Testing the reassembly consistency of IDS and OS in the presence of overlapping data

"Journée thématique" GDR RSD, GPL and SI, Orléans

Presenter: Lucas Aubard

Supervisors: Johan Mazel, Gilles Guette

Pierre Chifflier

Ph.D dates: 01/10/2022 - 30/09/2025

30/09/2024

Plan

1. Context
2. Threat model
3. Method
4. Results

Plan

1. Context
2. Threat model
3. Method
4. Results

Context

Chunking mechanism in some Internet protocols

Generic networking problem

Application wants to send a lot of data and medium/underlying protocol is limited.

Solution

Chunk it

- Ethernet/IPv4||IPv6: fragmentation
- Ethernet/IP/TCP: segmentation

Context

Chunking mechanism in some Internet protocols: examples

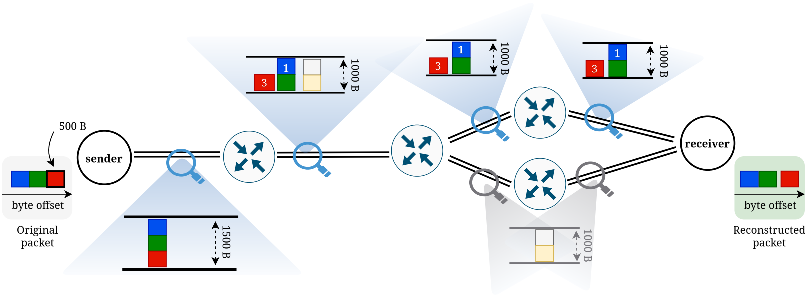


Figure 1: Normal chunk transmission

Context

Chunking mechanism in some Internet protocols: examples

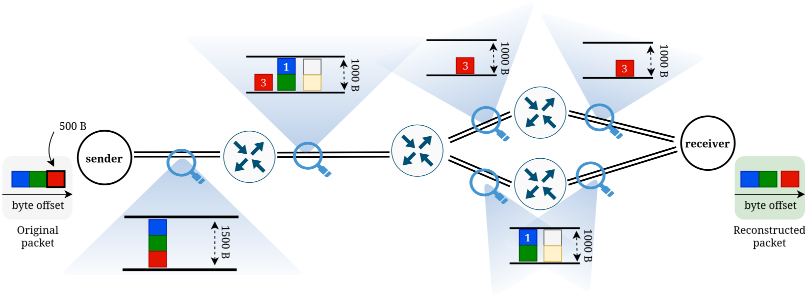


Figure 2: Chunk reordering

Context

Chunking mechanism in some Internet protocols: examples

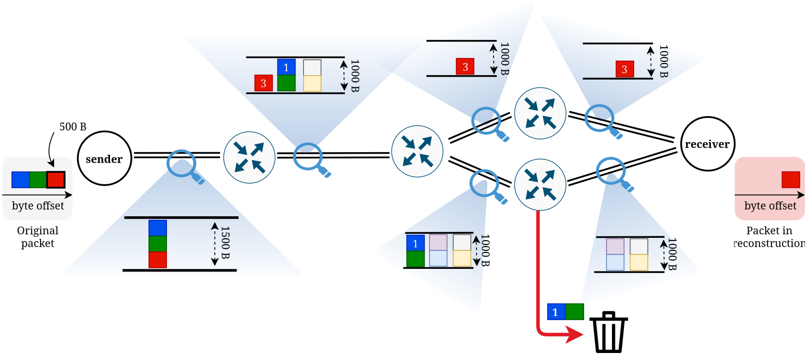


Figure 3: Chunk loss

Context

Chunking mechanism in some Internet protocols: examples

Reassembly policies may change depending on OSes for IPv4[1], IPv6[2], TCP[3] protocols and depending on QUIC implementations[4]

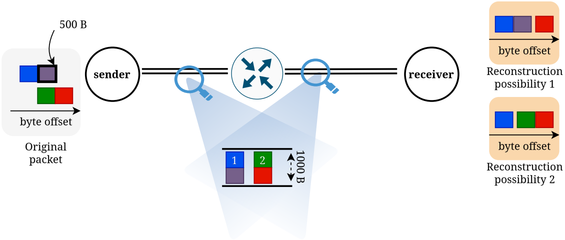


Figure 4: Chunk overlap

[1] J. Novak. *Target-based fragmentation reassembly*. 2005, U. Shankar **and** V. Paxson. *Active mapping: Resisting NIDS evasion withouts altering traffic*. 2003.

[2] A. Atlasis. *Attacking ipv6 implementation using fragmentation*. 2012.

[3] J. Novak **and** S. Sturges. *Target-based tcp stream reassembly*. 2007, U. Shankar **and** V. Paxson. *Active mapping: Resisting NIDS evasion withouts altering traffic*. 2003.

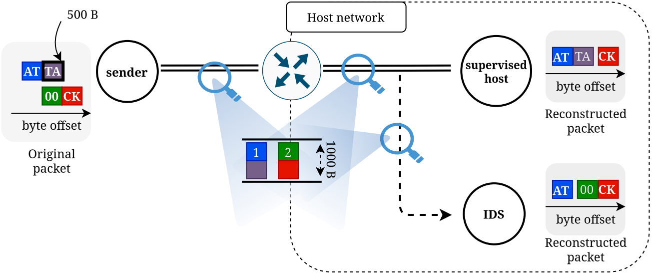
[4] G-S. Reen **and** C. Rossow. *DPIFuzz: a differential fuzzing framework to detect DPI elusion strategies for QUIC*. 2020.

Context

Attacks targetting IDSes using chunking mechanism

Problem

- Attacks targeting IDSes and exploiting data overlap exist[5]



Existing countermeasure

- manually configure an IDS to associate an IP address with a reassembly policy

[5] T. Ptacek **and** T. Newsham. *Insertion, evasion, and denial of service: Eluding network intrusion detection*. 1998.

Context

Considered attack types

Attack type	Host	Target	Reassembled data	Attack scenario
Evasion	IDS		-	E1
	End host	✗	"ATTACK"	
	IDS		"AT00CK"	E2
	End host	✗	"ATTACK"	
Insertion	IDS	✗	"ATTACK"	I1
	End host		-	
	IDS	✗	"ATTACK"	I2
	End host		"AT00CK"	

Table 1: Attack type illustration. - means the implementation *ignores* the flow chunk data.

Related work limits

- Manual or semi-automatic (fuzzing, symbolic execution) methods are used to generate overlap test cases

RQ1. *Are these methods exhaustive? If not, can we do better?*

- It's been 10 years no work have specifically addressed OSes' IPv4 and TCP policy reassemblies

RQ2. *Have the reassembly policies of recent OSes changed?*

- Some IDSes allow one to configure the host reassembly policy

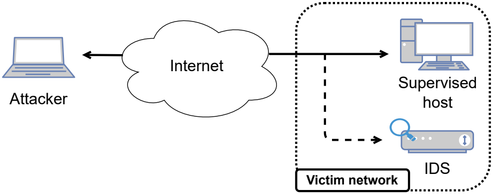
RQ3. *Do such IDSes reassemble consistently with OSes?*

Plan

Threat model

Attacker needs to:

- identify victim host OS and IDS reassembly policies.
- craft IP header fields and payload (IP fragment-based attack).
- craft TCP header fields and payload (TCP segment-based attack).



Plan

Test case modeling

Relation *R*	Interpretation	Relation *R* inverse	
X ***M*** Y	meets	X ***Mi*** Y	non-overlapping relations
X ***B*** Y	before	X ***Bi*** Y	
X ***Eq*** Y	equal	-	overlapping relations
X ***O*** Y	overlap	X ***Oi*** Y	
X ***S*** Y	start	X ***Si*** Y	
X ***D*** Y	during	X ***Di*** Y	
X ***F*** Y	finish	X ***Fi*** Y	

Table 2: Allen's interval algebra relations.

Test case modeling and related works

Relation ***R***	Illustration
X ***M***eets Y	Y X
X ***B***efore Y	Y X
X ***Eq***ual Y	Y X
X ***O***verlaps Y	Y X
X ***S***tarts Y	Y X
X ***D***uring Y	Y X
X ***F***inishes Y	Y X

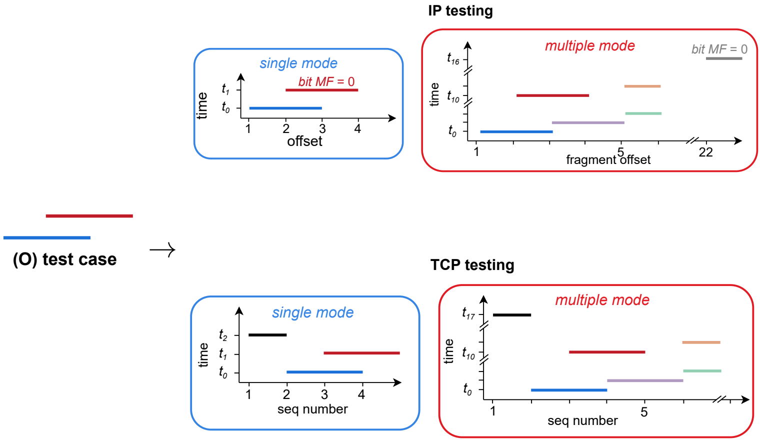
Table 3: Allen's interval algebra relations.

→

Author	**Work**	**Year**	**Protocol**	**Tested Allen relations**
Ptaceck et al.	[5]	1998	IPv4 /TCP	*Fi*, *D*
Shankar et al.	[7]	2003	IPv4	*O*, *Oi*, *Eq*
			TCP	*O*, *D*
Novak et al.	[3]	2005	IPv4	*O*, *Oi*, *S*, *Si*, *F*, *Fi*, *D*, *Di*, *Eq*
	[4]	2007	TCP	
Atlasis	[1]	2012	IPv6	*O*, *Oi*, *S*, *Si*, *F*, *Fi*, *D*, *Di*, *Eq*
Di Paolo et al.	[2]	2023	IPv6	*O*, *Oi*, *Eq*
Us	-	-	IPv4/IPv6/ TCP	*O*, *Oi*, *S*, *Si*, *F*, *Fi*, *D*, *Di*, *Eq*

Table 4: Summary regarding overlap-based works.

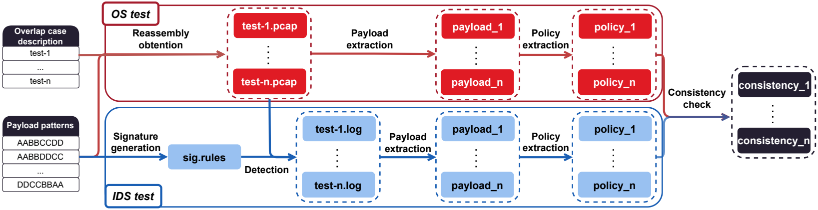
Test modes



(O) test case
IP testing
single mode
bit MF = 0
time
offset
multiple mode
bit MF = 0
time
fragment offset
TCP testing
single mode
time
seq number
multiple mode
time
seq number

Pyrolyse test pipeline

Easy to extend tool written in Rust that implements the following generic steps:



Plan

Results

OS reassembly policy evolution

OS	Protocol version	Testing mode	F	Fi	S	Si	O	Oi	D	Di	Eq
Windows 10	IPv4	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
		single	n	∅	n	o	∅	∅	n	o	n
	IPv6	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
		single	n	∅	n	o	∅	∅	n	o	n
	TCP	*multiple*	o	o	o	o	o	o	o	o	o
		single	o	o	o	o	o	o	o	o	o
Debian 12	IPv4	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
		single	n	∅	n	o	∅	∅	n	o	n
	IPv6	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
		single	n	∅	n	o	∅	∅	n	o	n
	TCP	*multiple*	n	o	o	o	o	n	n	o	o
		single	n	o	n	o	o	n	n	o	o
SunOS 5.11	IPv4	*multiple*	n	o	o	o	o	o	n	o	o
		single	n	∅	n	o	o	o	n	o	n
	IPv6	*multiple*	n	o	o	o	o	o	n	o	o
		single	n	∅	n	o	o	o	n	o	n
	TCP	*multiple*	n	o	n	o	n	o	n	o	n
		single	n	o	n	o	n	o	n	o	o
FreeBSD 13.1/ OpenBSD 7.4	IPv4	*multiple*	n	o	o	o	o	n	n	o	o
		single	n	∅	n	o	o	n	n	o	n
	IPv6	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
		single	n	∅	n	o	∅	∅	n	o	n
	TCP	*multiple*	n	o	o	o	o	n	n	o	o
		single	n	o	o	o	o	n	n	o	o

Results

Debian 12 reassembly policy evolution

Protocol	Testing mode	F	Fi	S	Si	O	Oi	D	Di	Eq
		Test case								
		Overlapping relation								
IPv4	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
	single	**n**	∅	**n**	**o**	∅	∅	**n**	**o**	**n**
IPv6	*multiple*	∅	∅	∅	∅	∅	∅	∅	∅	∅
	single	**n**	∅	**n**	**o**	∅	∅	**n**	**o**	**n**
TCP	*multiple*	n	o	**o**	o	o	n	n	o	o
	single	n	o	**n**	o	o	n	n	o	o

Table 5: IP and TCP reassembly policies of Debian 12. o means that oldest fragment data is prefered, n means that newest fragment data is prefered and ∅ means that the OS ignores the overlap. Bold blue means that *multiple* and *single* strategies are reassembled differently. Green (resp. red) means the observed reassembly is consistent (resp. inconsistent) with latest related works[6].

[6] J. Novak. *Target-based fragmentation reassembly*. 2005, J. Novak **and** S. Sturges. *Target-based tcp stream reassembly*. 2007, Edoardo Di Paolo, Enrico Bassetti **and** Angelo Spognardi. "A New Model for Testing IPv6 Fragment Handling". **in** *European Symposium on Research in Computer Security*. Springer. 2023, **pages** 277–294.

Results

IDS/OS consistency

Implementation	Rule file	Testing mode	F	Fi	S	Si	O	Oi	D	Di	Eq
			Test case								
			Overlapping relation								
Windows 10	-		**∅**	**∅**	**∅**	**∅**	**∅**	**∅**	**∅**	**∅**	**∅**
Suricata-*windows*	any	*multiple*	o	o	o	o	o	o	n	o	o
Snort-*windows*	any		o	o	o	o	o	o	n	o	o
Zeek	-		o	o	o	o	o	o	o	o	o
Windows 10	-		**n**	**∅**	**n**	**o**	**∅**	**∅**	**n**	**o**	**n**
Suricata-*windows*	default		n	∅	n	o	o	o	n	o	n
Suricata-*windows*	flow		∅	∅	∅	∅	o	o	∅	∅	∅
Snort-*windows*	default	*single*	n	∅	n	o	o	o	n	o	n
Snort-*windows*	flow		∅	∅	∅	∅	o	o	∅	∅	∅
Zeek	-		n	o	n	o	o	o	n	o	n
Debian 12	-		**∅**	**∅**	**∅**	**∅**	**∅**	**∅**	**∅**	**∅**	**∅**
Suricata-*linux*	any	*multiple*	n	o	n	n	o	n	n	o	n
Snort-*linux*	any		n	o	n	n	o	n	n	o	n
Zeek	-		o	o	o	o	o	o	o	o	o
Debian 12	-		**n**	**∅**	**n**	**o**	**∅**	**∅**	**n**	**o**	**n**
Suricata-*linux*	default		n	o	n	o	o	n	n	o	n
Suricata-*linux*	flow		∅	o	∅	∅	o	n	∅	∅	∅
Snort-*linux*	default	*single*	n	∅	n	o	o	n	n	o	n
Snort-*linux*	flow		∅	∅	∅	∅	o	n	∅	∅	∅
Zeek	-		n	o	n	o	o	o	n	o	n
SunOS 5.11	-		**n**	**o**	**o**	**o**	**o**	**o**	**n**	**o**	**o**
Suricata-*solaris*	any	*multiple*	n	o	o	o	o	o	n	o	o
Snort-*solaris*	any		n	o	o	o	o	o	n	o	o
Zeek	-		o	o	o	o	o	o	o	o	o
SunOS 5.11	-		**n**	**∅**	**n**	**o**	**o**	**o**	**n**	**o**	**n**
Suricata-*solaris*	default		n	o	n	o	o	o	n	o	n
Suricata-*solaris*	flow		∅	o	∅	∅	o	o	∅	∅	∅
Snort-*solaris*	default	*single*	n	∅	n	o	o	o	n	o	n
Snort-*solaris*	flow		∅	∅	∅	∅	o	o	∅	∅	∅
Zeek	-		n	o	n	o	o	o	n	o	n
FreeBSD 13.1	-		**n**	**o**	**o**	**o**	**o**	**n**	**n**	**o**	**o**
Suricata-*bsd*	any	*multiple*	n	o	o	o	o	n	n	o	o
Snort-*bsd*	any		n	o	o	o	o	n	n	o	o
Zeek	-		o	o	o	o	o	o	o	o	o
FreeBSD 13.1	-		**n**	**∅**	**n**	**o**	**o**	**n**	**n**	**o**	**n**
Suricata-*bsd*	default		n	o	n	o	o	o	n	o	n
Suricata-*bsd*	flow		∅	o	∅	∅	o	o	∅	∅	∅
Snort-*bsd*	default	*single*	n	∅	n	o	o	n	n	o	n
Snort-*bsd*	flow		∅	∅	∅	∅	o	n	∅	∅	∅
Zeek	-		n	o	n	o	o	o	n	o	n

Table 6: IDS IPv4 reassembly policy consistency with OSes.

Results

IDS evasion and insertion attack opportunities

Protocol	IDS	Reassembly inconsistencies	Number of OSes w/ possible attack type	
			Evasion	Insertion
IPv4	Suricata	8 (22%)	1/4	4/4
	Snort	4 (11%)	0/4	2/4
	Zeek	9 (25%)	4/4	1/4
IPv6	Suricata	9 (25%)	0/4	4/4
	Snort	6 (17%)	0/4	3/4
	Zeek	28 (78%)	4/4	4/4
TCP	Suricata	1 (3%)	1/4	1/4
	Snort	1 (3%)	1/4	1/4
	Zeek	11 (31%)	3/4	3/4

Table 7: IDS inconsistencies with OS reassemblies and corresponding attack opportunities for the *single* test mode.

Responsible disclosure

Every reassembly inconsistency is a possible security issue

- communication with IDS developers
- Suricata already fixed the some misassemblies

Conclusion and future works

Conclusion

- OS reassembly policies evolve
- overlap-based attacks can still target IDSes $\rightarrow$ they must take into account OS reassembly evolutions

Future works

- Investigate $n > 2$ overlapping chunks
- Target more protocol implementations (e.g., offloaded stacks on NIC, embedded stacks)

Testing the reassembly consistency of IDS and OS in the presence of overlapping data

"Journée thématique" GDR RSD, GPL and SI, Orléans

Thanks!