SFPM Threat Mitigation

Ulisses Araújo Costa (ucosta)

Cisco Systems Inc.

December 12, 2010

Index

1 Introduction to Testing

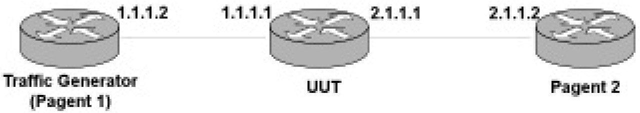
2 Sessin-based Flexible Packet Matching (SFPM)

3 SFPM Tests
- Testcases
- Traffic generators

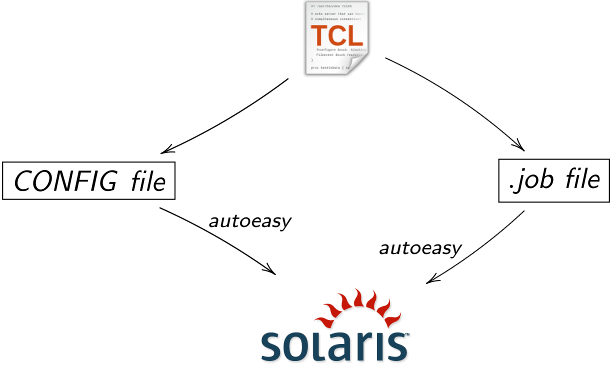
4 Performance Tests

Tests Jargon

- ATS - Automated Test Solutions
- eARMS - Extended Automated Regression Management System
- TFT - Test Feature Tracker
- TIMS - Test Information Management System
- TRADe - Test Results Analysis and Debugging
- Testbed



ATS Setup



CONFIG file

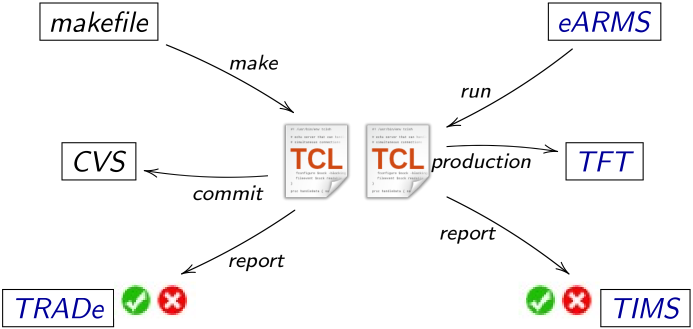
.job file

autoeasy

autoeasy

- Scripts written in Tcl
- CONFIG file with topology of the testbed (plus cleanup and setup)
- .job file with all calls to the scripts

Management of Scripts and Results



makefile

make

eARMS

run

CVS

commit

production

TFT

report

report

TRADe ✓ ✗

✓ ✗ TIMS

Index

1. Introduction to Testing
2. Sessin-based Flexible Packet Matching (SFPM)
3. SFPM Tests
   - Testcases
   - Traffic generators
4. Performance Tests

FPM vs *SFPM*

Flexible Packet Matching (*FPM*)

- Current *FPM* is a stateless per packet classification mechanism.
- *FPM* works well when the filter information exists in all packets of a flow.
- However, *FPM* can only apply actions to the those packets, and miss the rest of the packets in the same flow.

Session-based Flexible Packet Matching (*SFPM*)

- *SFPM* allows customers to create their own filtering policies that can immediately detect and block attacks.
- Session-based *FPM* allows session-based classification and actions.

Configuration Example - Action

```
router(config)#load fpm
Try to load bundle PHDF files ...
router(config)#class-map type access-control match-all c1
router(config-cmap)#match field TCP source-port eq 1024
router(config-cmap)#class-map type access-control match-any c2
router(config-cmap)#match start TCP payload-start offset 0 size 5 regex "GET /"
router(config-cmap)#policy-map type access-control p1
router(config-pmap)#class c1
router(config-pmap-c)#log all
router(config-pmap-c)#class c2
router(config-pmap-c)#log all
router(config-pmap-c)#policy-map type access-control fpm1
router(config-pmap)#class ip_tcp_stack
router(config-pmap-c)#service-policy p1
router(config-pmap-c)#interface FastEthernet0/1
router(config-int)#service-policy type access-control input fpm1
```

- Match TCP source-port number
- Match TCP payload regular expression
- log the sessions
- Attach the policy to the interface

Configuration Example - Nested

```
router(config)#load fpm
Try to load bundle PHDF files ...
router(config)#class-map type access-control match-all c1
router(config-cmap)#match field ICMP type eq 8
router(config-cmap)#class-map type access-control match-all c2
router(config-cmap)#match field ICMP checksum eq 123456
router(config-cmap)#class-map type access-control match-all c3
router(config-cmap)#match class c1 session
router(config-cmap)#policy-map type access-control p1
router(config-pmap)#class c3
router(config-pmap-c)#drop all
router(config-pmap-c)#policy-map type access-control fpm1
router(config-pmap)#class ip_icmp_stack
router(config-pmap-c)#service-policy p1
router(config-pmap-c)#interface FastEthernet0/1
router(config-if)#service-policy type access-control input fpm1
```

- Match ICMP type
- Match ICMP checksum
- drop the sessions
- Attach the policy to the interface

Configuration Example - Session Packet Range

```
router(config)#load fpm
Try to load bundle PHDF files ...
router(config)#class-map type access-control match-all c2
router(config-cmap)#match field TCP source-port eq 1024
router(config-cmap)#class-map type access-control match-all c3
router(config-cmap)#$ TCP payload-start offset 0 size 5 regex "GET /"
router(config-cmap)#class-map type access-control match-all c1
router(config-cmap)#match class c3 packet-range 3 4
router(config-cmap)#policy-map type access-control p1
router(config-pmap)#class c1
router(config-pmap-c)#log all
router(config-pmap-c)#policy-map type access-control fpm1
router(config-pmap)#class ip_tcp_stack
router(config-pmap)#service-policy p1
router(config-pmap-c)#interface FastEthernet0/1
router(config-if)#service-policy type access-control input fpm1
```

- Match TCP source-port
- Match TCP regexp (HTTP)
- log all sessions that have this match between packet 3 and 4
- Attach the policy to the interface

SFPM Demo

SFPM Demo

Index

1. Introduction to Testing
2. Sessin-based Flexible Packet Matching (SFPM)
3. SFPM Tests
   - Testcases
   - Traffic generators
4. Performance Tests

Index

1 Introduction to Testing

2 Sessin-based Flexible Packet Matching (SFPM)

3 SFPM Tests
- Testcases
- Traffic generators

4 Performance Tests

TCP/UDP/ICMP testcases - case1_config

1. Add a filter to existing class-map
2. Remove then add a new filter to existing class-map
3. Add a SFPM action for class-map
4. Remove action from class-map
5. Add class-map to using policy-map
6. Remove class-map from policy-map
7. Add child class-map in stack class-map
8. Remove child class-map from stack class-map
9. Remove child policy-map
10. Remove parent policy-map

TCP/UDP/ICMP testcases - case2_config

1. Add nested class session into class-map
2. Remove nested class session from class-map
3. Add filter in nested class
4. Remove filter from nested class
5. Add action into nested class
6. Remove action from nested class
7. Remove parent class-map (contains nested class) in policy-map
8. Add parent class-map (contains nested class) in policy-map
9. Remove child policy-map (contains nested class)
10. Remove parent policy-map attached to interface
11. Create consecutive nested class-map
12. Create circular nested class

TCP/UDP/ICMP testcases - case3_config

1. Add nested class session into class-map
2. Remove nested class session from class-map
3. Add filter in nested class
4. Remove filter from nested class
5. Add action into nested class
6. Remove action from nested class
7. Remove parent class-map (contains nested class) in policy-map
8. Add parent class-map (contains nested class) in policy-map
9. Remove child policy-map (contains nested class)
10. Remove parent policy-map attached to interface
11. Create consecutive nested class-map
12. Create circular nested class
13. Check packet range number

TCP/UDP/ICMP testcases - Case Ingress

```
                    +------------+
             fpm1   |            |
Client  ...  =======>|            |===== ...  Server
                G0/3|    U U T   |
                    |            |
                    |            |
                    +------------+
```

TCP/UDP/ICMP testcases - Case Egress

```
                    +------------+
                    |            | fpm1
Client  ... ========|            |=====> ... Server
                    |    U U T   |G0/2
                    |            |
                    |            |
                    +------------+
```

TCP/UDP/ICMP testcases - Case Ingress+Egress

```
                    +------------+
             fpm1   |            | fpm1
Client  ... =======>|            |====> ... Server
              G0/3  |    U U T   |G0/2
                    |            |
                    |            |
                    +------------+
```

TCP/UDP/ICMP testcases - Case Input+Output

```
                    +------------+
       |            |            |
       |     fpm1   |            |
       |... =======>|            |
       |      G0/3  |    U U T   |
Client |            |            |===== ... Server
       |     fpm3   |            |
       |... <=======|            |
       |       G0/3 |            |
       |            |            |
                    +------------+
```

Testcases - CEF/Process

Cisco Express Forwarding

- Cisco's Express Forwarding (CEF) is an advanced, Layer 3 switching technology inside a router. It defines the fastest method by which a Cisco router forwards packets from ingress to egress interfaces.
- Process switching uses the CPU on every packet, CEF only needs to the CPU for the first packet of each session.

TCP/UDP/ICMP testcases - case1_<config|traffic>_<cef|process>_<TCP|UDP|ICMP>

Action policies with log all as action

- case1 config
- case1_traffic_cef_TCP
- case1_traffic_process_TCP
- case1_traffic_cef_UDP
- case1_traffic_process_UDP
- case1_traffic_cef_ICMP
- case1_traffic_process_ICMP

TCP/UDP/ICMP testcases - case1_2_traffic_<cef|process>_<TCP|UDP|ICMP>

Nested policies with log as action

- case1_2_traffic_process_TCP
- case1_2_traffic_cef_TCP
- case1_2_traffic_process_UDP
- case1_2_traffic_cef_UDP
- case1_2_traffic_process_ICMP
- case1_2_traffic_cef_ICMP

TCP/UDP/ICMP testcases - case2_<config|traffic>_<cef|process>_<TCP|UDP|ICMP>

Nested policies with log all as action

- case2_config
- case2_traffic_cef_TCP
- case2_traffic_process_TCP
- case2_traffic_cef_UDP
- case2_traffic_process_UDP
- case2_traffic_cef_ICMP
- case2_traffic_process_ICMP

Multiple Flows testcases

For each testcase send multiple-flows TCP/UDP/ICMP traffic

- action_multiple_flow_process
- action_multiple_flow_cef
- nested_multiple_flow_cef_log
- nested_multiple_flow_cef_logAll
- nested_multiple_flow_process_log
- nested_multiple_flow_process_logAll

Change Configuration testcases - Change config

- action_change_config_cef
- action_change_config_process
- nested_change_config_cef
- nested_change_config_process

Method

1. Create a new policy
2. Send TCP traffic
3. In the midlle of traffic sending change the policies

Change Configuration testcases - Apply config

- action_apply_config_cef
- action_apply_config_process
- nested_apply_config_cef
- nested_apply_config_process

Method

1. Delete all the policies
2. Send TCP traffic
3. In the midlle of traffic sending apply the policies

Change Configuration testcases - Delete config

- action_delete_config_cef
- action_delete_config_process
- nested_delete_config_cef
- nested_delete_config_process

Method

1. Create a new policy
2. Send TCP traffic
3. In the midlle of traffic sending delete the policies

Bugs

CSCtg61173 UDP classification fails for output direction in cef

CSCtg60872 Regex classification is not working in TCP traffic with input+output

CSCtg61221 SFPM (FFPM) Stateful classification in input and output direction

Index

1. Introduction to Testing
2. Sessin-based Flexible Packet Matching (SFPM)
3. SFPM Tests
   - Testcases
   - Traffic generators
4. Performance Tests

Pagent Make your NETWORK

PAGENT is an IOS Based Testing Tool used to generate and capture, emulate large routed networks, and generate session based traffic.
The test tools are included in special IOS Pagent images.

TGN	Traffic Generator - generates TCP/UDP/ICMP traffic
HTTPSE	HTTP Session Emulator
PKTS	Packet Count and Capture

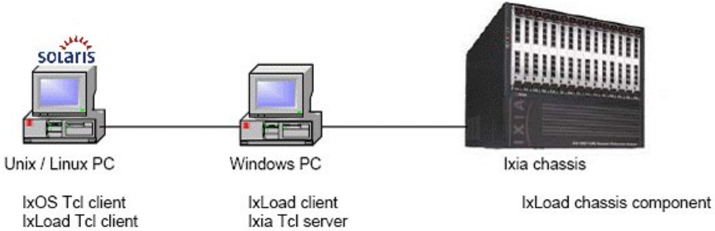
Index

1. Introduction to Testing
2. Sessin-based Flexible Packet Matching (SFPM)
3. SFPM Tests
   - Testcases
   - Traffic generators
4. Performance Tests

Why *IxLoad*?

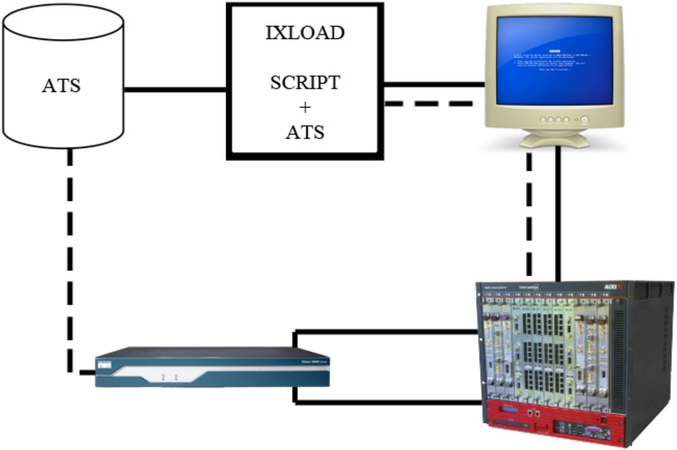
- Works on L4 and up
- Creates real-world traffic scenarios
- Emulate clients and servers of *HTTP*, *SSL*, *FTP*, *SMTP*, *POP*3

Getting started - *Ixia* Lab Setup



- SunOS 5.10 running in sjc-cde-006
- PC running Windows in 172.27.241.81
- *Ixia* chassis in 172.27.240.23

Whole picture



Getting started - Download

- Dowload[1] compatible versions[2]
  - IxLoad 4.30 EA SP1 Build Number: 4.30.119.78
  - IxOS 5.50 EA SP3 (Early Adopter) Build Number: 5.50.500.27
- Make sure you have installed in chassis and in Windows PC compatible versions
  - If you have multiple *IxOS* and/or *IxLoad* versions, force the system to use the one you want (with Ixia Application Selector)
- If you don't have an *Ixia* account you can request for one from their web site.

[1] Download and Updates page

[2] Compatibility Matrix

Getting started - Install

I will suppose that you already have *ATS* installed under `$ATS_USER_PATH`

- Install first *IxOS* and then *IxLoad* on Windows PC and *Ixia* Chassis if so needed
- For Solaris 10 machine
  - Install *IxOS* under `$IXIA_ATS_FOLDER`[3]
  - Install *IxLoad* under `$IXIA_HOME`

[3] This variable must be created by you, see next slide for further understanding

Getting started - Install - Solaris

Your .bashrc file should look like this

```
IXIA_ATS_FOLDER="/auto/stg-devtest/ucosta/"

IXIA_HOME="${IXIA_ATS_FOLDER}/ixia"
IXIA_VERSION="5.50.500.27"
IXIA_RESULTS_DIR="${HOME}/results_ixia"
IXIA_LOGS_DIR="${HOME}/logs_ixia"
IXIA_TCL_DIR="${IXIA_HOME}/lib"
TCLLIBPATH="${IXIA_TCL_DIR}"

ATS_USER_PATH="${IXIA_ATS_FOLDER}/ats"
AUTOTEST="$ATS_USER_PATH"
ATS_EASY="$ATS_USER_PATH"

MANPATH="${MANTPATH}:${IXIA_HOME}/man:/usr/local/man:/usr/man:/usr/share/man:/
     usr/autotool/devel/man:"
PATH="${PATH}:${ATS_USER_PATH}/bin:$IXIA_HOME/bin:${ATS_USER_PATH}/man:"
export ATS_USER_PATH AUTOTEST ATS_EASY PATH MANPATH LD_LIBRARY_PATH IXIA_HOME
     IXIA_VERSION IXIA_RESULTS_DIR IXIA_LOGS_DIR IXIA_TCL_DIR TCLLIBPATH
     IXIA_ATS_FOLDER
```

Getting started - Install - Solaris - part 2

After change the .bashrc file[4] don't forget to type:

```
[ucosta@sjc-cde-006:/]-$ source $HOME/.bashrc
```

If the installation of *IxLoad* in Solaris 10 fails

You can activate the debug flag and then try to understand whats wrong (`log.txt` file):

```
[ucosta@sjc-cde-006:ixia]-$ export LAX_DEBUG=true
[ucosta@sjc-cde-006:ixia]-$ ./IxLoadTclAPI4.30.119.78 2> log.txt
```

[4] If you use csh as your shell, translate the previous code

Getting started - Install - Solaris - part 3

If the installation of *IxLoad* in Solaris 10 fails and you run out of patience

You can copy my *IxLoad* directory into your *Ixia* folder.

```
[ucosta@sjc-cde-006:ixia]-$ cp -r /auto/stg-devtest/ucosta/ixia/IxLoadTclAPI4
     .30.119.78-EB $IXIA_HOME
```

I also have a zip file that contains this folder, if you want copy that and unzip it in your folder

```
[ucosta@sjc-cde-006:ixia]-$ cp /auto/stg-devtest/ucosta/ixia/IxLoad4.30EASP1.zip
      $IXIA_HOME
[ucosta@sjc-cde-006:ixia]-$ cd $IXIA_HOME
[ucosta@sjc-cde-006:ixia]-$ unzip IxLoad4.30EASP1.zip
```

Getting started - Checking installation

If you follow the steps you should be able to see that *IxOS* and *IxLoad* are properly installed

```
[ucosta@sjc-cde-006:ixia]-$ expect
expect1.1> package require IxLoad
Tcl Client is running Ixia Software version: 5.50.500.27
4.30.119.78
expect1.2>
```

AutoEASY

AutoEASY files

CONFIG file contains how to access our routers, passwords, etc

Job file contains the scripts and parameters that need to be submitted for execution

Script is where the recipe is (in this case *ATS*+*IxLoad* tests)

ATS files - .job and CONFIG files

CONFIG file

```
#activate ATS debug
set LOG_LEVEL {
    aereport debug
}
set REPORTS ucosta@cisco.com
set TESTBEDS {ucosta_router_tb}
set ROUTERS(ucosta_router_tb) {ucosta_router}

global _device
set _device(ucosta_router) "telnet 172.19.218.32 2013"
TacacsPw {}
EnablePw {}
```

run.job file

```
ats_run -on_proc abc123 test.ixload test.ixload 1 ixia DEBUG  172.27.240.23
     "1,1,9  1,1,10" 100 full 172.27.241.81
```

ATS files - Makefile(optional)

Makefile to automate the process of run the test scripts

make run_log works only in Solaris machine

make watch its good to watch the output that is being generated

make run run the test scripts and send the output to sdtout

make clean keep our dir clean

makefile

```
run_log:run.job CONFIG
    autoeasy -D run.job -cf CONFIG > log
watch:log
    watch -n 1 ’cat log | tail -n 30’
run:run.job CONFIG
    autoeasy -D run.job -cf CONFIG
clean:
    rm -f *˜ *.*˜ *.log *.report *.rerun log
```

IxLoad+ATS script

For now we will use the GSBU Dev Test team framework[5]
We will use as example a generation of *HTTP* traffic.

Structure of the script

```
<imports>
<parse args>
test_config { ... }
test_analyze { ... }
test_unconfig { ... }
```

[5] Can be found in regression/tests/functionality/gsg/ after you checkout the most recent version of regression tests

IxLoad+ATS script - test_config

Structure of the script

```
test_config {
tg-ixiaLoad_connect $PCServerIP $tgArgs
tg-ixiaLoad_client_net -port $tgPort2 -firstIp
    $IxLoadClientIP  -firstMac 00:C6:12:02:01:00 -ipCount 1
     -networkMask $netmask -gateway 172.31.254.254 #
    configure client network
tg-ixiaLoad_server_net -port $tgPort1 -firstIp
    $IxLoadServerIP  -firstMac 00:B6:12:02:01:00 -ipCount 1
     -networkMask $netmask -gateway 172.16.254.254 #
    configure server network
tg-ixiaLoad_client_http_traffic -maxSessions 1 -pageList
    $pageList -httpVersion 1.1 #configure client
tg-ixiaLoad_server_http_traffic -httpPort 80 #configure
    server
}
```

IxLoad+ATS script - test_config - part 2

Structure of the script - cont.

```
test_config {
tg-ixiaLoad_client_traf_net_map -objectiveType
    concurrentConnections -objectiveValue 2000 -iterations 1
     -rampDownTime 10  -sustainTime 20 #configure client
    traffic
tg-ixiaLoad_server_traf_net_map #configure server traffic
tg-ixiaLoad_create_test #create test
}
```

IxLoad+ATS script - test_analyze

Structure of the script - cont.

```
test_analyze {
set ixLoadStats tg-ixiaLoad_run_test_with_stats #run HTTP
    test and get IxLoad stats
}
```

The result is given in the form of HashTable, we can access it by:

```
set clientBytesReceived [keylget ixLoadStats
    client,BytesReceived]
set clientBytesReceived [double $clientBytesReceived]
set serverBytesSent     [keylget ixLoadStats
    server,BytesSent]
set serverBytesSent     [double $serverBytesSent]
echo "--- clientPacketsReceived = $clientPacketsReceived"
echo "--- serverPacketsSent     = $serverPacketsSent"
```

Demo

Demo

Questions

?