

Some insights about the recent TCP DoS (Denial of Service) vulnerabilities

Fernando Gont

project carried out on behalf of
UK CPNI

HACK.LU 09 Conference
October 28-30, 2009. Luxembourg



Agenda

- Overview of the project carried out on behalf of UK CPNI
- Disclosure process of our results
- The recent TCP DoS (Denial of Service) vulnerabilities
- Disclosure process of the “Sockstress” vulnerabilities
- Disclosure process
- Co-operation with vendors
- Conclusions



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 and of possible mitigations has been spread among a number of vulnerability reports and a variety of online documents.
- Some of this documentation proposes counter-measures for these issues without analyzing their interoperability implications on the protocols. (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,
 - New implementations of the protocols re-implement vulnerabilities found in older implementations.
 - New protocols re-implement mechanisms or policies with "known" security implications (e.g., Router Header Type 0 in IPv6 vs. IPv4 source routing).

Project Overview

- During the last few years, CPNI – formerly NISCC – embarked itself in a project to fill this gap.
- The goal was to produce a set of documents that would serve as 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 community.
- Finally, we planned to take the results of this project to the IETF, so that the relevant specifications could be modified where needed. Both documents have already been adopted by the IETF:
 - draft-ietf-opsec-ip-security
 - draft-ietf-tcpm-tcp-security



Disclosure process of our results

(“who got what, and when?”)

Results of our project

- The IP security document was released in August 2008, after review from vendors.
- The results of our TCP security project was shared with main vendors and operators during 2006-2008 (and publicly released in February 2009).
- We didn't hear much from vendors.
- For the most part, we got feedback from colleagues (via "would you take a look at this?"), and some gubernamental organizations.
- By mid 2008 vendors didn't seem to be concerned about the contents of our document.

But later in 2008...

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



Sockstress Attacks

- To be seen and experienced live at the show...*
- We are still working with vendors, so we must limit the details of what Sockstress is Attacking
 - We will share more background information at the talk
 - We will also demonstrate the attacks live

The new TCP DoS attacks

- Some (supposedly) new and killer attacks had been discovered by Outpost24.
- The attacks received a lot of press.
- And there were many claims about their impact.
- The information provided by Outpost24 was really scarce, and some of it simply didn't make sense
- What followed: More press, some panic, speculation by the community.
- At some point, some vendors and CSIRTs that were aware about our efforts on TCP & IP security came back to us ("What is this all about?")
- So we concentrated on the aforementioned issues, and developed and provided specific advice to vendors. – it was the only advice that they had.

Summary of the vulnerabilities

- For the most part, the vulnerabilities are:
 - Connection-flooding attacks (naphta and FIN-WAIT-2 flooding attacks)
 - TCP send buffer attacks (Netkill and closed windows)
 - TCP receive buffer attacks
- No countermeasures were proposed as part of the Outpost 24 report to vendors and CSIRTs..
- Outpost24 did find some pathological behavior of some stacks, e.g., when transiting through the FIN-WAIT-2 state, though.



Some insights on the recent TCP DoS vulnerabilities

(our view of these issues)



Connection-flooding attacks

(Naphtha and FIN-WAIT-2)

Connection-flooding attacks (Naphta)

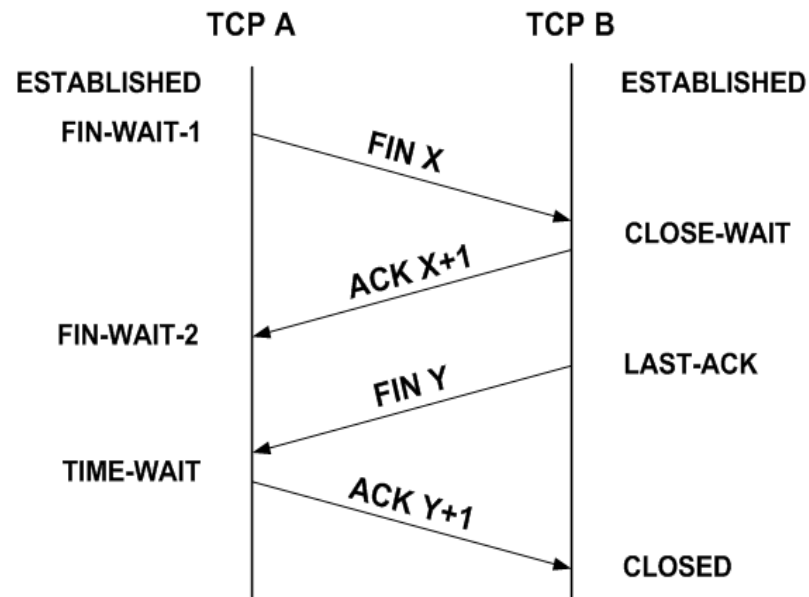
- The creation and maintenance of a TCP connection requires system memory to maintain shared state between the local and the remote TCPs.
- Given that system memory is a limited resource, this can be exploited to perform a DoS attack (this attack vector has been referred to as “Naphta”. See CERT Advisory CA-2000-21).
- In order to avoid wasting his own resources, an attacker can bypass the kernel implementation of TCP, and simply craft the required packets to establish a TCP connection with the remote endpoint, without tying his own resources.
- Outpost24 stated that in order to exploit the attack, they had to introduce the concept of “client-side cookies”.
 - This is not needed: Simply fire the SYNs, and respond with an ACK all the SYN/ACKS that you receive (KISS principle).

Mitigating 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 potentially 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
 - At the point a TCP gets into the FIN-WAIT-2 state there's no user-space controlling process

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.
- The counter-measures for the Naptha attack still apply. However, the fact that this attack aims at leaving lots of connections in the FIN-WAIT-2 state will usually prevent an application from enforcing limits on the number of ongoing connections.
- Applications should be modified so that they retain control of the connection for most states. This can be achieved with a combination of the `shutdown()`, `setsockopt()`, and `close()`.
- TCP should also enforce limits on the number of ongoing connections with no controlling process.



TCP send buffer

TCP send buffer

- The TCP 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 TCP 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 close the receive window.

Netkill

- The amount of data that are allowed for delivery to the remote TCP end-point is governed by the TCP sliding-window mechanism.
- The TCP window 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 application-request to the remote end-point, and close the receive window.

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
 - Only a small amount of data successfully transferred to the remote endpoint

Closed windows

- The amount of data that are allowed for delivery to the remote TCP end-point is governed by the TCP sliding-window mechanism.
- The TCP window 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 application-request to the remote end-point, and close the receive window.

Closed windows (countermeasures)

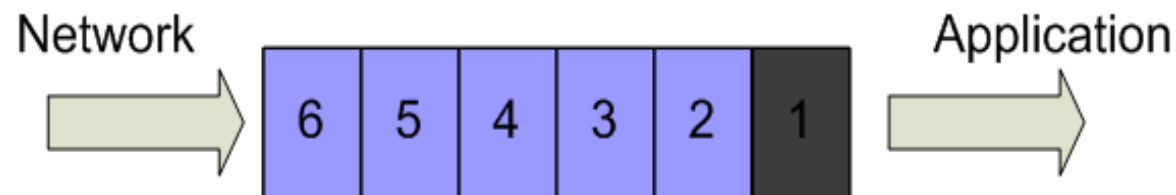
- Problem: it's very hard to infer attack from the behavior of a single connection.
- It has been proposed that TCP should impose a limit on the amount of time that a window can be closed.
- However, this counter-measure is trivial to circumvent: just open the window a bit from time to time.
- Possible 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

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 lots of connections, and send a large amount of data on each of those connections to the receiving TCP, leaving a hole in the data stream so that those data cannot be delivered to the application.
 - Same as above, but 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 receive 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.
- If necessary, out-of-order data could be discarded, with no effect on interoperability. This has a performance penalty, though.



Cooperation with vendors

(too frequently, an oxymoron)

Oxymoron (NOUN): A rhetorical figure in which incongruous or contradictory terms are combined.

What one would expect from vendors

- You provide advice to them on possible security problems in their product.
- You delay publication of your work so that they have enough time to fix their products before going public.
- They fix their products before the issues become public
 - Good for the vendors
 - Good for their customers
- They credit your work (possibly in their vulnerability advisories).

Unfortunately, the process usually fails in this last step...

Case 1: Microsoft

- Microsoft had received draft versions of our document during 2007-2008.
- I personally delivered the document to key people at Microsoft.
- However, when MS09-048 was released by Microsoft, there was no credit to our work.
- Microsoft's response to my query was:

"As it was not reported directly here, we were never tracking you paper as a part of this case. We also did not credit any other ICASI members or CERT-FI who were also providing guidance for mitigation and workarounds to enable users to protect themselves.", and...

"As your documents discuss the TCP/IP protocol and did not discuss any specific issues with Microsoft products, there is nothing actionable for us to triage or investigate."

Case 2: Cisco Systems

- Cisco had received draft versions of our document during 2007-2008.
- I personally delivered the document to key people at Microsoft.
- However, when cisco-sa-20090908-tcp24 was released by Microsoft, there was no credit to our work.
- Cisco's response to my query was:

"Thank-you for contacting Cisco about your concerns in the recent TCP Security Advisory, cisco-sa-20090908-tcp24. Yes, we reviewed your research paper in the August 2008 timeframe and at that time had no concerns about the publication. Our advisory was in direct response to what was being coordinated by CERT-FI across industry and the existence of the proof-of-concept tool delivered to us by Outpost24"



Conclusions



Conclusions and Further Work

- Working on TCP/IPv4 security in 2005-2008 probably didn't have much glamour. However, this was something that needed to be done.
- Still in 2009, there's lots of work to do to improve the available TCP implementations.
- We're aware of some efforts in the vendor community to improve the security/resiliency of TCP. Not sure what the end result will be.
- There are efforts in the IETF to update the specifications where necessary. However, there's also some resistance by some participants to update/fix the specs (talk about politics). – **Get involved!**
- **Your feedback really matters.**



Questions?

Acknowledgements

- UK CPNI, for their continued support
- HACK.LU organizers, and Fred in particular, for their support in this conference.

Fernando Gont

fernando@gont.com.ar

<http://www.gont.com.ar>



Thank you!