

# IDS Insertion and Evasion Techniques

Herbert Haas

January 2009

# Table of contents

- 1 Introduction
- 2 Ambiguities
- 3 IP Fragments
- 4 TCP Attacks

# Outline

- 1 Introduction
- 2 Ambiguities
- 3 IP Fragments
- 4 TCP Attacks

## The IDS as primary target

- If the attacker knows about the existence of an IDS, the IDS will be the most interesting target.
- Because when the IDS can be deactivated, no complicated evasion technique is necessary.
- Availability of IPS often means reachability of internal network — unless a *fail-open* feature has been configured.

## The IDS as primary target (cont.)

At least the attacker is interested to fool the IDS:  
Increase the number of false positives  
Make the IDS to provide wrong or confusing information (*false negatives*)

## Insertion attacks

**Idea:** The IDS does not know that the end-system would reject that packet. Then the attacker can for example insert blinders within a malicious string (sent byte by byte).

- The IDS would accept all bytes and recognizes the string as harmless
- The host only accepts those bytes that belong to the malicious sequence

When UDP is used, the blinders could be datagrams with wrong checksum (which are only dropped by the destination host). Widespread vulnerability! Therefore also the IDS must check the checksum of each packet, etc. . .

Insertion: The IDS gets *more* packets than the destination.

## Evasion attacks

Previously, with **insertion attacks** the IDS is less strict in packet processing than the host. Alternatively, if the IDS is more strict, then this could lead to **evasion attacks**:

For example, if the malicious sequence is sent byte-by-byte, and one byte is rejected by the IDS, the IDS cannot detect the attack. Generally, the IDS rejects packets, the victim does not.

Evasion: The IDS gets *less* packets than the destination.

## Other types of Evasion

- **Encryption** – If the attacker already established an encrypted session to the victim, this would be the most effective evasion attack.
- **Flooding** – The attacker sends lots of nonsense traffic to produce *noise*. If the IDS is too busy to analyze the noise traffic the true attack traffic may go undetected.

# Outline

- 1 Introduction
- 2 Ambiguities**
- 3 IP Fragments
- 4 TCP Attacks

## General problem of ambiguity

Sometimes traffic analysis requires additional information from the network, e. g.

- Is TTL too large for destination?
- Is MTU too large for destination?
- Does destination accept source-routed packets?
- How does destination handle overlapping fragments?
- etc ...

## TTL-based attacks

- 1 Attacker sends fragments 1 and 2. But fragment-2 has TTL=1 and will be dropped by the router between IDS and destination host!
- 2 Attacker sends the last fragment (3) and IDS can now reassemble all fragments (1,2,3) and drop the packet. However the destination host only possesses fragments 1 and 3.
- 3 Finally, the attacker sends another fragment-2 (now with normal TTL) which allows the destination host to reassemble a malicious packet; however at this time the IDS only has fragment-2 in its reassembly buffer.

This is actually an **insertion** attack because the IDS sees more fragments as the destination host.

## Packet discard points in the kernel

A *very* interesting question: Which packets are discarded by an OS at which point in the stack?

- This is different in each operating system!
- Find out what is checked last (cause much CPU cycles → DoS)
- Find out which type of invalid packets are not detected as such

# Outline

- 1 Introduction
- 2 Ambiguities
- 3 IP Fragments**
- 4 TCP Attacks

## Different reassembly timer - too short!

When IDS has **shorter** IP fragment reassembly timer than destination host, the attacker could:

- 1 Send fragment-1
- 2 Wait until timeout expires on IDS and the IDS clears its reassembly buffer; the destination host still keeps fragment-1 in its reassembly buffer!
- 3 Send fragment-2
- 4 Now both fragments are in the reassembly buffer of the destination host while the IDS has only fragment-2 in its buffer!

## Different reassembly timer - too long!

When IDS has **longer** IP fragment reassembly timer than destination host, the attacker could:

- 1 Send fragment-2 and fragment-4
- 2 Wait until timeout expires on the destination host; now the fragments only exist in the reassembly buffer of the IDS
- 3 Send fragment-1 and fragment-3
- 4 Now the destination host has only fragment-1 and fragment-3 but the IDS has four fragments (1,2,3,4) that make up a valid packet, hence the IDS clears the buffer.
- 5 Send fragment-2 and fragment-4 which will complete a malicious packet at the destination host. The IDS thinks these fragments belong to a new packet. . .

## Fragment resending

Hacker sends fragments 1,2,3 and then 2,3,4; fragments 2 and 3 contain new data but same IP headers:

- Windows assumes only the first fragments are valid
- Unix (also Cisco IOS) assumes only the recent fragments are valid

IDS must know how end-systems work. . .

## Fragment overlapping

Offset of second fragment is too small – so second fragment may partly overwrite (the tail of) the first fragment. Again this depends on the operating system.

Again, the IPS must reassemble the fragments in the same manner as the operating system of the end-system.

# Outline

- 1 Introduction
- 2 Ambiguities
- 3 IP Fragments
- 4 TCP Attacks**

# Packet authenticity

We assume unauthenticated packets in general (no IPsec or similar). We must differentiate two cases:

- 1 **Connection-oriented protocols** (such as TCP) – can be tracked more or less reliably because of observable session states and parameters.

The IDS should monitor the usage of SQNRs; big gaps or sporadic anomalous SQNRs might be interpreted as blind spoofing attacks.

The IDS must also track the Window size and the Urgent pointer...

- 2 **Connectionless protocols** (UDP in general, e. g. DNS) can be easily spoofed and there is no way (except message authentication and maybe unicast RPF) to detect this.

## Session Monitoring

TCP input processing is complex; many parameter values, combinations of, depending on the TCP state, are invalid.

The IDS must monitor TCP sessions by creating a session entry upon every new valid session → **TCP Control Block (TCB)**.

The IDS can be confused via insertion attacks at three critical points:

- TCB creation
- Stream reassembly
- TCB teardown

Stateful firewalls and anti-spoofing ACLs can significantly reduce this vulnerability.

## TCP header parameters

IDS must check header (flags!) before looking at the data. Some combinations are invalid.

RFC 793 even permits TCP-SYN packets to contain data! This is differently handled by practical operating systems.

Some operating systems do not validate checksums. Some applications (e. g. some SMTP or IMAP implementations) create a corrupt checksum → insertion attacks!

Theoretically, the IDS could detect malformed segments through the response message sent by the inside host (complex).

## TCP options

Various TCP options may appear at different points within a session  
Some operating systems ignore bad options, others reject the whole packet. The IDS should know the behavior. . .

RFC 1323 defines PAWS<sup>1</sup> which allows hosts to add a timestamp in an option:

- If timestamp is lower than a threshold the packet is considered old and dropped
- This can be abused by the attacker (sending packets with correct SQNR but low timestamp); the IDS would need to know whether PAWS is enabled and also the current threshold. . .

---

<sup>1</sup>Protection against wrapped sequence numbers

## TCB Creation

**Upon three-way handshake only:** IDS might miss sessions for which the 3WHS has not been observed, e. g.

- Sessions that already existed before IDS has started
- The attacker already created a similar but invalid 3WHS with the host — subsequent sessions with same parameters but different SQNRs will be rejected by the IDS
- Under a high traffic load the IDS may be too busy and miss some packets of the 3WHS

In order to support certain options such as PAWS the IDS **must** see the 3WHS.

## TCB Creation (cont.)

**Upon partial handshake:** This may also work more reliable under high traffic load but which part of the 3WHS is trustworthy?

- The SYN is not
- The SYN-ACK comes from the inside device, but maybe we have an embryonic session?
- The ACK is an indication that the handshake has been completed – but an attacker could send ACK floods; therefore we must validate the SQNRs and socket parameters gathered from one of the previous packets.

Therefore a stateful firewall supporting e. g. SYN-Cookies is of great help!

## TCB Creation (cont.)

**Synching on data:** Look at data packets!

The advantages are:

- No session-loss upon packet floods
- Can detect old sessions (that exist before IDS has started)

The disadvantages:

- The attacker can send forged packets that lead to TCB creation
- The attacker can desync the IDS much easier

## TCB Creation (cont.)

Therefore the IDS should *additionally* observe some of the 3WHS packets to **reinitialize** its TCB (all previously monitored packets belong to a faked session)

- Looking for SYN packets: An attacker can desync the IDS by sending forged SYNs
- Looking for SYN-ACKs: Only works if the IDS knows where the client and the server is located

So this is generally a bad solution. . .

## Desynchronization

The IDS (as passive device) cannot easily determine the reason of missing packets (packet drop? out of order?)

It cannot ask the peer to re-send the data; therefore if the IDS misses data it is difficult to resync. The attacker can try to desync the IDS permanently, then no segment reassembly is possible.

## Window-based attacks

A TCP-peer continuously announces the number of bytes it is able to receive 'at once' (i. e. without ACKs inbetween) using the **window** parameter.

- If an attacker inserts packets with SQNRs beyond the current window, the receiving host will reject them.
- Therefore the IDS must track the window parameters
- Additionally the ACK-numbers of the inside host gives a good indication which data has been accepted already.
- But not every packet is ACKed – hosts send **cumulative** ACKs!
- Additionally the IDS must deal with duplicate ACKs, indicating missing or out-of-order packets.

## Same-SQNR segments

- The attacker could send segments with different data but same header parameters (same SQNR).
- Only one of those packets will be processed by the host – the IDS must know which. . .
- The ACKs from the host give no indication which data they really acknowledge!

This is a simple method to desynchronize the IDS.

## Overlapping segments

Segments can arrive out of order and in varying sizes: New data can overlap old data – similar as with IP fragments.

Typically UNIXes favor new data, Windows NT favor old data – in any case the IDS must process the segments equally to avoid evasion and insertion!

TCP allows so many parameters that it would be a tedious task to verify the IDS behavior under all circumstances

## TCB Teardown

- Necessary to free resources (memory and timers)
- TCP connections may last infinitely without any data being sent (unless the TCP keepalive feature is enabled)
- The attacker could create lots of sessions that are never FINished
- The attacker could fool the IDS and make it think that the session closes although it is still active — the IDS clears the TCB and stops pattern scanning for that session!

## TCB Teardown (cont.)

If the IDS does not recognize that a session is legitimately closed, the attacker could use the same session parameters (SQNRs, sockets)<sup>2</sup> for a new session — the IDS would ignore those packets!

---

<sup>2</sup>Actually the OS should wait some time before reusing the same session parameters, but typically nobody cares. . .

## TCB Teardown (cont.)

There are several ways to tear down a TCP session; all include the flags FIN and RST:

- The FIN flag is used for an ordered teardown; the inside host must also send a FIN, and both FINs must be ACKed – good hints for the IDS
- The RST flag is more critical: It is not ACKed, and some systems do not even check the SQNRs!

## Sneakers attack

- AN IDS could ignore teardown messages and only time out TCBs but then it can be easily desynchronized (if a new session uses the same socket parameters but different SQNRs)
- IDS *must* time out connections after some time
- If the attacker introduces long delays between his/her packets, the IDS might loose state
- Attacker can create lots of meaningless sessions to force the IDS to quicker time out older connections

# DoS

- Packet processing on IDS is very complex – expect lots of unknown bugs
- Some IDS dynamically create filters (dynamic threat mitigation) – fool the IDS via false positives
- Resource exhaustion – CPU, RAM, disk, bandwidth. Which packets are most expensive for the IDS?
  - Tiny IP fragments consume lots of CPU and some RAM
  - TCP connections consume lots of RAM
  - Analysis of encrypted packets
  - Analysis of ASN.1-coded packets (SNMP, H.323, ...)
  - Network flooding at a high rate causes IDS buffer overruns
  - Abuse reactive capabilities (e. g. senseless filter rules etc. against spoofed but important address)

# The TCP state machine

RFC 793 defines eleven TCP states:

- LISTEN** represents waiting for a connection request from any remote TCP and port.
- SYN-SENT** represents waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED** represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
- ESTABLISHED** represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
- FIN-WAIT-1** represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
- FIN-WAIT-2** represents waiting for a connection termination request from the remote TCP.
- CLOSE-WAIT** represents waiting for a connection termination request from the local user.
- CLOSING** represents waiting for a connection termination request acknowledgment from the remote TCP.
- LAST-ACK** represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
- TIME-WAIT** represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
- CLOSED** represents no connection state at all.

## Guidelines

- To prevent TTL-based insertion attacks, attach the IDS to the LAN of the destination hosts (or as close as possible)
- Monitor the CPU consumption of the IDS; it only works reliable if it has enough spare resources
- Deny fragmented packets at the perimeter
- Install a stateful firewall with SYN-Cookies (or TCP Intercept)
- Configure anti-spoofing ACLs on a screening router
- Regularly check for IDS security fixes; attackers search and focus on bugs

# Toolz

- Snot, Stick, Fragroute, Whisker/Nikto, Nessus
- Mausezahn (version 0.36 or newer)

Attacks against HTTP servers (especially IIS) is another wide field. Many servers have vulnerable URL-string parsers and related weaknesses. Consult the whisker/nikto papers for further information.