Lecture 6: In Depth Passive Testing using Network Traces

Passive Testing Techniques for Communication Protocols

Dr. Jorge López, PhD.
jorgelopezcoronado[at]gmail.com

National Research
Tomsk
State
University

February 24, 2016

Outline

Passive Testing using Network Traces

We want to guarantee some properties always hold in the network traces

Passive Testing using Network Traces

We want to guarantee some properties always hold in the network traces

- For instance, "for all VSNP replies, a *corresponding* request should have existed before, and the reply number must be odd for an even request ID or even for an odd request ID"

Passive Testing using Network Traces

We want to guarantee some properties always hold in the network traces

- For instance, "for all VSNP replies, a *corresponding* request should have existed before, and the reply number must be odd for an even request ID or even for an odd request ID"
- Properties that search (match) the same packets under the same constraints and conditions, but having a different semantical meaning are possible

Passive Testing using Network Traces

We want to guarantee some properties always hold in the network traces

- For instance, "for all VSNP replies, a *corresponding* request should have existed before, and the reply number must be odd for an even request ID or even for an odd request ID"
- Properties that search (match) the same packets under the same constraints and conditions, but having a different semantical meaning are possible
  - For instance, the same property as shown before can be expressed as: "for all VSNP requests, a *corresponding* response should follow, and the reply number must be odd for an even request ID or even for an odd request ID"'

Passive Testing using Network Traces

We want to guarantee some properties always hold in the network traces

- For instance, "for all VSNP replies, a *corresponding* request should have existed before, and the reply number must be odd for an even request ID or even for an odd request ID"
- Properties that search (match) the same packets under the same constraints and conditions, but having a different semantical meaning are possible
  - For instance, the same property as shown before can be expressed as: "for all VSNP requests, a *corresponding* response should follow, and the reply number must be odd for an even request ID or even for an odd request ID"'
  - The chronological order of the packet **prototypes** (the characterization of a specific network packet) is preserved in any case

What to do if

What to do if

- We successfully observe the property holds for certain packets

What to do if

- We successfully observe the property holds for certain packets
- We fail to observe that the property holds for certain packets

What to do if

- We successfully observe the property holds for certain packets
- We fail to observe that the property holds for certain packets
- We concluded something w.r.t. certain property...

What to do if

- We successfully observe the property holds for certain packets
- We fail to observe that the property holds for certain packets
- We concluded something w.r.t. certain property...

When a property is evaluated over a set of network packets and the property concludes about that set of packets, the conclusion provided is known as a **verdict**

What to do if

- We successfully observe the property holds for certain packets
- We fail to observe that the property holds for certain packets
- We concluded something w.r.t. certain property...

When a property is evaluated over a set of network packets and the property concludes about that set of packets, the conclusion provided is known as a **verdict**

- Verdict sets are different from in On-line and Off-line approaches!

Passive Testing using Network Traces (Monitoring)

On-line VS. Off-line approaches

Verdicts for Passive Testing using Network Traces

For the purpose of demonstrating the possible verdicts and their differences between on-line and off-line approaches

Verdicts for Passive Testing using Network Traces

For the purpose of demonstrating the possible verdicts and their differences between on-line and off-line approaches

- Assume the VSNP trace:

ID:2	**ID:3**	**ID:4**	**ID:2**	**ID:4**	**ID:21**	**ID:21**
N:	N:	N:	N: 77	N: 89	N:	N: 101

Verdicts for Passive Testing using Network Traces

For the purpose of demonstrating the possible verdicts and their differences between on-line and off-line approaches

- Assume the VSNP trace:

ID:2	**ID:3**	**ID:4**	**ID:2**	**ID:4**	**ID:21**	**ID:21**
N:	N:	N:	N: 77	N: 89	N:	N: 101

- Assume the property:
  'for all VSNP requests, a *corresponding* response should follow, and the reply number must be odd for an even request ID or even for an odd request ID"'

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

Actions:

The Off-line Monitoring Verdicts

Read packets

ID:2

N:

Off-line Tester Storing queue / Memory

Actions:

Read REQ with ID = 2

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:2

N:

Actions:

Store packet in the requests to be replied queue

The Off-line Monitoring Verdicts

Read packets

ID:3

N:

Off-line Tester Storing queue / Memory

ID:2

N:

Actions:

Read REQ with ID = 3

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:2	**ID:3**
N:	N:

Actions:

Store packet in the requests to be replied queue

The Off-line Monitoring Verdicts

Read packets

ID:4

N:

Off-line Tester Storing queue / Memory

ID:2 **ID:3**

N: N:

Actions:

Read REQ with ID = 4

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Store packet in the requests to be replied queue

The Off-line Monitoring Verdicts

Read packets

ID:2

N: 77

Off-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Read RES with ID = 2

The Off-line Monitoring Verdicts

Read packets

ID:2

N: 77

Off-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Check to which stored packet it *corresponds*

The Off-line Monitoring Verdicts

Read packets

ID:2

N: 77

Off-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Verify the property

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:
Report PASS (+)

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:
Remove corresponding stored packet from the stored requests queue

The Off-line Monitoring Verdicts

Read packets

ID:4

N: 89

Off-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:

Read RES with ID = 4

The Off-line Monitoring Verdicts

Read packets

ID:4

N: 89

Off-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:

Check to which stored packet it *corresponds*

The Off-line Monitoring Verdicts

Read packets

ID:4

N: 89

Off-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:

Verify the property

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:
Report PASS (+)

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3

N:

Actions:

Remove corresponding stored packet from the stored requests queue

The Off-line Monitoring Verdicts

Read packets

ID:21

N:

Off-line Tester Storing queue / Memory

ID:3

N:

Actions:

Read REQ with ID = 21

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Store packet in the requests to be replied queue

The Off-line Monitoring Verdicts

Read packets

ID:21

N: **101**

Off-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Read RES with ID = 21

The Off-line Monitoring Verdicts

Read packets

ID:21

N: **101**

Off-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Check to which stored packet it *corresponds*

The Off-line Monitoring Verdicts

Read packets

ID:21

N: **101**

Off-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Verify the property

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:
Report FAIL (-)

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3

N:

Actions:

Remove corresponding stored packet from the stored requests queue

The Off-line Monitoring Verdicts

Read packets

EOT

(end of trace)

Off-line Tester Storing queue / Memory

ID:3

N:

Actions:

Read EOT

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

ID:3

N:

Actions:

For each packet left on memory report **?** (Attention! грильяж is on the line!)

The Off-line Monitoring Verdicts

Read packets

Off-line Tester Storing queue / Memory

Actions:
Report INCONCLUSIVE (?)! (What if the trace was cut before the packet arrived?)

The Off-line Monitoring Verdicts (cont.)

Note that: for the FAIL verdict to occur, at least a two stage (fail stage) verification should exist

The Off-line Monitoring Verdicts (cont.)

Note that: for the FAIL verdict to occur, at least a two stage (fail stage) verification should exist

1. Find a partially matching packet (for our previous example it translates to find a matching REQ/RES ID)
2. Verify the property (for our previous example it translates to verifying the odd/even even/odd property)

The Off-line Monitoring Verdicts (cont.)

Note that: for the FAIL verdict to occur, at least a two stage (fail stage) verification should exist

1. Find a partially matching packet (for our previous example it translates to find a matching REQ/RES ID)
2. Verify the property (for our previous example it translates to verifying the odd/even even/odd property)

- If the verification is performed in a single stage (for our previous example to check for a matching packet and the matching packet shouls hold the even/odd odd/even property), non-matched packets will output an inconclusive verdict

Off-line Monitoring Resource Consumption

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze
  - Worst-case scenario, all packets in the trace are stored, no verdict is provided until the end of the trace, all inconclusive (for our previous example, only REQs in the trace)

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze
  - Worst-case scenario, all packets in the trace are stored, no verdict is provided until the end of the trace, all inconclusive (for our previous example, only REQs in the trace)
  - An example of a property that will store all packets?

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze
  - Worst-case scenario, all packets in the trace are stored, no verdict is provided until the end of the trace, all inconclusive (for our previous example, only REQs in the trace)
  - An example of a property that will store all packets: "for all VSNP requests a VSNP with the same ID will follow"

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze
  - Worst-case scenario, all packets in the trace are stored, no verdict is provided until the end of the trace, all inconclusive (for our previous example, only REQs in the trace)
  - An example of a property that will store all packets: "for all VSNP requests a VSNP with the same ID will follow"
- The processing depends on the checking algorithm, a good algorithm will be polynomial with respect to the size of the maximal number of stored packets and the total number of individual checks (comparisons)

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze
  - Worst-case scenario, all packets in the trace are stored, no verdict is provided until the end of the trace, all inconclusive (for our previous example, only REQs in the trace)
  - An example of a property that will store all packets: "for all VSNP requests a VSNP with the same ID will follow"
- The processing depends on the checking algorithm, a good algorithm will be polynomial with respect to the size of the maximal number of stored packets and the total number of individual checks (comparisons)
  - Not that good if the size of the trace is somewhat big (in the order of GB $\mapsto$ millions of packets) and there are many stored packets

[1] At least the necessary stored elements

Off-line Monitoring Resource Consumption

- The [free] memory required is the at least total size of the network trace[1] to analyze
  - Worst-case scenario, all packets in the trace are stored, no verdict is provided until the end of the trace, all inconclusive (for our previous example, only REQs in the trace)
  - An example of a property that will store all packets: "for all VSNP requests a VSNP with the same ID will follow"
- The processing depends on the checking algorithm, a good algorithm will be polynomial with respect to the size of the maximal number of stored packets and the total number of individual checks (comparisons)
  - Not that good if the size of the trace is somewhat big (in the order of GB $\mapsto$ millions of packets) and there are many stored packets
  - The total number of checks can be split with a parallel algorithm

[1] At least the necessary stored elements

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

Actions:

The On-line Monitoring Verdicts

Incoming (live) packets

ID:2

N:

On-line Tester Storing queue / Memory

Actions:

Read REQ with ID = 2

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:2

N:

Actions:

Store packet in the requests to be replied queue

The On-line Monitoring Verdicts

Incoming (live) packets

ID:3

N:

On-line Tester Storing queue / Memory

ID:2

N:

Actions:

Read REQ with ID = 3

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:2	**ID:3**
N:	N:

Actions:

Store packet in the requests to be replied queue

The On-line Monitoring Verdicts

Incoming (live) packets

ID:4

N:

On-line Tester Storing queue / Memory

ID:2 **ID:3**

N: N:

Actions:

Read REQ with ID = 4

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:
Store packet in the requests to be replied queue

The On-line Monitoring Verdicts

Incoming (live) packets

ID:2

N: 77

On-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Read RES with ID = 2

The On-line Monitoring Verdicts

Incoming (live) packets

ID:2

N: 77

On-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Check to which stored packet it *corresponds*

The On-line Monitoring Verdicts

Incoming (live) packets

ID:2

N: 77

On-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:

Verify the property

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:2	**ID:3**	**ID:4**
N:	N:	N:

Actions:
Report PASS (+)

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:
Remove corresponding stored packet from the stored requests queue

The On-line Monitoring Verdicts

Incoming (live) packets

ID:4

N: 89

On-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:

Read RES with ID = 4

The On-line Monitoring Verdicts

Incoming (live) packets

ID:4

N: 89

On-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:

Check to which stored packet it *corresponds*

The On-line Monitoring Verdicts

Incoming (live) packets

ID:4

N: 89

On-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:

Verify the property

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3	**ID:4**
N:	N:

Actions:
Report PASS (+)

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

Remove corresponding stored packet from the stored requests queue

The On-line Monitoring Verdicts

Incoming (live) packets

ID:21

N:

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

Read REQ with ID = 21

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Store packet in the requests to be replied queue

The On-line Monitoring Verdicts

Incoming (live) packets

ID:21

N: **101**

On-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Read RES with ID = 21

The On-line Monitoring Verdicts

Incoming (live) packets

ID:21

N: **101**

On-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Check to which stored packet it *corresponds*

The On-line Monitoring Verdicts

Incoming (live) packets

ID:21

N: **101**

On-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:

Verify the property

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3	**ID:21**
N:	N:

Actions:
Report FAIL (-)

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

Remove corresponding stored packet from the stored requests queue

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

What's next?

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

Wait. . . wait until another packet comes (live capture)

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

Until when do we wait? What do we do with this left-alone packet (grilyazh question)

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

ID:3

N:

Actions:

Until a determined **timeout**

The On-line Monitoring Verdicts

Incoming (live) packets

On-line Tester Storing queue / Memory

Actions:
after the determined timeout, report TIME_FAIL (!)

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive?

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive, time fail!

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive, time fail!

Timeouts

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive, time fail!

Timeouts

- Timeouts can be treated as a global timeout or per prototype (for better control when to "discard" a packet)

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive, time fail!

Timeouts

- Timeouts can be treated as a global timeout or per prototype (for better control when to "discard" a packet)
- Timeout function runs in parallel. Verifies all the packets in all the queues and deletes the those packets that the time in the queue is bigger than the associated timeout

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive, time fail!

Timeouts

- Timeouts can be treated as a global timeout or per prototype (for better control when to "discard" a packet)
- Timeout function runs in parallel. Verifies all the packets in all the queues and deletes the those packets that the time in the queue is bigger than the associated timeout
- A great idea! However, concurrent access to stored packet queues can make implementations error prone (good luck in your laboratory (:)

The On-line Monitoring Verdicts (cont.)

Two stage verification is also needed for the FAIL verdict

- if not, instead of inconclusive, time fail!

Timeouts

- Timeouts can be treated as a global timeout or per prototype (for better control when to "discard" a packet)
- Timeout function runs in parallel. Verifies all the packets in all the queues and deletes the those packets that the time in the queue is bigger than the associated timeout
- A great idea! However, concurrent access to stored packet queues can make implementations error prone (good luck in your laboratory (:)
- Just kidding, I will provide you with a library :) (good luck to me making your laboratory):)

Two stage verification

Two stage verification

1-stage verification

TWO STAGE VERIFICATION

1-stage verification

```
if REQ
(
   REQ->VSNP->Num = NULL
)
then RES>REQ
(
   RES->TCP->srcP = 1010 &
   RES->VSNP->Num % 2 = 0 &
   RES->IP->srcIP = REQ->IP->dstIP &
   REQ->VSNP->ID = RES->VSNP->ID &
   REQ->VSNP->ID %2 != 0
)
```

TWO STAGE VERIFICATION

1-stage verification

```
if REQ
(
   REQ->VSNP->Num = NULL
)
then RES>REQ
(
   RES->TCP->srcP = 1010 &
   RES->VSNP->Num % 2 = 0 &
   RES->IP->srcIP = REQ->IP->dstIP &
   REQ->VSNP->ID = RES->VSNP->ID &
   REQ->VSNP->ID %2 != 0
)
```

2-stage verification

Two stage verification

1-stage verification

```
if REQ
(
   REQ->VSNP->Num = NULL
)
then RES>REQ
(
   RES->TCP->srcP = 1010 &
   RES->VSNP->Num % 2 = 0 &
   RES->IP->srcIP = REQ->IP->dstIP &
   REQ->VSNP->ID = RES->VSNP->ID &
   REQ->VSNP->ID %2 != 0
)
```

2-stage verification

```
if
(
   if REQ
   (
      REQ->VSNP->Num = NULL
   )
   then RES>REQ
   (
      RES->TCP->srcP = 1010 &
      RES->IP->srcIP = REQ->IP->dstIP &
      REQ->VSNP->ID = RES->VSNP->ID
   )
)
then VAL
(
   RES->VSNP->Num % 2 = 0 &
   REQ->VSNP->ID %2 != 0
)
```

On-line Monitoring Resource Consumption

On-line Monitoring Resource Consumption

- The [free] memory required depends of **many** parameter

On-line Monitoring Resource Consumption

- The [free] memory required depends of **many** parameter
  - The traffic flow ratio (max = bandwidth), the particular timeout for each prototype, the required stored packets depending on the properties to check, the worse case scenario, bandwidth*timeout (if all packets should be stored)

On-line Monitoring Resource Consumption

- The [free] memory required depends of **many** parameter
  - The traffic flow ratio (max = bandwidth), the particular timeout for each prototype, the required stored packets depending on the properties to check, the worse case scenario, bandwidth*timeout (if all packets should be stored)
- The processing depends on the checking algorithm, a good algorithm will be polynomial with respect to the size of the maximal number of stored packets and the total number of individual checks (comparisons)

On-line Monitoring Resource Consumption

- The [free] memory required depends of **many** parameter
  - The traffic flow ratio (max = bandwidth), the particular timeout for each prototype, the required stored packets depending on the properties to check, the worse case scenario, bandwidth*timeout (if all packets should be stored)
- The processing depends on the checking algorithm, a good algorithm will be polynomial with respect to the size of the maximal number of stored packets and the total number of individual checks (comparisons)
  - The timeout **may** actually help to reduce the complexity (since there will be less stored packets given the timeout). Normally it will have a better performance than off-line, but, it might produce false positives, given the nature of on-line monitoring this trade-off is accepted

On-line Monitoring Resource Consumption

- The [free] memory required depends of **many** parameter
  - The traffic flow ratio (max = bandwidth), the particular timeout for each prototype, the required stored packets depending on the properties to check, the worse case scenario, bandwidth*timeout (if all packets should be stored)
- The processing depends on the checking algorithm, a good algorithm will be polynomial with respect to the size of the maximal number of stored packets and the total number of individual checks (comparisons)
  - The timeout **may** actually help to reduce the complexity (since there will be less stored packets given the timeout). Normally it will have a better performance than off-line, but, it might produce false positives, given the nature of on-line monitoring this trade-off is accepted
  - The total number of checks can be split with a parallel algorithm

On-line VS. Off-line Network Monitoring – Final Remarks

The difference:

On-line VS. Off-line Network Monitoring – Final Remarks

The difference:

- In Essence, when to verify properties over the collected traces (of course not saved to a file if it is on-line)

On-line VS. Off-line Network Monitoring – Final Remarks

The difference:

- In Essence, when to verify properties over the collected traces (of course not saved to a file if it is on-line)
- On-line is supposed to provide information ASAP. To provide information in 10ms is better than 3h.

On-line VS. Off-line Network Monitoring – Final Remarks

The difference:

- In Essence, when to verify properties over the collected traces (of course not saved to a file if it is on-line)
- On-line is supposed to provide information ASAP. To provide information in 10ms is better than 3h.
- Off-line is targeted to check conformance of the implementations over long periods of time

On-line vs. Off-line Network Monitoring – Final Remarks

The difference:

- In Essence, when to verify properties over the collected traces (of course not saved to a file if it is on-line)
- On-line is supposed to provide information ASAP. To provide information in 10ms is better than 3h.
- Off-line is targeted to check conformance of the implementations over long periods of time

Specific Requirements:

On-line vs. Off-line Network Monitoring – Final Remarks

The difference:

- In Essence, when to verify properties over the collected traces (of course not saved to a file if it is on-line)
- On-line is supposed to provide information ASAP. To provide information in 10ms is better than 3h.
- Off-line is targeted to check conformance of the implementations over long periods of time

Specific Requirements:

- On-line needs to have better performance so that its processing can be "keep-up" with the generated data

On-line vs. Off-line Network Monitoring – Final Remarks

The difference:

- In Essence, when to verify properties over the collected traces (of course not saved to a file if it is on-line)
- On-line is supposed to provide information ASAP. To provide information in 10ms is better than 3h.
- Off-line is targeted to check conformance of the implementations over long periods of time

Specific Requirements:

- On-line needs to have better performance so that its processing can be "keep-up" with the generated data
- Off-line needs to guarantee better fault (or property violation) coverage

Distributed Passive Testing Techniques
(using Network Traces)

Distributed Network Monitoring

What's the motivation?

Distributed Network Monitoring

What's the motivation?

- There exists properties that can only be verified when **correlating** network traces from different P.O.s

Distributed Network Monitoring

What's the motivation?

- There exists properties that can only be verified when **correlating** network traces from different P.O.s
- I won't go into detail which properties, but, there are...and there are many

Distributed Network Monitoring

What's the motivation?

- There exists properties that can only be verified when **correlating** network traces from different P.O.s
- I won't go into detail which properties, but, there are...and there are many

Architecture

Distributed Network Monitoring

What's the motivation?

- There exists properties that can only be verified when **correlating** network traces from different P.O.s
- I won't go into detail which properties, but, there are...and there are many

Architecture

- Under the assumption that all traces from all P.O.s are necessary to evaluate the property, a tester that collects all those traces is necessary, therefore a centralized tester

Distributed Network Monitoring

What's the motivation?

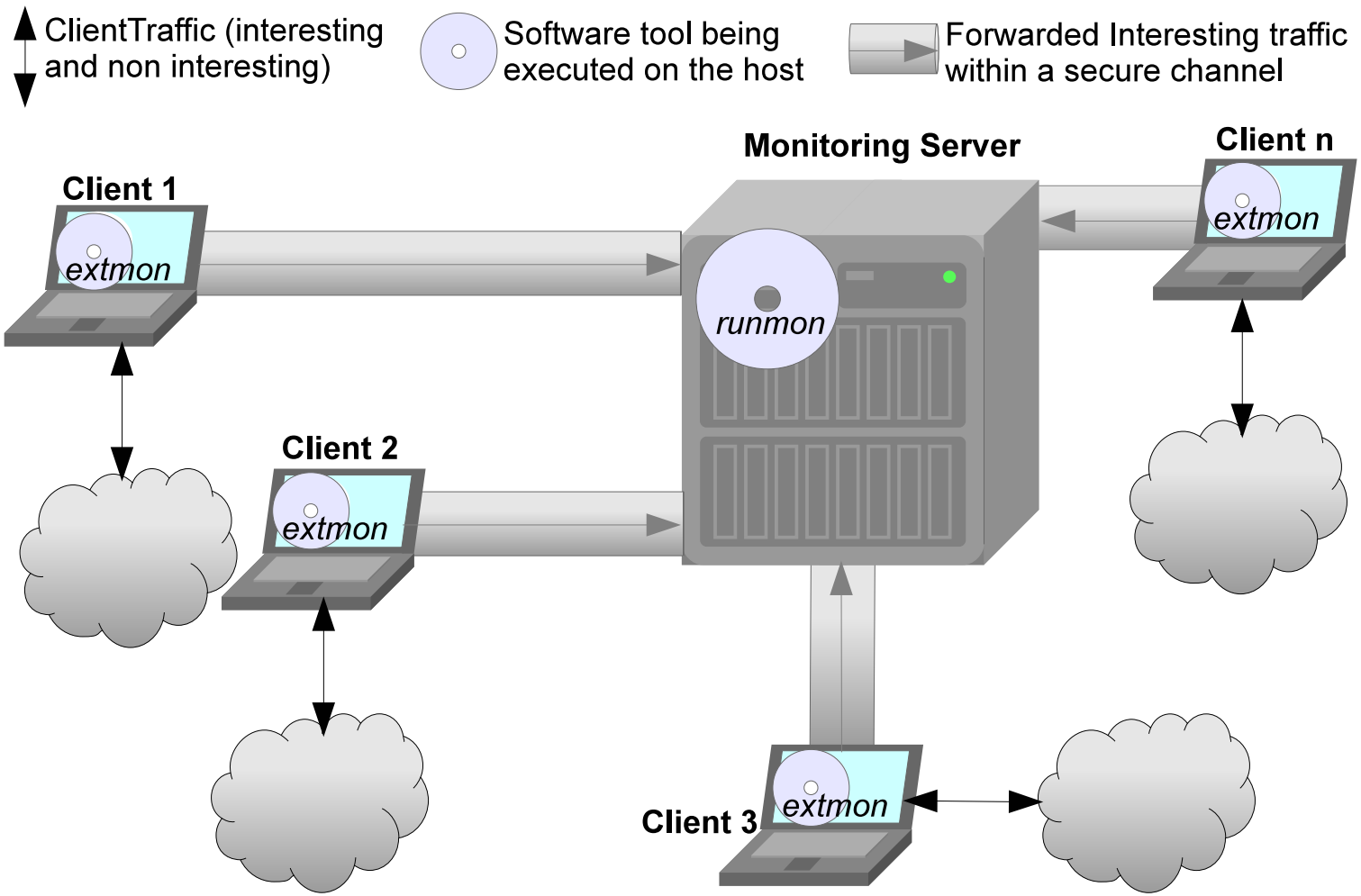
- There exists properties that can only be verified when **correlating** network traces from different P.O.s
- I won't go into detail which properties, but, there are...and there are many

Architecture

- Under the assumption that all traces from all P.O.s are necessary to evaluate the property, a tester that collects all those traces is necessary, therefore a centralized tester
- All P.O.s (or clients) should send the filtered traffic to the centralized tester (yes, this is the tool which I should upload to a repo)

Distributed Network Monitoring (cont.)

Architecture



Network Trace Correlation

What is to “correlate“ a trace [with another]?

Network Trace Correlation

What is to “correlate“ a trace [with another]?

- In the general case to make comparisons from the values of one P.O. with another

Network Trace Correlation

What is to “correlate“ a trace [with another]?

- In the general case to make comparisons from the values of one P.O. with another
- A P.O., a host which sends its data to the centralized server? or one IP address of that host? or one network interface of that host?

Network Trace Correlation

What is to “correlate“ a trace [with another]?

- In the general case to make comparisons from the values of one P.O. with another
- A P.O., a host which sends its data to the centralized server? or one IP address of that host? or one network interface of that host?

Time synchronization

Network Trace Correlation

What is to "correlate" a trace [with another]?

- In the general case to make comparisons from the values of one P.O. with another
- A P.O., a host which sends its data to the centralized server? or one IP address of that host? or one network interface of that host?

Time synchronization

- It represents one of the most difficult issues to overcome since there is no way to guarantee the chronological arrangement of packets in different P.O.s

Network Trace Correlation

What is to "correlate" a trace [with another]?

- In the general case to make comparisons from the values of one P.O. with another
- A P.O., a host which sends its data to the centralized server? or one IP address of that host? or one network interface of that host?

Time synchronization

- It represents one of the most difficult issues to overcome since there is no way to guarantee the chronological arrangement of packets in different P.O.s
- Distributed properties can be guaranteed under the assumption that all hosts are synchronized with an external time server, usign for intance the Network Time Protocol (NTP)

Fundamentals of Packet Analysis Programming

LIBPCAP – THE HOLY LIBRARY TRACE CAPTURE

Executive Summary:

LIBPCAP – THE HOLY LIBRARY TRACE CAPTURE

Executive Summary:

- Captures the received network packets at local interfaces

LIBPCAP – THE HOLY LIBRARY TRACE CAPTURE

Executive Summary:

- Captures the received network packets at local interfaces
- Portable, there exists even WinPCAP (don't even think about this (;)

LIBPCAP – THE HOLY LIBRARY TRACE CAPTURE

Executive Summary:

- Captures the received network packets at local interfaces
- Portable, there exists even WinPCAP (don't even think about this (;)
- To listen to all traffic going through a network interface, admin privileges are required!

LIBPCAP – THE HOLY LIBRARY TRACE CAPTURE

Executive Summary:

- Captures the received network packets at local interfaces
- Portable, there exists even WinPCAP (don't even think about this (;)
- To listen to all traffic going through a network interface, admin privileges are required!
- It provides some information about the captured packet (time of capture, size, etc.), and the packet as a block of bytes (unsigned chars)

LIBPCAP – THE HOLY LIBRARY TRACE CAPTURE

Executive Summary:

- Captures the received network packets at local interfaces
- Portable, there exists even WinPCAP (don't even think about this (;)
- To listen to all traffic going through a network interface, admin privileges are required!
- It provides some information about the captured packet (time of capture, size, etc.), and the packet as a block of bytes (unsigned chars)
- It does NOT provide all the passive testing using network traces concepts, prototypes, storing queues, etc.

Packet Analysis: The data structures

The data structures, should be defined tailored to the protocol

Packet Analysis: The data structures

The data structures, should be defined tailored to the protocol

- For example, for the Session Initialization Protocol (SIP):

Packet Analysis: The data structures

The data structures, should be defined tailored to the protocol

- For example, for the Session Initialization Protocol (SIP):

```
typedef struct start_line_tag{
   char *method;
   char *request_URI;
   char *version;
   short status_code;
   char *reason_phrase;
}start_line_s;

typedef struct header_field_tag{
   char *name;
    char *value;
}header_field;

typedef struct sip_packet_tag{
   start_line_s *start_line;
   linked_list *header_fields;
   char *message_body;
}sip_packet;
```

Packet Analysis: The data structures

The data structures, should be defined tailored to the protocol

- For example, for the Session Initialization Protocol (SIP):

```
typedef struct start_line_tag{
   char *method;
   char *request_URI;
   char *version;
   short status_code;
   char *reason_phrase;
}start_line_s;

typedef struct header_field_tag{
   char *name;
    char *value;
}header_field;

typedef struct sip_packet_tag{
   start_line_s *start_line;
   linked_list *header_fields;
   char *message_body;
}sip_packet;
```

- a function is needed to transform the byte stream to this data structure:

Packet Analysis: The data structures

The data structures, should be defined tailored to the protocol

- For example, for the Session Initialization Protocol (SIP):

```
typedef struct start_line_tag{
   char *method;
   char *request_URI;
   char *version;
   short status_code;
   char *reason_phrase;
}start_line_s;

typedef struct header_field_tag{
   char *name;
    char *value;
}header_field;

typedef struct sip_packet_tag{
   start_line_s *start_line;
   linked_list *header_fields;
   char *message_body;
}sip_packet;
```

- a function is needed to transform the byte stream to this data structure:

```
sip_packet*
   getSIPFromStream(unsigned char stream, int siz)
```

Passive Testing using Network Traces – Final Remarks

Passive Testing using Network Traces – Final Remarks

- Very useful when no access to the code is possible, real data analysis is desirable, and the system cannot be influenced by test cases

Passive Testing using Network Traces – Final Remarks

- Very useful when no access to the code is possible, real data analysis is desirable, and the system cannot be influenced by test cases
- It can be computationally expensive, nonetheless, good algorithms are being studied to perform the property checking. This is an interesting are of research you might be interested

Passive Testing using Network Traces – Final Remarks

- Very useful when no access to the code is possible, real data analysis is desirable, and the system cannot be influenced by test cases
- It can be computationally expensive, nonetheless, good algorithms are being studied to perform the property checking. This is an interesting are of research you might be interested
- Many languages exist to describe the properties to check, however, all have shortcomings. Studying a language that can describe such systems can be highly appreciated (specially if it is formal / has an equivalent automata model)

Passive Testing using Network Traces – Final Remarks (cont.)

Passive Testing using Network Traces – Final Remarks (cont.)

- The area of distributed passive testing using network traces can use some work

Passive Testing using Network Traces – Final Remarks (cont.)

- The area of distributed passive testing using network traces can use some work
  - Studying not centralized testers

Passive Testing using Network Traces – Final Remarks (cont.)

- The area of distributed passive testing using network traces can use some work
  - Studying not centralized testers
  - Time synchronization (I have an interesting idea that needs to be implemented, it's stated in my thesis as future work (:)

Passive Testing using Network Traces – Final Remarks (cont.)

- The area of distributed passive testing using network traces can use some work
  - Studying not centralized testers
  - Time synchronization (I have an interesting idea that needs to be implemented, it's stated in my thesis as future work (:)
- That is all for our lectures, I hope you enjoyed them! See you on the last laboratory!

Passive Testing using Network Traces – Final Remarks (cont.)

- The area of distributed passive testing using network traces can use some work
  - Studying not centralized testers
  - Time synchronization (I have an interesting idea that needs to be implemented, it's stated in my thesis as future work (:)
- That is all for our lectures, I hope you enjoyed them! See you on the last laboratory!

Спасибо за внимание!