Design and Evaluation of a Policy-Based Security Routing and Switching System for Data Interception Attacks

Yudong Zhao[1(✉)], Ke Xu[1], Rashid Mijumbi[2], and Meng Shen[3]

[1] Tsinghua University, 9-402 Room, East-main Building, Beijing, China
zhaoyd10@mails.tsinghua.edu.cn, xuke@mail.tsinghua.edu.cn
[2] Universitat Politècnica de Catalunya, Barcelona, Spain
rashid@tsc.upc.edu
[3] Beijing Institute of Technology, Beijing, China
shenmeng@bit.edu.cn

Abstract. In recent years, the world has been shocked by the increasing number of network attacks that take advantage of router vulnerabilities to perform data interceptions. Such attacks are generally based on low cost, unidirectional, concealed mechanisms, and are very difficult to recognize let alone restrain. This is especially so, because the most affected parties – the users and Internet Service Providers (ISPs) – have very little control, if any, on router vulnerabilities. In this paper, we design, implement and evaluate a policy-based security system aimed at stopping such attacks from both the routing and switching network functions, by detecting any violations in the set policies. We prove the system's security completeness to data interception attacks. Based on simulations, we show that 100% of normal packets can pass through the policy-based system, and about 99.92% of intercepting ones would be caught. In addition, the performance of the proposed system is acceptable with regard to current TCP/IP networks.

Keywords: Router vulnerabilities · Data interception attacks · Policy-based routing and switching system · Security completeness

1 Introduction

Data interception attacks (DIAs) that routers and switches take advantage of vulnerabilities to monitor and redirect traffic to third party have been seriously harming the security of users over many years. However, users and ISPs used to underestimate its impact and scale until June 2013 when it was revealed that the National Security Agency (NSA)'s PRISM Project was intercepting and collecting data across the Internet backbone [1][2]. By hacking huge internet routers, PRISM could get access to communications of hundreds of thousands of computers without the need to hack each one independently. These revelations have since underscored the importance of enhancing the security of routing and switching functions in TCP/IP networks.

Y. Wang et al. (Eds.): BigCom 2015, LNCS 9196, pp. 179–192, 2015.
DOI: 10.1007/978-3-319-22047-5_15

In addition to being low cost and having abilities to create high damage, DIAs can be characterized in two main ways: (1) they are stubborn in that not only are users and ISPs not able to easily detect and/or stop router vulnerabilities, but developers are also not able to eliminate them, (2) they are usually highly concealed such that it is almost impossible for ISPs to distinguish data interception packets from genuine ones. Such characteristics have made it difficult to develop and implement complete and practical security systems to detect and stop the attacks.

One simple idea for regulating routing and switching functions is to self-develop routers and switches to eliminate vulnerabilities. Unfortunately, self-developed routers and switches either lack efficiency, or are not able to completely eliminate vulnerabilities. As a result, we have to regulate routing and switching functions with secure mechanisms. However, current mechanisms are either only fit for specific core networks and specific types of DIAs, or are difficult to implement. To the best of our knowledge, there are still no security complete, universal and easily implementable mechanisms for DIAs.

In this paper, we propose a policy-based system with the objective of enhancing the security of routing and switching functions in TCP/IP networks. The proposed system is based on proposing a set of policies that routers and switches[1] can use to detect abnormal traffic behavior. In contrast to previous approaches, our proposal does not focus on the routing and switching operations, but rather, on changes to a 3-tuple (<Source IP, Destination IP, Payload>) characteristic code for each packet before and after it passes through a router. Compared with other mechanisms for stopping DIAs, the proposed system has three major features: (1) *security complete*, we prove that the set of policies are necessary and sufficient to stop DIAs that take advantage of router vulnerabilities, (2) *universal*, each policy of the system is applicable to all types of TCP/IP networks, and transparent to network topologies, routing protocols and configuration & management plans, and (3) *implementable*, the policy-based routers can be designed, and the simulation result shows that policy violations can be detected and restrained, yet the performance of the routers is acceptable with regard to current TCP/IP networks

The rest of this paper is organized as follows: We discuss related work in Section 2. In section 3, we present the policy-based security system for routing and switching in TCP/IP networks. The security completeness of this system to DIAs is proved in Section 4, and the final model of the policy-based router presented in Section 5. Section 6 evaluates the functionality and performance of the proposed system by simulation, and the paper is concluded in Section 7.

2 Related work

Researchers have been paying more attention to DIAs that result from router vulnerabilities. Most current solutions can be categorized as being based on three broad areas: (1) the level of routing and switching operation, (2) the OS and devices, and (3) core network.

[1] In the rest of this paper, we use “router” to mean both routers and switches.

With regard to the level of routing and switching operation, software dataplanes set against the risk of disrupting the network with bugs, unpredictable performance, or security vulnerabilities. Dobrescu et al. [3] present a verification tool by applying existing verification ideas and compositionality and combining them with certain domain specific packet processing software, to explore the feasibility of verifying software dataplanes to ensure smooth network operation. Although ensures software dataplanes both verification and performance, this tool relies on a given set of conditions, and is only effective for crash-freedom, bounded-execution, and filtering properties. Based on a novel combination of static analysis with symbolic execution and dynamic analysis with concrete execution, Kothari et al. [4] developed a method to help discover manipulation attacks in protocol implementations. However, since defenders do not always grasp the protocol code developed by third parties, this method is not effective for all types of protocol implementation attacks.

Considering the level of OSs and devices, the Trusted Computing Group developed and promoted Trusted Computing (TC) [5]. Challener et al. [6] showed that it is possible to provide a complete and open industry standard for implementing TC hardware subsystems in PCs. They demonstrated what TC can achieve, how it works, and how to write applications for it. However, although promoted for many years, there is still no accepted trusted computer model and effective theory or approach to dynamically quantize the degree of trustworthiness of software, which greatly limits the development of TC. Chen et al. [7] argue that the OS and applications currently running on a real machines should relocate into a virtual machine to service, which enables services, such as secure logging, intrusion prevention and detection, and environment migration, to be added without trusting or modifying the OS or applications. While its flexibility provides tremendous benefits for users, this approach undermines the fact that today's relatively static security architectures rely on the number of hosts in a system, their mobility, connectivity, patch cycle, etc. [8].

At the level of the core network, to construct high-level security mechanisms for secure connections, Kim et al. [9] proposed lightweight, scalable, and secure protocols for shared key setup, source authentication, and path validation. Although its computing overhead is low, there must be an Origin and Path Trace (OPT) header embedded into each packet head. The length of the header is variable according to path length. Even more, the longer the path is, the lower the throughput it produces under such protocols. With the aim of embedding security into the network as an internal service, and designing a secure network architecture, the idea of a Trustworthy Network (TN) is proposed by Peng et al. [10]. The authors propose three essential properties to interpret trustworthiness of a network, and discuss four key issues needed to be resolved. Although the idea of TN has been around for many years now, most works base on the theories or techniques in certain fields, and as a result, the complete system has not been built so far.

Finally, focusing on the security of the content of each IP packet, instead of routing and switching processes, IPSec [11] is aimed at ensuring communication safety by authenticating and encrypting each IP packet. Payloads and/or packets are encrypted, which prevents plain text from being intercepted by attackers. However,.IPSec cannot know what attackers try to intercept and where the attack host locates. In addition,

when the number of clients increases, available bandwidth to each client is the bottleneck that prevents the increase in number of connections handled by the web server [12].

In summary, it is our opinion that there are still no theories or solutions security complete, universal and implementable for DIAs. Our proposal is able to recognize and restrain intercepting packets efficiently with designable policy-based routers that are universal to any TCP/IP network, and can locate the attack hosts

3 The Policy-Based Security System for Routing and Switching in TCP/IP Networks

3.1 Motivation and Design Challenges

As the processing and communicating hubs of core networks, routers are usually designed in a fixed logical structure [13] and process packets according to a fixed series of operations. However, once embedded vulnerabilities, a router may work abnormally. Figure 1 shows the normal routing and switching process (left) and all possible abnormal behaviors (right). With abnormal behavior AB1 in figure 1, the packets sent by data forwarding plane are neither from upstream routers, nor generated by control plane. With AB2, the payloads of packets are changed or substituted. With AB3, although a packet seems to be forwarded normally, its output interface is altered secretly. Attackers can take advantage of such abnormal behaviors to perform data interception. As it is difficult for current router providers to eliminate vulnerabilities, and for ISPs to detect these three kinds of abnormal behaviors, DIAs can be highly stubborn and concealed.

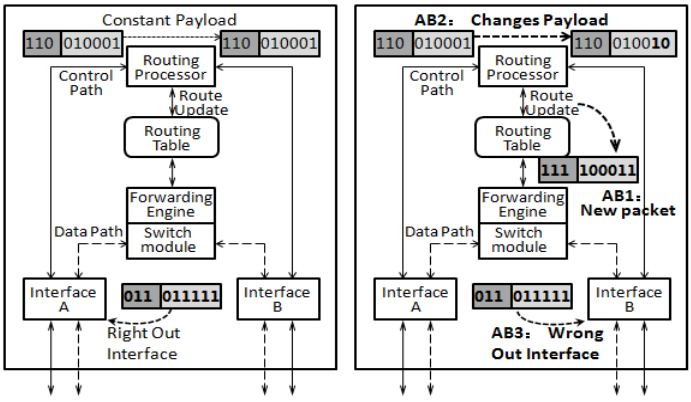

Fig. 1. Normal routing process (left) and Possible abnormal behaviors (right)

To this end, we propose a system that are achievable and applicable to all types of TCP/IP networks. Once the designed routers are deployed throughout a core network, the intercepting packets can be recognized and restrained efficiently, while all normal packets can pass through the system.

3.2 Static vs Dynamic Policy-Based System

There are generally two ways of designing systems for recognizing abnormal behaviors of routers. One is the static way such as code analyzing and operation result checking, while the other is the dynamic way such as function simulation and operation decode matching. Because this paper presents our first foray into a policy-based approach for securing routers, we concentrate on the static approach by proposing a set of static policies and the model of policy-based route. Instead of focusing on the routing and switching processes, the proposed set of policies concentrate on the changes in a 3-tuple characteristic code <Source IP, Destination IP, Payload> before and after it passes through the policy-based router. In other words, the policies are designed for recognizing and restraining the output packets whose characteristic codes are abnormal as shown in figure 1. As we show later in the paper, such a conception does not only make the policies universally applicable to all types of TCP/IP network, but also makes it easier to prove the security completeness, and to design high-performance policy-based routers.

3.3 The Policy-Based Security System for Routing and Switching in TCP/IP Networks

From the perspective of routing and switching operation results, the abnormal behaviors shown in figure 1 appear as the three kinds of patterns indicated in figure 2. The thick full curve Alice-Bob in figure 2 presents the routing path from Alice to Bob without any attack. However, once there is a vulnerable router R within the path, an attacker, Eve, can launch three kinds of attacks. Corresponding to abnormal behavior AB1 in figure 1, while routing packets normally, a vulnerable router R secretly reserves packets Alice sends to Bob, and redirects their payloads to Eve, which leads to abnormal result AR1. For AB2, R changes or substitutes the payloads of packets, which leads to different payloads between what Alice sends and what Bob receives as AR2 shows. For AB3, R alters the output interface by interfering the normal routing and switching process, as shown in figure 2, the output link of a packet is altered away from the thick full curve to the wrong link along AR3.

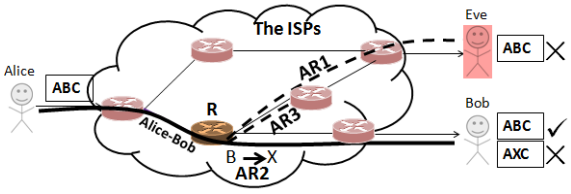

Fig. 2. The three patterns of abnormal behaviors of vulnerable router

Although being able to recognize the three kinds of abnormal behaviors, these three policies suffer from difficulties in implementation, universality and effectiveness. Notice that attackers usually intercept communication by “stealing” the payloads of the reserved packets, instead of “robbing” the original packets, these policies can then be optimized to three final policies below.

FP1: For a packet encapsulated payload, when the source IP address is not that of the router, if there are no input packets characterized the same 3-tuple code <Source IP, Destination IP, payload> with it, the packet is not allowed to output.

FP2: A router is not allowed to send packets generated by itself to any unauthorized end host.

FP3: When a packet is forwarded to an end host by a boundary router, the IP address of the host must match the destination IP address of the packet.

4 The Security Completeness of the Policy-Based System

For ease of reference, Table 1 defines the variables used for proving the security completeness of the three final policies to DIAs. In addition, an over bar means the opposite, for example, $\overline{R_{LE}}$ means a router which is not linked to Eve.

Table 1. Variable Definition

Variable	Definition
Alice	The sending user and host
Bob	The receiving user and host
Eve	The attacker and the attacking host
$\mathbf{R_{ARB}}$	An arbitrary router
$\mathbf{R_{LE}}$	A boundary router linked to Eve
$\mathbf{Pack_{A-B}}$	A packet Alice sends to Bob
$\mathbf{Payload_{A-B}}$	The payload of $\mathrm{Pack_{A-B}}$

As listed by ISO/IEC [14], there are generally three properties of information system security: confidentiality, integrity and availability (CIA). As for preventing data interception, users and ISPs hope all packets can and only be accurately and completely received by the destination hosts, therefore security completeness here means that the set of policies we proposed is necessary and sufficient to ensure packets the properties of confidentiality and integrity.

Lemma 1. Once all routers in a core network abide by the proposed routing and switching policies, when a device (Dev) receives a packet characterized by <Sou, Des, $\mathrm{Payload_{Sou-Des}}$>, then,

$$\begin{cases} \mathbf{1}\text{: It is Sou who sent } \mathrm{Pack_{Sou-Des}} \\ \mathbf{2}\text{: } \mathrm{Payload_{Sou-Des}} \text{ is the original payload Sou sent} \end{cases}$$

Proof. We prove 1 by contradiction. Without loss of generality, let $\mathrm{Path_{Sou-Dev}}$= [Sou, $\mathrm{R_1}$,…, $\mathrm{R_i}$,…, $\mathrm{R_n}$, Dev] be an arbitrary reachable path from Sou to Dev. If $\mathrm{Pack_{Sou-Des}}$ is not sent by Sou, then there must be another device $\mathrm{Sou}' \neq \mathrm{Sou}$ in figure 3, such that Sou' can send $\mathrm{Pack_{Sou-Des}}$ to Des through $\mathrm{Path_{Sou'-Dev}}$. When

Sou' is an end host, and can send $Pack_{Sou-Des}$, this means that Sou' had known $Payload_{Sou-Des}$ before it sent $Pack_{Sou-Des}$. Therefore it can send $Pack_{Sou'-Des-Payload_{Sou-Des}}$, instead of sending $Pack_{Sou-Des}$ in the pattern of IP spoofing. As a result, Sou' is a router. Because $Sou' \neq Sou$, according to FP1, when router Sou' sends $Pack_{Sou-Des}$, , there must be an input packet of Sou' characterized by <Sou, Des, $Payload_{Sou-Des}$>, this means that Sou' is not the sender of $Pack_{Sou-Des}$, which contradicts the hypothesis, then Sou is the only sender of $Pack_{Sou-Des}$.

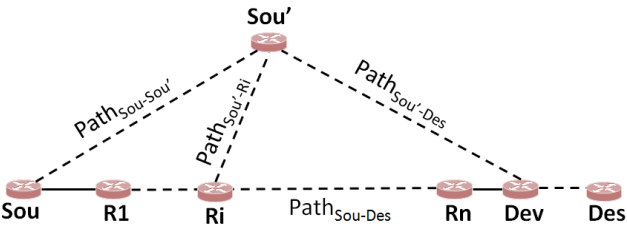

Fig. 3. Sou' sends $Pack_{Sou-Des}$ through Dev

Then we prove 2: For an arbitrary $Path_{Sou-Dev}$ in figure 3, we define $R_0 = Sou$; $R_{n+1} = Dev$. For each packet output from R_i ($1 \leq i \leq n$) according to FP1, there must be a matching input packet characterized by the same 3-tuple code with it. If the input packet is forwarded by R_{i-1}, for the atrbitrariness of i, each device in $Path_{Sou-Dev}$ shares the same 3-tuple characteristic code for $Pack_{Sou-Dev}$. As a result, $Payload_{Sou-Des}$ is the original payload Sou sent. On the contrary, if the matching packet is forwarded directly by $\overline{R_{i-1}}$, without loss of generality, we assume the matching packet is output by Sou' in figure 3, because $Pack_{Sou-Dev}$ can reach Sou' and R_i successively, according to result 1, there must be a path from Sou to Sou' , and a path from Sou' to R_i, such that $Pack_{Sou-Dev}$ is generated by Sou, and sent to Dev along a new path $Path'_{Sou-Dev}$ composed of $Path_{Sou-Sou\prime}$, $Path_{Sou\prime-R_i}$, and $Path_{R_i-Dev}$. Similarly, each device in $Path'_{Sou-Dev}$ shares the same 3-tuple characteristic code for $Pack_{Sou-Dev}$, therefore $PayLoad_{Sou-Des}$ is the original payload Sou sent.

Therefore, lemma 1 is right. Based on lemma 1, we prove the integrity and confidentiality of the policy-based system to DIAs.

Proposition 1. Once all routers in a core network abide by the set of policies, if the network can ensure reachability for each packet, *what is sent, is what is received*.

Proof. In lemma 1, particularly when Dev is Des, each packet $Pack_{Sou-Des}$ Des received was sent by Sou, and the sent payload matched the received payload. Because the network can ensure reachability for each packet, the communication Des receives is equal to what Sou sends.

Proposition 2. Once all routers in a core network abide by the set of policies, packets sent from a user can only be received by the target user.

Proof. We prove proposition 2 by contradiction. In a network shown in figure 4, if a non-target user Eve can receive $Payload_{A-B}$, then there must be a router R_{ARB} which

reserves $Pack_{A-B}$ and redirects $Payload_{A-B}$ to Eve. With AR1, R_{ARB} reserves $Payload_{A-B}$ and generates a new packet, whose destination IP address is Eve. With AR3, R_{ARB} forwards $Payload_{A-B}$ to Eve by altering the output interface of the reserved packet. Because the step of forwarding is limited to one link, AR3 can only be carried out when R_{ARB} is a boundary router linked to Eve, such as R_{LE} in figure 4.

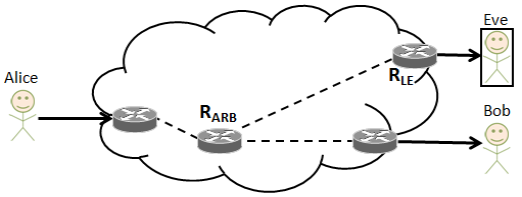

Fig. 4. R_{ARB} reserves $Pack_{A-B}$ and redirects $Payload_{A-B}$ to Eve

For an input packet characterized by <Alice, Bob, $Payload_{A-B}$>, the new packet generated by R_{ARB} is characterized by <*, Eve, $Payload_{A-B}$>, where * is the IP address of an arbitrary device for the convenience of attack. When *=R_{ARB}, R_{ARB} is trying to send information to Eve, which violates FP2. Also when * = $\overline{R_{ARB}}$, according to FP1, there must be a packet characterized by <$\overline{R_{ARB}}$, Eve, $Payload_{A-B}$> input to R_{ARB}before. By lemma 1, the new generated packet must be sent by $\overline{R_{ARB}}$, which means that $\overline{R_{ARB}}$ is trying to send information to Eve, this violates FP2 too.

When R_{LE} tries to forward a reserved $Payload_{A-B}$ to Eve as AB3 does, the characteristic code of the reserved $Pack_{A-B}$ is <*, $\overline{Eve}$, $Payload_{A-B}$>, which means that R_{LE} is trying to forward a packet to a mismatching end host. This violates FP3.

In summary, due to the set of policies we proposed, attackers cannot receive non target packets, which proves proposition 2.

FP2 and FP3 are obviously necessary for preventing DIAs. As for FP1, each tuple in the characteristic code is necessary too. If there is no source IP in the characteristic code, an assistant Carol can help Eve to perform data interception by sending Eve packets with different payloads. Once one of these payload happens to be equal to what Alice sends to Bob, a vulnerable router can send $Pack_{*-Eve-Payload_{A-B}}$ to Eve. If there is no destination IP in the characteristic code, R can send $Packet_{Alice-Eve-Payload_{A-B}}$ without violating the set of policies. If there is no payload in the characteristic code, Eve can ask for Carol to send packets to him. Once one of these packets pass through R, the latter can send $Packet_{Carol-Eve-Payload_{A-B}}$legally. As a result, we get the conclusion below.

Proposition 3. Each policy in the policy-based system is necessary to prevent DIAs.

The three propositions above show that the proposed system is security complete to DIAs. It is worth noting that network topology and routing protocols are not involved during the course of the proofs, which means that the policies are transparent to these aspects.

5 The Policy-Based Router Model and the Policy Violations Detecting Model

5.1 The Procedure of Policy-Based Abnormally Detecting

As shown in figure 5, the procedure of policy-based detecting is composed of two sub procedures, one collects the 3-tuple characteristic code <Source IP, Destination IP, payload> of every input packet to the database, the other detects the consistence of all output packets with the three policies one by one.

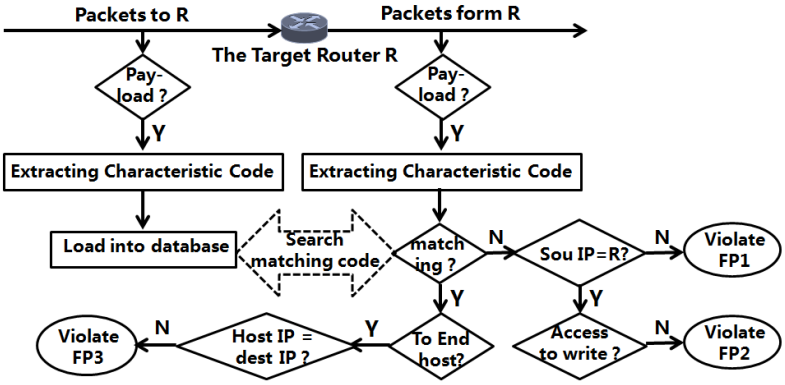

Fig. 5. The procedure for detecting the consistence of routers' behavior with the policies

5.2 The Policy-Based Router Model and the Policy-Violations Detecting Model

The policy-based router model and the policy-violations detecting model are shown in figure 6. The area inside the darker rectangle is the model of the policy-based router, and the outside part is the policy-violations detecting model.

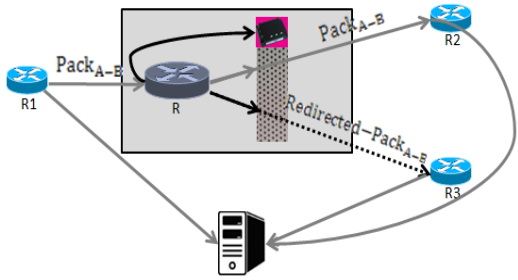

Fig. 6. The policy-based router model and the policy-based violations detecting model

Generally, the proposed policy-based router is made up of a traditional router R and a policy-violations detecting system. The latter is composed of an information collecting module and an output violations detecting module. The detecting system

does not take part in the process of the traditional router. When a packet is input to R, the information collecting module collects and sends its characteristic code to the output violations detecting module. The detecting model is composed of a chip designed according to the policy-based detecting procedure, and a random access memory for storing the database of all input characteristic codes. For each output packet, the detecting module picks up its characteristic code and seeks the matching code in the database, then judges whether R is trying to output an abnormal packet. In addition, the detecting module forbids violations departing.

The original $\mathrm{Pack}_{\mathrm{A-B}}$ can pass through a policy-based router in figure 6 legally, while once R tries to reserve $\mathrm{Pack}_{\mathrm{A-B}}$ and redirect $\mathrm{Payload}_{\mathrm{A-B}}$ to Eve, the $\mathrm{redirected\_Pack}_{\mathrm{A-B}}$ is judged as violation by the policy-based system

Because providers may not be able to (or unwilling to) design devices strictly under the policy-based router model, it is necessary to design the policy-based violations detecting model for ISPs to check whether providers abide by their promise. The area outside the dark rectangle in figure 6 is the model of the policy-violations detecting system, which is also composed of an information collecting module and an output violations detecting module. The collecting module is composed of all neighbor routers of the target router, while the detecting module is a server with the function of policies detecting. All neighbor routers of R are bypassed to the server. All neighbors send all packets input to R from bypass links to the server, and the server checks whether each output packet voilates the policies. Since servers are good at storing, if only the abnormal output packets are recorded to the server, we can be aware of what the attacker wants to intercept and where the position of the attack host locates.

5.3 A Fast Matching Algorithm

As expected, the policy-based router should be efficient. However, additional overhead in storing, computing and communicating decreases policies cheacking speed, particularly that in input and output codes matching As a result, it is necessary to design rapid matching algorithms. In what follows, we propose one.

In the algorithm, the characteristic codes database is substituted by a digest table, which record the fact that a packet with a certain hash digest has input into R. The size of the table is 2^{n} (where n is limited by the size of RAM, for example 2^{30}) bits, each position in the table is initialized to 0. As shown in figure 7, when a packet is input into R, the detecting system computes the digest of its characteristic code using a hash algorithm, and preserves a fixed part and length of the digest as I-digest-result. The value of the position corresponding to I-digest-result in the digest table is set to 1. For an output packet, the system computes and acquires its O-digest-result with the same hash algorithm, and look-up the value of the corresponding position in the digest table. If the value is 1, the algorithm asserts that there is an input packet matching the output packet. Otherwise, it confirms that there is no matching input packet.

It can be noted that the weight of the digest table will continue to increase as more packets are input into R, such that it is probable for an intercepting packet to have a 1 O-digest-result by coincidence. Therefore we propose 2 digest tables to record I-digest-results. Table 1 in figure 7 starts at the beginning of each $(2i) \times \rho$, while table 2 starts at the beginning of each $(2i + 1) \times \rho$, where $i = 0, 1, 2\ldots$, ρ is a time

period no less than the maximum time a packet is configured to stay in a router. With the 2 tables, the record of each input packet will be preserved at any time point, which ensures that each genuine packet is not categorized as an intercepting one.

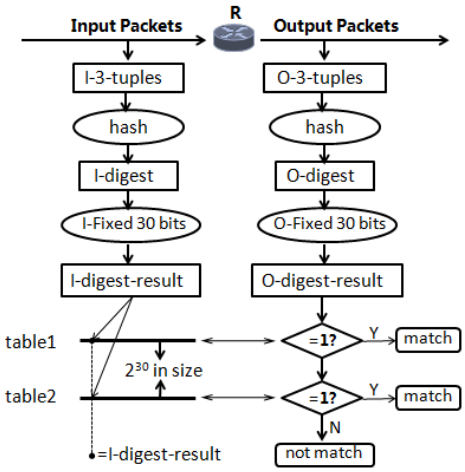

Fig. 7. A fast matching algorithm

When the length of digest result is 30 bits, the memory size of the detecting system is $2 * 2^{30}$= 2G bits. It is practical for the detecting system to assemble a 2G RAM.

6 Evaluation of the Functionality and Performance

6.1 Functionality

Focusing only on the operation result of the output packets, FP2 and FP3 can always be carried out correctly. As for FP1, we have to check whether there is an input packet matching it. We preserve part of the hash digest as the input record to improve the implementability of the system, which may lead to a false positive (a normal packet is confirmed as an intercepting one) and a false negative (an intercepting packet is confirmed as a normal one) result. We analyze the two false probabilities theoretically and by simulation.

For each normal packet, let IT be its input time, OT be the output time, there must be a i = 0, 1,…, such that $IT \in [\rho \times i, \rho \times (i+1))$, then $OT \in (IT, IT + \rho]$. As a result, for a packet with input time IT, one or both of the digest tables must preserve its output record, then the probability of false positive is 0.

The throughput of a router, ρ, and the size of the digest table (dtsize) determine the probability of a false negative. When throughput = 10Gbps, ρ=0.5s, dtsize=1G bits, the maximum number of packets input to R within 2ρ is

$$(\text{throughput}/\max\text{packet length}) \times 2\rho = {}^{10G}/_{1500 \times 8} \times 2 \times 0.5 = 833333$$

The process of input packets filling in the digest tables obeys a (0, 1) distribution, where the probability of the value of a position is replaced by 1 is $1/\text{dtsize} = 1/2^{30}$, then the expectation of the maximum total weight of the 2 tables (duplicate removal) in every 2ρ is

$$E = \text{throughput} \times (1 - (1 - 1/\text{dtsize})^{\text{maxnum}} = 833010$$

Therefore, the maximum expected probability of a false negative is 833010/1G = 0.07758%, which means that the expected maximum number of intercepting packets passing through the policy-based system is 833010×0.07758% = 647.

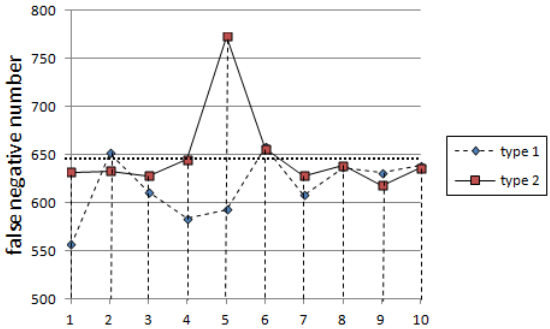

Fig. 8. Number of intercepting packets pass through the policy-based system by coincidence

We conduct the simulation with an X86 system. 1,250,450 packets are obtained from a real network. We use MD-5 as the hash algorithm, and preserve the value from the 65th to the 94th bit of the digest as the digest result. The simulation result shows that all the first 833,333 packets pass through the policy-based system. Based on the simulation system, we carry out two types of attacks. Type 1 changes the destination IP of the 833,333 packets to some random address, while type 2 reverses the payload of each packet together with changing destination IP. We carry out each type of attack 10 times. Figure 8 indicates the active numbers of the attacking packets which pass through the system. The average number of false negatives is 633, which is actually smaller than the expected 647. We calculate the weight of the 2 digest result tables (duplicate removal) after the first 833,333 packets finished passing through R, and get a result of 818,773, based on which the average number of false negative is 635. As a result, the simulation result matches the theoretical expectation.

In summary, we draw a theoretical conclusion that 100% normal packets can pass through the policy-based system, and that no less than 99.92242% intercepting packets will be recognized.

6.2 Performance

The performance of a router is inversely proportional to the average time T it takes to process a packet. For our policy-based router, T is composed of the traditional routing

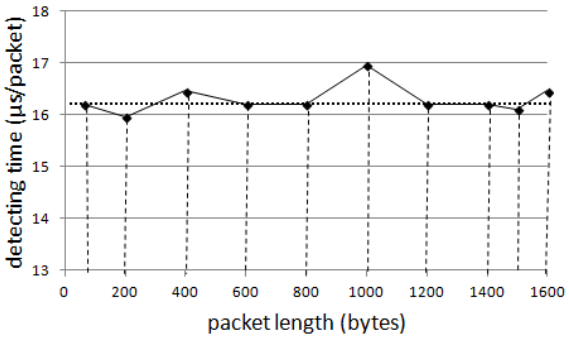

Fig. 9. Average detecting time for packets of different lengths

time T1 and the time T2 used for detecting policy violations. For a certain traditional router R and a fixed packet length, T1 is a constant. Therefore the performance of the proposed policy-based router is depended on T2.

Using the simulation system described above, we determine T2 with traffic from real network. The amount of the traffic is 4,897,900 packets, among which there are 4,010,903 packets whose length is 1,500 bytes, accounting for 81.9% of the total. Figure 9 indicates the test values of T2 for the simulated system. Because T2 might be influenced by the length of a packet, we conduct 10 experiments with different packet lengths. We choose 64, 200, 400, 600, 800, 1000, 1200, 1400, 1500 and all packets with difference lengths as packet length.

The time unit in figure 9 is µs. From this figure we conclude that when lower than 1500 bytes, packet length has little impact on average T2. In fact, the average value of T2 is 16.30µs, and the maximum policy-based detecting rate varies from 31.4M to 736.2M bps according to the average packets length, which is acceptable for edge core networks. This is because the X86 system for simulation is assembled a dual core processer of Intel i5-2410 and a 4 G RAM, detecting system with hardware no lower than the simulation system can be developed. Moreover, current routing devices in convergence layer are usually assembled by dedicated chip, also routing and forwarding with hardware platform, instead of routing and forwarding with software platform composed of general processor as the experiment does. If only it is meaningful for the whole core network deploying policy-based routers, as Janaka et al. [15] shows, it is reasonable to construct MD5 accelerators using hardware and FPGA, by which the performance of policy-based routers can be greatly promoted.

7 Conclusion

In this paper we designed, implemented and evaluated a policy-based routing and switching system aimed at stopping DIAs. Our proposal includes a set of policies that can be embedded in routers so as to detect interception output packets. We have been able to prove the security completeness of the proposed set of policies to DIAs, and discussed the implementability of the proposed system, in which case we also proposed a fast matching algorithm. In addition, by simulation, we have shown that our

proposal is able to positively identify 100% of genuine traffic, and reject about 99.9% of violating traffic, and the performance of the proposed system is acceptable with regard to current TCP/IP networks.

However, there are some technical issues that we continue to study before we can consider our design as complete for commercial purposes. To prevent DIAs, all routers in the core network must abide by the proposed policies. In order to decrease deployment cost and running overhead, it would be interesting to consider an incremental deployment of the policy-based system. In addition, as a new attempt in this field, we proposed a static policy-based system in this paper. Future work should consider a dynamic approach which could be based on operation decode matching for higher efficiency and broader application fields.

Acknowledgment. We gratefully acknowledge funding support for this research from Chinese 863 Project *The key Technique and Verification for Address Driven Networking*.

References

1. National Security Agency: PRISM/US-984XN Overview, April 2013
2. Bowden, C.: The US national security agency (NSA) surveillance programmes (PRISM) and foreign intelligence surveillance Act(FISA) activities and their impact on EU citizens' fundamental rights (2013)
3. Dobrescu, M., Argyraki, K.: Software dataplane verification. In: NSDI (2014)
4. Kothari, N., Mahajan, R., Millstein, T., et al.: Finding protocol manipulation attacks. In: SIGCOMM (2011)
5. Trusted Computing Group: TCG Specification Architecture Overview [EB/OL]. [2005-03-01]. https://www.trustedcomputinggroup.org/groups/TCG_1_0_Architecture_Overview.pdf
6. Challener, D., Yoder, K., Catherman, R., et al.: A Practical Guide to Trusted Computing, 1st edn. IBM Press (2007)
7. Chen, P.M., Noble, B.D.: When virtual is better than real. In: HOTOS-VIII, May 2001
8. Garfinkel, T., Mendel, R.: When virtual is harder than real: security challenges in virtual machine based computing environments. In: Proc of the 10th Workshop on Hot Topics in Operating Systems, pp. 210–217. USENIX Association, Berkeley (2005)
9. Kim, T.H., Basescu, C., Jia, L., Lee, S.B., Hu, Y., Perrig, A.: Lightweight source authentication and path validation. In: SIGCOMM (2014)
10. Xue-hai, P., Lin. C.: Architecture of trustworthy networks. In: 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing (2006)
11. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol, IETF RFC 2401, November, 1998. http://tools.ietf.org/html/rfc2401
12. Meenakshi, S.P., Raghavan, S.V.: Impact of IPsec overhead on web application servers. In: International Conference Advanced Computing and Communications, ADCOM 2006, pp. 652–657 (2006)
13. Chao, H.J.: Next generation routers (invited paper). Proceedings of the IEEE **90**(9), 1518–1558 (2002)
14. ISO/IEC: International Standard ISO/IEC 27000, 3rd edn, January 15, 2014
15. Deepakumara, J, Heys, H.M, Venkatesan, R.: FPGA implementation of MD5 hahs algorithm. In: Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, CCECE 2001 (2001)