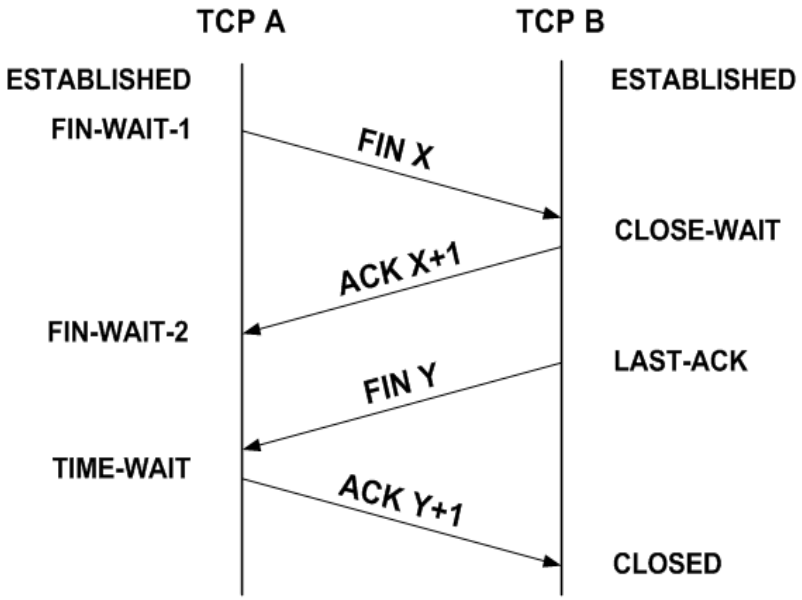


Countermeasures for Naphta

- Key problem: an actual attack does not necessarily differ from a high-load scenario
- Possible counter-measures:
  - Enforce per-user and pre-process limits
  - Enforce limits on the number of ongoing connections from a single system/prefix at the application-layer
  - Enforce limits on the number of ongoing connections from a single system/prefix at a firewall

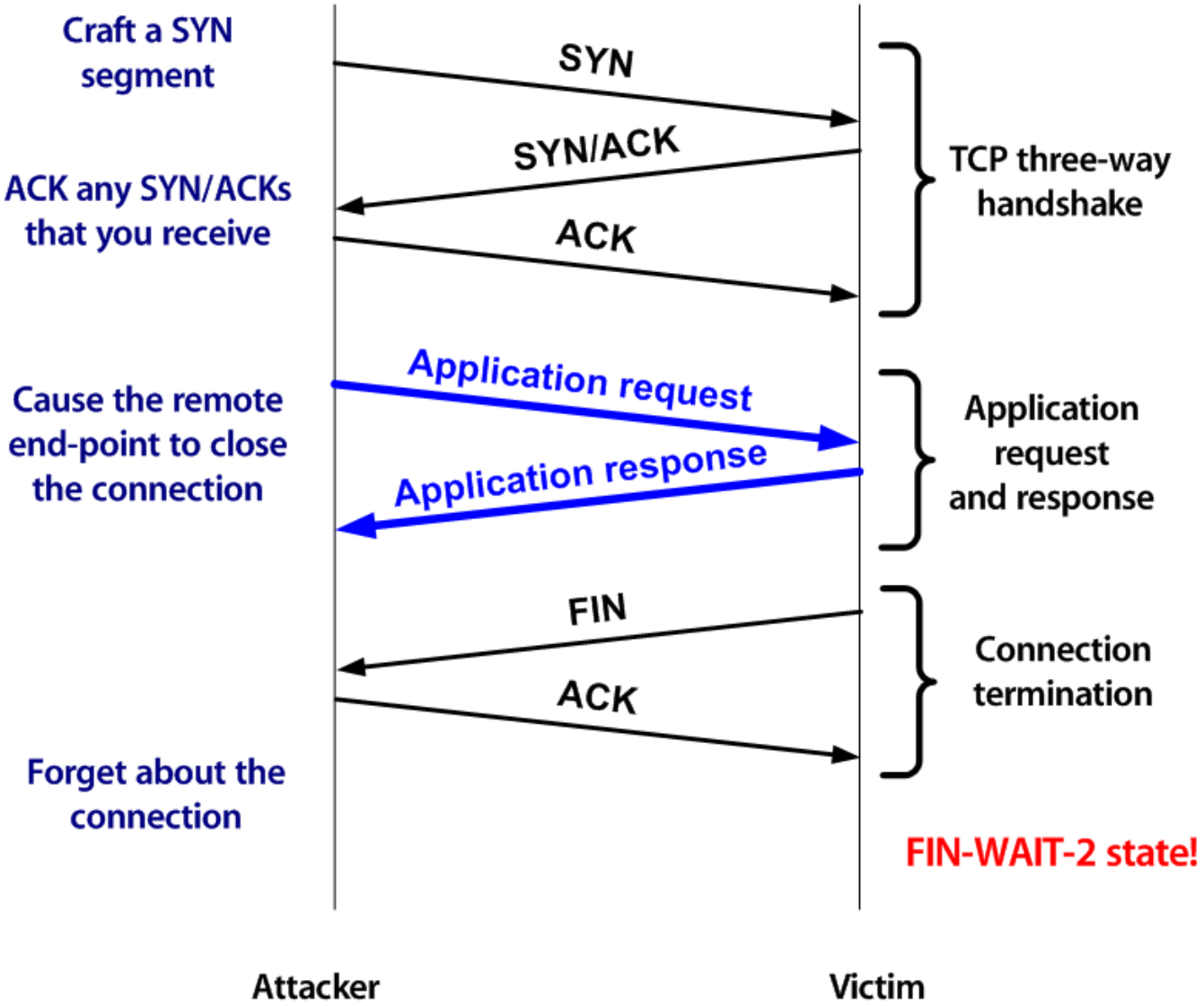
FIN-WAIT-2 flooding attack

- A typical connection-termination scenario:



- Problems that may arise due to the FIN-WAIT-2 state
  - There's no limit on the amount of time a connection can stay in the FIN-WAIT-2 state – connections could stay forever in FIN-WAIT-2.
  - When TCP gets into the FIN-WAIT-2 state there's no user-space controlling process (i.e., it's hard to enforce application-layer limits)

FIN-WAIT-2 attack (example)



Countermeasures for FIN-WAIT-2 flooding

- Enforce a limit on the duration of the FIN-WAIT-2 state. E.g., Linux 2.4 enforces a limit of 60 seconds. Once that limit is reached, the connection is aborted.
- Enforce on the number of ongoing connections with no controlling process.
- The counter-measures for the Naptha attack still apply. However, it is difficult for applications to enforce limits (remember: no controlling process for the connections).
- Applications should be modified so that they retain control of the connection for most states. This can be achieved with a conbination of the shutdown(), setsockopt(), and close().

Socket send buffer vulnerabilities

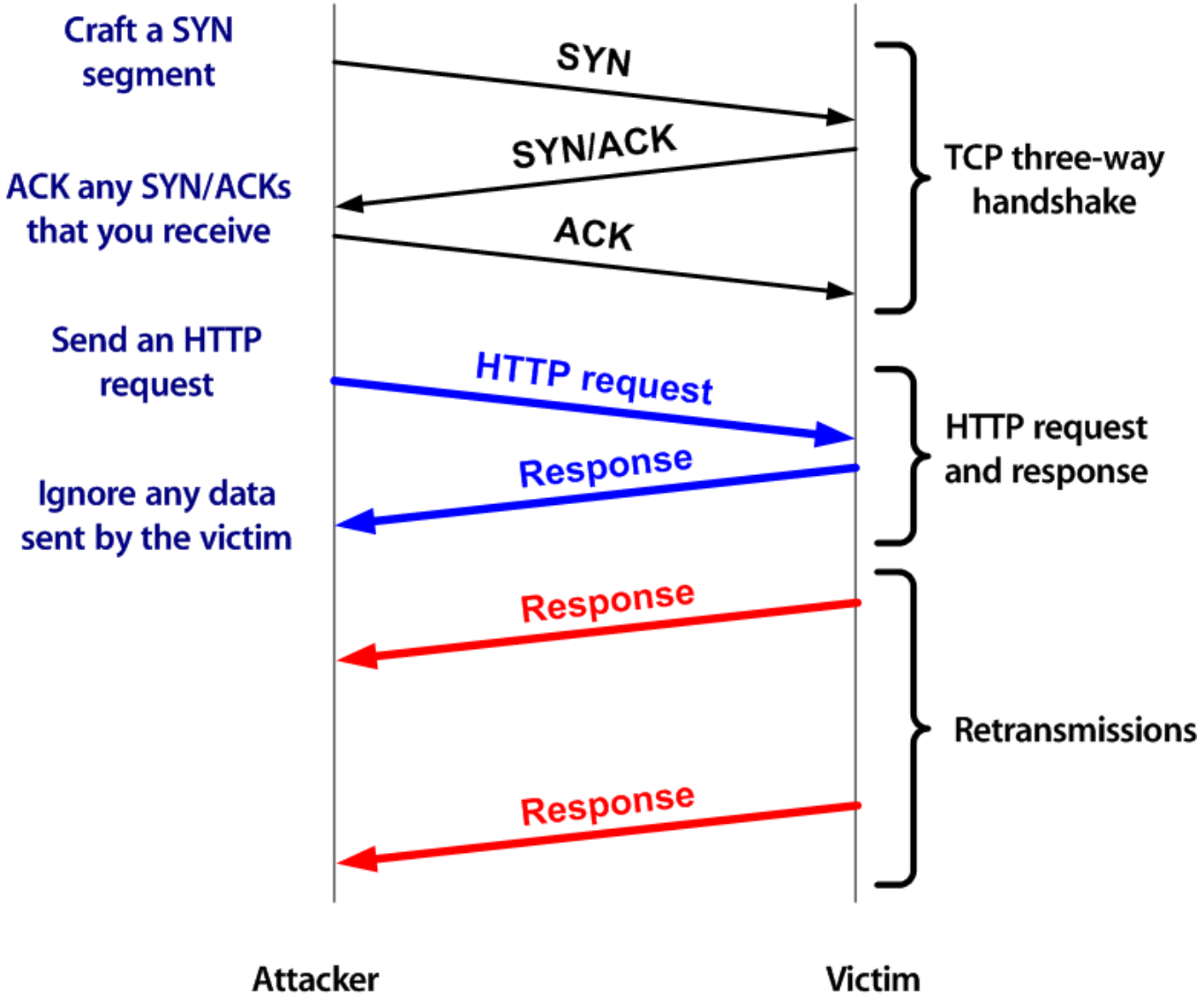
Socket send buffer

- The socket send buffer keeps a copy of those data that have been accepted by TCP for delivery to the remote TCP end-point.
- It is possible to exploit the Socket send buffer for a memory exhaustion attack:
  - Send an application request to the target system, but never acknowledge the response (Netkill).
  - Send an application request, but immediately close the receive window, so that the target TCP refrains from actually sending the response.

Netkill

- Data that have been sent but not yet acknowledged are kept in the socket send buffer for their possible retransmission.
- TCP will retransmit those data until they either get acknowledged or the connection times out. In the mean time, system memory is tied to those data.
- Easy to exploit for memory exhaustion: establish lots of TCP connections, send an applicattion-request on each of them, and never acknowledge the received data.

Netkill (example)



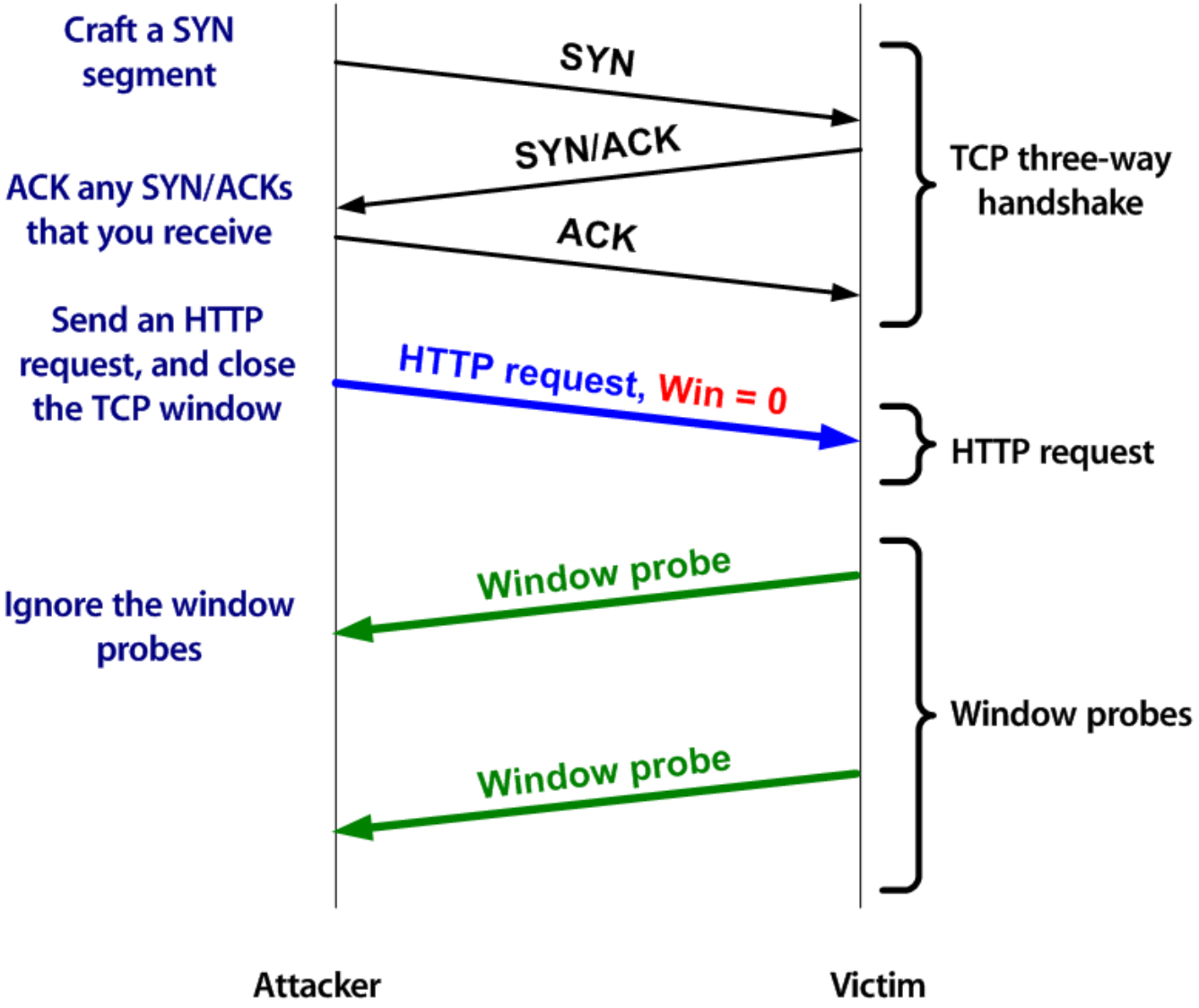
Netkill (countermeasures)

- Problem: it's very hard to infer attack from the behavior of a single connection.
- Possible counter-measures:
  - Measure connection progress at the application-layer
  - Do not use an unnecessarily large socket send buffer
  - Enforce per-user and pre-process limits
  - Enforce limits on the number of ongoing connections from a single system/prefix at the application-layer
  - Enforce limits on the number of ongoing connections from a single system/prefix at a firewall
- When dropping connection, these are possible parameters that may provide hints for selecting the target connection:
  - Large amount of data queued in the TCP retransmission buffer
  - Small amount of data successfully transferred to the remote endpoint

Closed windows

- The TCP sliding-window mechanism prevents a fast sender from overwhelming a slow consumer application.
- When the advertised window is zero, the window is said to be closed.
- The TCP sender polls the receiver from time to time to find out if the window has opened (persist timer). However, there's no limit on the amount of time that the window can be closed.
- Easy to exploit for memory exhaustion: just send an applicattion-request to the remote end-point, and close the receive window.

Closed windows (example)



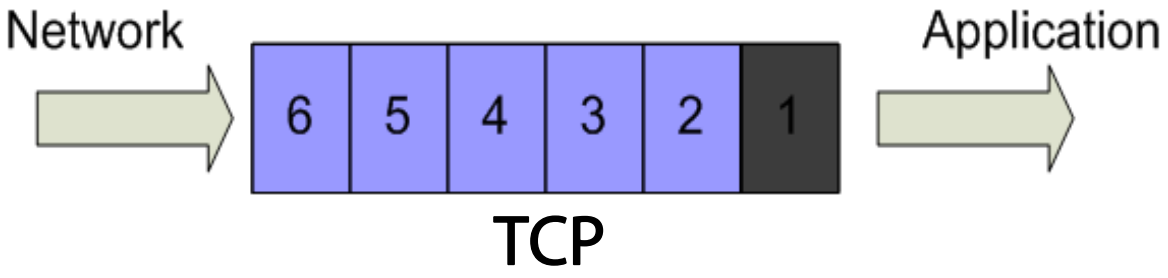
Closed windows (countermeasures)

- Problem: it's very hard to infer attack from the behavior of a single connection.
- It has been proposed that TCP should impose a limit on the amount of time that the window can be closed. However, this counter-measure is trivial to circumvent: just open the window a bit from time to time.
- Possible counter-measures:
  - Measure connection progress at the application-layer
  - Do not use an unnecessarily large socket send buffer
  - Enforce per-user and pre-process limits
  - Enforce limits on the number of ongoing connections from a single system/prefix at the application-layer
  - Enforce limits on the number of ongoing connections from a single system/prefix at a firewall

TCP reassembly buffer attacks

TCP reassembly buffer

- When out-of-order data are received, a “hole” momentarily exists in the data stream which must be filled before the received data can be delivered to the application making use of TCP’s services.



- This mechanism can be exploited in at least two ways:
  - Create a hole in the data stream, and send a large amount of data.
  - Send e.g., chunks of one byte of data, separated by holes of e.g., one byte, targeting the overhead needed to hold and link each of these chunks of data.

Countermeasures for the reassembly buffer

- TCP implementations should enforce limits on the amount of out-of-order data that are queued at any time.
- TCP implementations should enforce limits on the maximum number of “holes” that are allowed for each connection.
- Per-user and per-process limits should be enforced.
- If necessary, out-of-order data could be discarded, with no effect on interoperability (this has a performance penalty, though).

Remote OS detection

(via TCP/IP stack fingerprinting)

Remote OS detection

- A number of tools, such as nmap, can detect the operating system in use at a remote system, via TCP/IP stack fingerprinting.
- They send a number of probe packets that different stacks process in different ways
- The precision of their results is amazingly good. – It shouldn't be that good!
- Question: Wouldn't it be possible for these TCP/IP stacks to respond to most of these probes in exactly the same way?

Some fingerprinting probes

- We have performed an analysis of each of the available TCP/IP stack fingerprinting probes, and provided advice on how to respond to each of these probe packets:
  - FIN probe
  - Bogus flag test
  - RST sampling
  - Port-0 probe
- We expect that in the long term remote OS detection based on these probes will have much less precision.

TCP option ordering

- Another important technique for remote OS detection is to fingerprint the TCP options used by the target system.
- Different TCP implementations enable different options (by default) in their TCP connections, set their values differently, and frame the options differently.
- More work is needed to get consensus on which options should be included by default, how to frame them, and what their default value shoul be.
- An additional benefit from such consensus would enable "TCP option prediction" (i.e., tune the code so that processing of packets with the usual options in the usual order is faster).

Further work & Conclusions

Further Work

- We plan to publish a revision of the UK CPNI documents whnever we find the cycles. Reviews are welcome!
- We are pursuing this effort in the IETF to update the specifications where necessary. However, there's some resistance to update/fix the specs (talk about politics). – **Get involved!**
  - Join the tcpm wg at: https://www.ietf.org/mailman/listinfo/tcpm
  - Join the opsec wg at: https://www.ietf.org/mailman/listinfo/opsec
- **Your feedback really makes a difference.**

Conclusions

- Working on TCP/IPv4 security in 2006-2008 probably didn't have much glamour. However, it was needed.
- Still in 2009, there's lots of work to do to improve the available TCP implementations.
- The same questions keep being asked, in different contexts (DNS, NATs, etc.)
- Fortunately, we have been converging on the "right" answer.

Questions?

Acknowledgements

- UK CPNI, for their continued support.
- DEEPSEC organizers, for their support in this conference.

Thank you!

Fernando Gont

fernando@gont.com.ar

http://www.gont.com.ar