IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 1, Ver. IV (Jan.-Feb. 2017), PP 16-22
www.iosrjournals.org

The Systematic Methodology for Accurate Test Packet Generation and Fault Localization

Palaparthi Sajiya[1], Dasari Ravi Kumar[2]
[#1]Student of M.Tech (CSE), Department of Computer Science & Engineering,
[#2] Assist.Prof, Department of Information Technology, QIS Institute of Technology, Ongole, AP, India.

Abstract: *As we probably aware now a days networks are broadly dispersed so administrators relies on upon different devices, for example, pings and follow course to troubleshoot the issue in network. So we proposed a robotized and orderly approach for testing and troubleshooting network called "Automatic Test Packet Generation"(ATPG). ATPG first peruses switch arrangement and produces a gadget free model. The model is utilized to produce least number of test packets to cover each connection in a network and every control in network. ATPG is equipped for researching both useful and execution issues. Test packets are sent at customary interims and separate strategy is utilized to confine flaws. The working of few disconnected devices which automatically create test packets are additionally given, yet ATPG goes past the prior work in static (checking liveness and fault localization).*

Keywords: *Fault Localization, Test Packet Selection, Network Debugging, Automatic Test packet Generation (ATPG), Forwarding Information Base (FIB).*

I. Introduction

We are ignorant of past strategies that automatically make test packets from designs and nearest operations, we perceive of disconnected apparatuses that ensure invariants in networks. There are a great deal of recommendations to develop engineering of estimation agreeable for networks [1]. Our work is related to work in programming dialects and additionally typical troubleshooting. In our work we think about a structure of automatic test packet generation (ATPG) which makes an immaterial arrangement of packets automatically to check the liveness of crucial topology and harmoniousness among information plane state and in addition setup determinations. This apparatus automatically makes packets to check execution declarations for example packet dormancy. ATPG produces test packets and additionally infusion focuses by method for existing sending of estimation gadgets. ATPG is not limited to liveness testing, however can be utilitarian to review of unrivaled level properties. Critical commitment of automatic test packet generation is not fault localization, however deciding a minimized end-to-end estimation that covers every run the show. ATPG show signs of improvement discovery granuality to manage level by method for using switch setup and in addition data of information plane [2]. In view of network representation, ATPG create unimportant number of test packets with the goal that every sending guideline inside network is secured by at least one test packet. At the point when a blunder is seen, ATPG make utilization of a fault restriction calculation to set up coming up diminutive standards.

II. Related Work

Analyzing android malware: Characterization and advancement Y. Zhou and X. Jiangis exhibited the prominence and selection of advanced cells has extraordinarily empowered the spread of portable malware, particularly on the prominent stages, for example, Android. In light of their fast development, there is a squeezing need to create compelling arrangements. Be that as it may, our protection ability is generally compelled by the restricted comprehension of these rising portable malware and the absence of opportune access to related specimens. In this paper, we concentrate on the Android stage and expect to systematize or describe existing Android malware. Especially, with over one year exertion, we have figured out how to gather more than 1,200 malware tests that cover the dominant part of existing Android malware families, running from their introduction in August 2010 to late ones in October 2011. Likewise, we deliberately portray them from different perspectives, including their establishment techniques, actuation systems and in addition the way of conveyed noxious payloads. The portrayal and a resulting development based investigation of delegate families uncover that they are advancing quickly to go around the location from existing versatile against infection programming. In view of the assessment with four delegate portable security programming, our tests demonstrate that the best case identifies 79.6% of them while the most pessimistic scenario distinguishes just 20.2% in our dataset. These outcomes obviously require the need to better create cutting edge hostile to versatile malware arrangements. 3. Ensuring against network diseases: A diversion theoretic point of view J. Omic, A. Orda, and P. V. Mieghemthe exhibited Security ruptures and assaults are basic issues in today's networking. A key-point is that the security of every host depends not just on the insurance techniques it receives additionally on those picked by different has in the network. The spread of Internet worms and infections is just a single case. This class of issues has two viewpoints. To start with, it manages pandemic procedures, and in that capacity requires the work of plague hypothesis. Second, the conveyed and self-sufficient nature of basic leadership in significant classes of networks (e.g., P2P, specially appointed, and most strikingly the Internet) require the work of diversion hypothetical methodologies. In like manner, we propose a brought together system that joins the N-entwined, SIS pandemic model with a no agreeable amusement show. We decide the presence of Nash harmony of the particular amusement and portray its properties. We demonstrate that its quality, as far as general network security, to a great extent relies on upon the fundamental topology. We then give a bound on the level of framework wastefulness because of the no helpful conduct, to be specific, the "cost of disorder" of the diversion. We watch that the cost of political agitation might be restrictively high; thus we propose a plan for directing clients towards socially productive conduct. 4. Control laws, pareto conveyances and zipf's law M. E. J. Newman is introduced the when the likelihood of measuring a specific estimation of some amount differs conversely as a force of that esteem, the amount is said to take after a power law, additionally referred to differently as Zipf's law or the Pareto circulation. Control laws show up broadly in material science, science, earth and planetary sciences, financial matters and fund, software engineering, demography and the sociologies. For example, the conveyances of the sizes of urban communities, seismic tremors, sun based flares, moon pits, wars and individuals' close to home fortunes all seem to take after influence laws. The starting point of powerlaw conduct has been a subject of open deliberation in established researchers for over a century. Here we audit a portion of the observational confirmation for the presence of force law frames and the hypotheses proposed to clarify them. 5. The impact of network topology on the spread of pandemics A. J. Ganesh, L. Massouli'e, and D. F. Towsleyis introduced the Many network marvels are very much demonstrated as spreads of plagues through a network. Noticeable illustrations incorporate the spread of worms and email infections, and, all the more by and large, issues. Many sorts of data scattering can likewise be displayed as spreads of plagues. In this paper we address the topic of what makes a pestilence either feeble or intense. All the more correctly, we distinguish topological properties of the diagram that decide the determination of scourges. Specifically, we demonstrate that if the proportion of cure to contamination rates is bigger than the ghostly sweep of the chart, then the mean pandemic lifetime is of request log n, where n is the quantity of hubs. Alternately, if this proportion is littler than a speculation of the isoperimetric consistent of the chart, then the mean plague lifetime is of request ena, for a positive steady a. We apply these outcomes to a few network topologies including the hypercube, which is an agent availability diagram for a dispersed hash table, the total chart, which is a critical network diagram for BGP, and the power law diagram, of which the AS-level Internet diagram is a prime case. We likewise ponder the star topology and the Erdos-Renyi chart as their plague spreading practices decide the spreading conduct of force law diagram.

III. Network Design

As mentioned in the last section, the automatic test packet generation (ATPG) system makes use of geometric model of header space analysis [4]. This section explains some of the key terms associated with geometric framework of header space analysis.

Packet

Packet in a network can be described as a tuple of the form (port, header) in such a way that, it is the job of port to show position of packet in a network at instantaneous time. Each one of the port is allotted with one and only one unique number [3].

Switch

Another term used in geometric model of header space analysis is switches. It is the job of switch transfer Function T, to model devices in a network. Example of devices can be switches or routers. There is a set of forwarding rules contained in each device, which decides how the packets should be processed. When a packet comes at a switch, a switch transfer function comperes it with each rule in descending order of priority. If packet does not match with any of the rule then it is dropped. Each incoming packet is coupled with exactly single rule [4].

Rules

Piece of work for rules is generation of list of one or more output packets associated with those output ports to which the packet is transferred, and explain how fields of port are modified. In other words, rules explains how the region of header space at entrance in changed into region of header space at exit [5].

Rule History

At any moment, every packet has its own rule history, which can be described as ordered list of rules packet have matched up to that point as it covers the network. Rule history provides necessary and important unprocessed material for automatic test packet generation (ATPG). That is the reason why it is fundamental to ATPG [6].

Topology

The network topology is modeled by topology transfer function. The topology transfer function gives the specification about which two ports are joined by links. Links are nothing but rules that forwards a packet from source to destination with no modification. If there is not a single topology rule matching an input port, the port is situated at edge of a network and packet has come to its desired destination [7].

Life of a Packet

One can see life of a packet as carrying out or executing switch transfer function and topology transfer function at length. When a particular packet comes in a network port p, firstly a switch function is applied to that packet. Switch transfer function also contains input port pk.p of that packet. The result of applying switch function is list of new packets [pk1, pk2, pk3,]. If the packet reached its destination it is recorded, and if that is not the case, topology transfer function is used to call upon switch function of new port. This process is done again and again unless packet is at its destination [8].

IV. ATPG Theory

Stand on the system standard analyzed above; Automatic test packet generation system makes use of least possible number of test packets to study whole forwarding rules in a network, on the condition that each forwarding rule is capped by at least one test packet. When the fault is encountered, ATPG is equipped with fault localization algorithm to resolve the declining rules or links. Figure 3 represents the work flow of automatic test packet generation (ATPG) system.

1) The ATPG system begins by gathering forwarding state from network, which is represented as first step in the figure. Work covered in this step is normally not only retrieving topology of network but also learning forwarding information base and configuration files etc.
2) The second step follows the first, in which header space analysis is used by ATPG system to figure out scope of each terminal.
3) The outcome of second step is taken as input by test packet generation algorithm to gauge smallest number of test packets sufficient to test all rules. This completes third step.
4) These test packets are sent regularly by the test terminals as a penultimate step.
5) Lastly, if an error is disclosed ATPG appeals to fault localization algorithm to curtail root of error [8].
6) Readers can see other version of figure 3 in figure 5 given in [1].

Origination of Test Packets

The ATPG system can be roughly divided into two parts namely test packet generation and fault localization. While developing an algorithm for test packet generation a supposition is that, set of test terminals may transmit or take in test packets. The target for algorithm is generating minimum number of test packets to practice every rule in every switch function, as a result if a fault occurs, it will be watched by at least one test packet. ATPG system makes use of test packets selection algorithm (TPS) to generate test packets.

ATPG must only make use of test terminals that are available and ATPG must utilize headers that each test terminal is authorized to send are two important restrictions of which ATPG must take a notice of at the time of generating test packets.

1) ATPG system begins by estimating entire set of test packet headers that can be forwarded from each test terminal to every other test terminal. ATPG achieves this by detecting full set of rules it can work out in entire journey. Thus, ATPG refers to all pair reachability algorithm [4] to perform this task.
2) Afterwards, ATPG selects greater than or equal to one test packet from identical class of test packets to use every rule which is within reachable distance. Automatic test packet generation can complete this with ease by haphazardly selecting single packet in each class. This method is capable of finding only those faults for which all packets screened by same rule suffer the same fault. Example of such faults includes link failure. On the other hand if someone desired to find out faults which are particular to headers, then he has to select every header in every class. This process is called sampling.
3) Lastly in the process of generating test packets ATPG goes to compression. Most of the times while using test packet selection algorithm there come situation such that same rule can be used by numerous test packets. Consequently ATPG chooses smallest family of test packets selected in above step in such a way that alliance of their rule histories cover total rules [1].

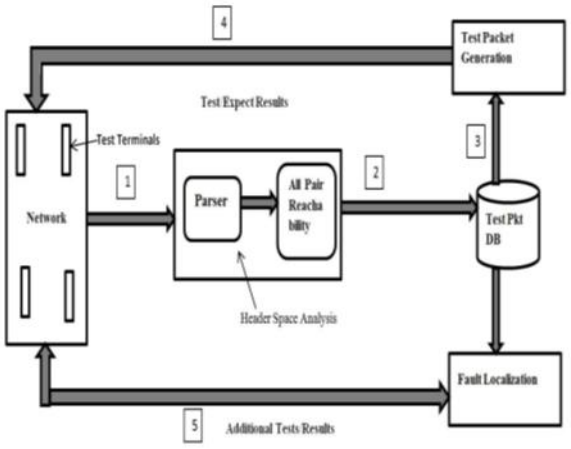

Fig 1: Working of Automatic Test Packet Generation

V. Proposed System

Automatic Test Packet Generation (ATPG) framework that automatically generates a minimal set of packets to test the liveness of the underlying topology and the congruence between data plane state and configuration specifications. The tool can also automatically generate packets to test performance assertions such as packet latency. It can also be specialized to generate a minimal setoff packets that merely test every link for network liveness. Below Figure shows the block diagram of ATPG system. The system first collects all the forwarding states from the network (step 1). This usually involves reading the FIBs, ACLsor config files and obtaining the topology. ATPG uses HeaderSpace Analysis [10] to find reachability between all the test terminals (step 2). The result is then used by the test packet selection algorithm to find a minimal set of test packets necessary for complete testing of all the rules in the network (step 3). These packets will be sent periodically in the network by the test terminals (step 4). Once an error is detected, the fault localization algorithm is invoked to narrow down the cause of the error (step 5).

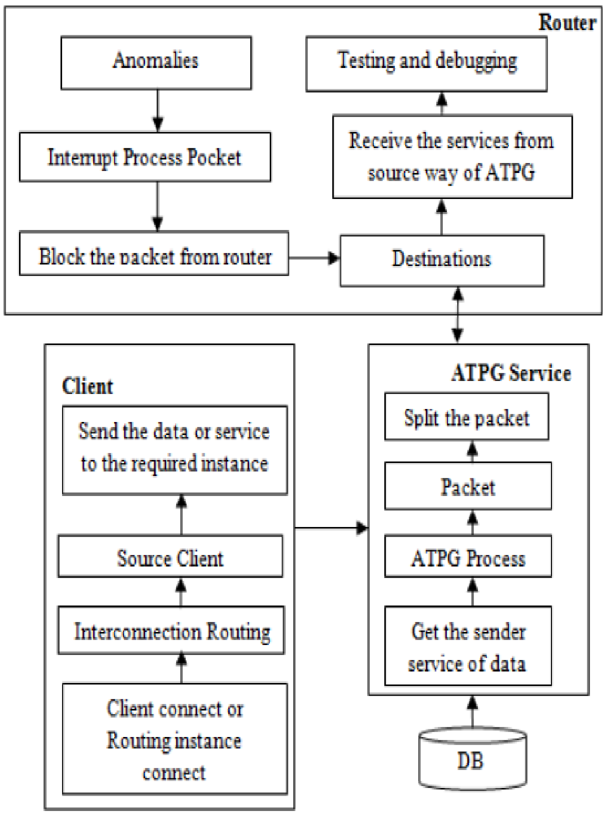
Router
Anomalies
Testing and debugging
Interrupt Process Pocket
Receive the services from source way of ATPG
Block the packet from router
Destinations
Client
Send the data or service to the required instance
Source Client
Interconnection Routing
Client connect or Routing instance connect
ATPG Service
Split the packet
Packet
ATPG Process
Get the sender service of data
DB

Fig 2. Proposed System Architecture

ATPG System

In view of the system model, ATPG creates the insignificant number of test parcels so that each sending govern in the system is practiced and secured by no less than one test bundle. At the point when a slip is distinguished, ATPG utilizes a flaw limitation calculation to focus the coming up short principles or connections. Fig. is a square chart of the ATPG framework. The framework first athers all the sending state from the system (step 1). This enerally includes perusing the FIBs, ACLs, and con fig documents, as well as acquiring the topology. ATPG utilizes Header Space Analysis [9] to register reachability between all the test terminals (step 2). The outcome is then utilized by the test parcel choice calculation to figure a negligible arrangement of test packets that can test all standards (step 3). These parcels will be sent occasionally by the test terminals (step 4). In the event that a lapse is identified, the flaw restriction calculation is summoned to tight down the reason for the blunder (step 5).

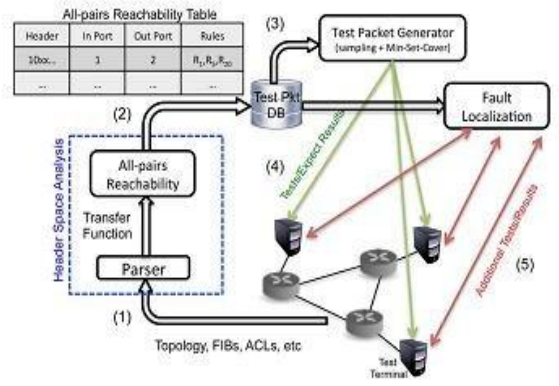

Fig 3. ATPG block diagram [1]

Algorithm

The life of a packet can be viewed as applying the switch and topology transfer functions repeatedly. When a packet Pk arrives at a network port, the switch function that contains the input port Pk.p is applied to Pk, producing a list of new packets [Pk1,…., Pkn]. If the packet reaches its destination, it is recorded. Otherwise, the topology function Γ is used to invoke the switch function containing the new port. The process repeats until packets reach their destinations (or are dropped).

```
Bit b=0|1|x
Header h=[b0,b1,….,bl]
Port p=1|2|…|N|drop
Packet pk=(p,h)
Rule r:pk→pk or [pk]
Match r.matchset : [pk]
Transfer Function T : pk→pk or [pk]
Topo Function Γ : (psrc,h) → (pdst,h)
Function Ti(pk)
#Iterate according to priority in switch i
For r є ruleseti do
If pk є r.matchset then
Pk.history ← pk.history U {r}
Return r(pk)
Return [(drop,pk.h)]
Input: Topology T ; End-to-end measurements fXkgk2R;
Y0 = 1;
W = ;;
recurse(1);
output: W;
subroutine recurse(k) {
if(k 2 R) fYk = Xkg;
else f
Yk = maxj2d(k) Yj ;
foreach(j 2 d(k)){
if (Yj = 0 && Yk = 1){
W = W [ fjg;
}
}
}
```

Fault Localization Algorithm

1) *Fault Model: A rule fails if its observed behavior differs from its expected behavior. ATPG keeps track of where rules fail using a result function. For a rule, the result function is defined as*

$$R(r, pk) = \begin{cases} 0, & \text{if } pk \text{ fails at rule } r \\ 1, & \text{if } pk \text{ succeeds at rule } r. \end{cases}$$

We divide faults into two categories: *action faults* and *match Faults.* An action fault occurs when *every* packet matching the rule is processed incorrectly. Action faults include unexpected packet loss, a missing rule, congestion, and miswiring. On the other hand, match faults are harder to detect because they only affect *some* packets matching the rule: for example, when a rule matches a header it should not, or when a rule misses a header it should match. We will only consider action faults because they cover most likely failure conditions and can be detected using only one test packet per rule.

2) *Problem 2 (Fault Localization):* Given a list of $(pk_{0,}\ (R(pk_0)),(pk_{1,}\ (R(pk_1))$ ... tuples, find all that satisfies $\exists pk_i, R(pk_i, r)=0$.

Step 1: Consider the results from sending the regular test packets. For every passing test, place all rules they exercise into a set of passing rules, *P*. Similarly, for every failing test, place all rules they exercise into a set of potentially failing rules *F*. By our assumption, one or more of the rules *F* are in error. Therefore *F-P*, is a set of *suspect rules*.

Step 2: ATPG next trims the set of suspect rules by weeding out correctly working rules. ATPG does this using the *reserved packets* (the packets eliminated by Min-Set-Cover). ATPG selects reserved packets whose rule histories contain *exactly one* rule from the suspect set and sends these packets. Suppose a reserved packet *p* exercises only rule *r* in the suspect set. If the sending of *p* fails, ATPG infers that rule *r* is in error; if *p* passes, *r* is removed from the suspect set. ATPG repeats this process for each reserved packet chosen in Step 2.

Step 3: In most cases, the suspect set is small enough after Step 2, which ATPG can terminate and report the suspect set. f needed, ATPG can narrow down the suspect set further by sending test packets that exercise two or more of the rules inthe suspect set using the same technique underlying Step 2. If these test packets pass, ATPG infers that none of the exercised rules are in error and removes these rules from the suspect set. If our Fault Propagation assumption holds, the method will notmiss any faults, and therefore will have no *false negatives*.

False Positives: Note that the localization method may introduce *false positives*, rules left in the suspect set at the end of Step 3. Specifically, one or more rules in the suspect set may in fact behave correctly. False positives are unavoidable in some cases. When two rules are in series and there is no path to exercise only one of them, we say the rules are *indistinguishable*; any packet that exercises one rule will also exercise the other. Hence, if only one rule fails, we cannot tell which one. For example, if an ACL rule is followed immediately by a forwarding rule that matches the same header, the two rules are indistinguishable. Observe that if we have test terminals before and after each rule (impractical in many cases), with sufficient test packets, we can distinguish every rule. Thus, the deployment of test terminals not only affects test coverage, but also localization accuracy.

VI. Conclusion

Nowadays network administrators depend on old apparatuses such as ping and traceroute to right the system. But as network is larger they need more refined instrument for right system. Due to huge network access administrator face issues in testing liveners of system. To overcome this issues we developed ATPG. By testing all guidelines comprehensive at all drop rules ATPG has capacity test reachability method. ATPG can figure execution soundness of a system. ATPG employments straightforward issue restriction strategy developed with the assistance of header space investigation to confine deficiencies. Customary model of ATPG framework serves to cover most extreme connections or standards in a system with least number of test packets.

References

[1]. Zeng, Kazemian, Varghese,and Nick “Automatic Test Packet Generation”,VOL. 22, NO. 2, APRIL 2014

[2]. Y. Bejerano and R. Rastogi, “Robust monitoring of link delays and faults in IP networks,” IEEE/ACM Trans Netw., vol. 14, no. 5, pp. 1092–1103, Oct. 2006

[3]. N. Duffield, “Network tomography of binary network performance characteristics,” IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.

[4]. N. Duffield, F. L. Presti, V. Paxson, andD.Towsley,“Inferringlink loss using striped unicast probes,” in Proc. IEEE INFOCOM, 2001, vol. 2, pp. 915–923.

[5]. Y. Bejerano and R. Rastogi, “Robust monitoring of link delays and faults in IP networks,” IEEE/ACM Trans. Netw., vol. 14, no. 5, pp. 1092–1103, Oct. 2006.

[6]. C. Cadar, D. Dunbar, and D. Engler, “Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs,” in Proc. OSDI, Berkeley, CA, USA, 2008, pp. 209–224.

[7]. M. Canini,D.Venzano, P. Peresini,D.Kostic, and J.Rexford, “A NICE way to test OpenFlow applications,” in Proc. NSDI, 2012, pp. 10–10.

[8]. A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, “Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data,” in Proc. ACM CoNEXT, 2007, pp. 18:1–18:12..

[9]. N. Duffield, “Network tomography of binary network performance characteristics,” IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.

[10]. N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, “Inferring link loss using striped unicast probes,” in Proc. IEEE INFOCOM, 2001, vol. 2, pp. 915–923.

Authors

Palaparthi Sajiya is Pursuing M.Tech (Computer Science and Engineering) in QIS Institute of Technology, Ongole, Prakasam Dist, Andhra Pradesh, India.

Dasari Ravi Kumar is currently working as Asst. Professor in QIS Institute of technology, in the Department of Information Technology, Ongole, Prakasam Dist, Andhra Pradesh, India.