VOL. 11, NO. 21, NOVEMBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

FPGA BASED HONEYPOT WITH STATEFUL TCP EMULATION FOR SMTP MALWARE COLLECTION

Sruthi Rajan and Harish Ram D. S.
Department of Electronics and Communication Engineering Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, Amrita University, India
E-Mail:sruthi.r.nambiar@gmail.com

ABSTRACT

Network Security issues and threats are getting more and more matured and hence the countermeasures have to be geared up to address the ever-growing number of potential attacks. Honeypots are widely employed to detection and collection of malware. Conventional software honeypots are not capable of handling the gigabit level traffic in today's network server backbones. This has evoked interest in FPGA based Honeypot for Malware Collection which can cope with speed and security challenges. Honeypots are network security devices designed to emulate real machines and work by fooling the attacker into believing that it is a legitimate system with vulnerabilities. The main issue when designing a hardware honeypot is the implementation of Transmission Control Protocol. Conventional TCP approach is not easy because being stateful protocol, it consumes much hardware resources. Hence the current work in this area is limited to the implementation of UDP as a transport layer protocol and no works have attempted stateful TCP in hardware. Some works consists of stateless TCP by compromising reliability. Here in this work, an emulation of TCP based SMTP server is implemented in which both stateful and stateless implementations of TCP are implemented in transport layer separately and a comparison of resource utilization is done.

Keywords: VEH, SMTP, stateful TCP, stateless TCP, IP, ARP.

1. INTRODUCTION

Honeypot is a network security system which emulates the behavior of an actual system. Whenever an attacker tries to attack the system it makes the attacker to believe that it is a legitimate system with great vulnerabilities. Hence the attacker exposes all its malicious behavior and the resulting information can be collected by the honeypot.

Most of the malware identification tools follow signature based approach [9]. In this approach, when a packet is found to be malicious, a pattern to uniquely identify that packet is generated and is known as the signature. These signatures are stored in a database and when new packets are received, it is matched with the existing entries in the database to reach in a conclusion whether the packet is a malware or not. The main issue in this approach is the failure in identifying new threats which are not entered in the database. So, new threats can easily pass through such a malware identification system.

Firewalls are security systems which filter the traffic between the private and public network based on some behavioral rules. The first generation firewalls were just packet filters, later they were designed in application layer and now they are capable enough to filter data at different layers. But firewalls cannot detect an attack from a private network. The system fails to detect an attack if the attacker intrudes into the private network and initiate the malicious action. This is a serious issue in corporate and campus networks.

There are software honeypots existing (e.g. HoneyD). They are not capable enough to operate with the multi Gbps speed of today's networks. Also there is a great chance that the processors which are software programmable themselves can be turned into an attacking source by the intelligent malware[5]. Also, if the malware is intelligent enough to detect the presence of a malware detection program in the system, it can hide its malicious behavior and can remain undetected. These loopholes that are present in software solutions made researchers to think about hardware honeypot in FPGA platform. As far as the incoming traffic has nothing to do with the configuration port in FPGA, there is no chance to get the functionality altered by the traffic. Hence the implementation in FPGA platform ensures high throughput, reliability and re-configurability.

The stateless TCP approach [1] which has been already implemented to save hardware resources has to pay in terms of reliability which is the basic purpose of TCP. Here we emulate a stateful TCP based SMTP server and compare the resource utilization with stateless TCP based SMTP server.

In this paper, we first give a brief introduction of the network security system structure with honeypot in section II. Section III discusses the Netstage architecture which we followed in our designs. Section IV gives an overview of how a stateless TCP implementation works. The approach we used for stateful TCP based SMTP is presented in Section V. Section VIdiscusses the state diagram realization of TCP and SMTP layer functions. Section VIIis devoted to the analysis of the results of the different TCP implementations their comparison.

2. HONEYPOT SYSTEM

Honeypots are placed at an exposed part of the network to attract attackers. See fig.1.The DMZ or demilitarized zone is a sub network to provide additional security to the internal network of an organization. It contains the organization's services which are exposed to the untrusted network. In fig.1, the honeypot is contained in the DMZ.

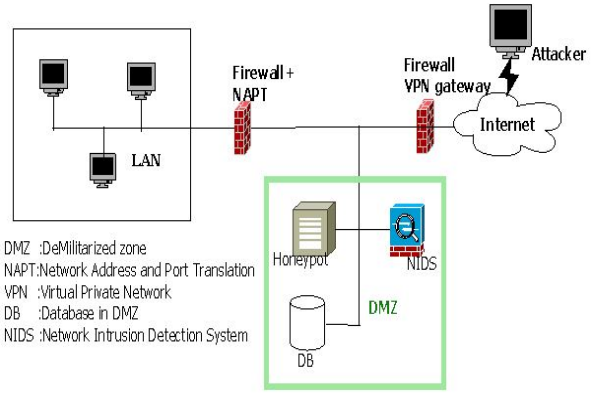

Figure-1. Network security system with honey pot.

Those sessions which are found to be suspicious by the firewall are directed to the honey pot in DMZ and those which are found to be legitimate are passed to the internal network or to the next level of security checking. Now, the work of honey pot is to collect the information on the suspicious session. The intruder has the feel of accessing the actual service in the network and it tries to expose all its malicious actions which can be monitored and recorded by the honey pot without the knowledge of the intruder.

3. HARDWARE HONEYPOT ARCHITECTURE

We follow net stage architecture which includes network core implementation modules, Vulnerability Emulation Handlers and management station [5].The core system consists of modules for handling different protocols corresponding to each communication layer.

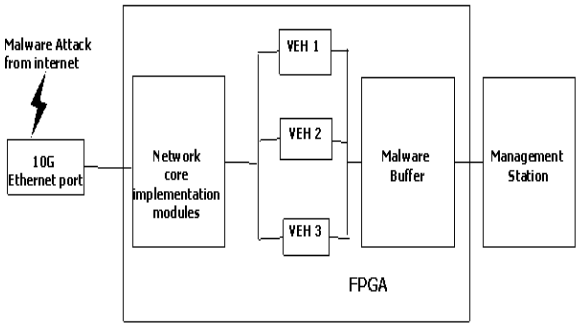

Figure-2. Hardware honey pot architecture [3].

Our design supports ARP, IP and TCP protocols which correspond to the layers in hierarchical order. It is connected to different vulnerability emulation handlers which emulate security flaws in different applications. As the packet proceeds through each layer, the corresponding headers are taken off and processed. An internal header follows the packet all its way to uniquely identify the packet as it reaches each layer. The modules are defined by state machines. Different states and state transitions make the client-server communication using TCP/IP stack proper without giving any hint to the attacker that it is not the actual system.

Buffers are used for interstage decoupling and also they can serve the purpose of storage in case of data rate variations between different stages. Different acceleration techniques like parallel processing can be added in any stage for better performance [3].

The Vulnerability Emulation Handler (VEH) is the heart of a hardware honey pot system which is responsible for the processing of data in network layer 7. It emulates vulnerabilities in specific applications. The preceding stage may have mapping tables that is used to direct packets to specific VEH according to the specific application (according to port numbers which specifies the application or service) .Since we are emulating SMTP server only, a check for port 25 which corresponds to SMTP is carried out at the transport layer and all other service requests are discarded.

The VEH does the main function of a honey pot which is malware collection. The VEH only responds to incoming messages and never initiate a transaction [5]. String matching is used for checking incoming packet fields and to initiate particular state transitions and also to identify malware presence. The collected data can be forwarded to the management station for analysis or any further action.

4. STATELESS TCP IMPLEMENTATION IN HARDWARE HONEYPOT

The requirement of storing connection state information in TCP makes our implementation memory hungry when dealing with hundreds of concurrent connections with 10+ Gigabits/s speed. Hence to save hardware resources we go for the idea of stateless TCP [1]. In this approach, we completely rely on the incoming packet header information and extract the state information (sequence and acknowledgement numbers) from the transport layer header of the received packet. Hence we are not storing sequence or acknowledgement number related information for a connection in memory.

In normal connections if an acknowledgement is not received for a packet, the packet is retransmitted. Avoiding packet retransmission to save hardware resources in stateless approach can be justified by relying on the fact that only 10 % of the TCP connections suffer packet loss which transmits less than 50 kB of data on the internet.

Here, we receive the packet and as it reaches the transport layer the connection state indicating information like connection status flags (SYN, FIN and ACK), data length, sequence number and acknowledgement numbers are extracted from its header.

The sequence and acknowledgement numbers in the reply header are determined from this collected information from incoming packet instead of going for saving and retrieving previous communication status.

The demerits when going for stateless approach are:

a) Out-of-order reception of packets from a packet group cannot be detected.

b) Lost packets cannot be detected.

c) Unnecessarily retransmitted packets cannot be detected.

5. STATEFUL TCP APPROACH FOR HARDWARE HONEYPOT

In normal TCP based communication, after completing the handshaking procedures the server allocates a Transmission Control Block (TCB) for the client to store the state related information. This is done to keep on tracking the communication by maintaining correct sequence and acknowledgement status.

TCP transmission can be conceptually divided into following categories:

a) Sent and acknowledged

b) Sent, but not acknowledged

c) Not sent, but the remote system is ready to accept

d) Not sent, and the remote system is not ready to accept also

The first category need not be considered because they are already received by the remote system. We track the states of other category bytes by using the following pointers. They are SND_UNACK, SND_WND and SND_NXT which corresponds to the first sequence number in the list which is sent but not yet acknowledged, send window which is the number of bytes the receiver is ready to accept and the sequence number of the byte which has to be sent next respectively.

The receiving bytes are handled using the pointers RCV_WND and RCV_NXT. They indicate the number of bytes our system is ready to accept more at this point of time and next expected sequence number from the sender respectively.

As we receive each packet, we carry out a check to ensure that the packet sequence number lies in the valid receive window. If it is not, then the packet is simply discarded. Thus we avoid treating the packets which are unnecessarily retransmitted. We also maintain a retransmission queue where all the sent bytes are stored. A counter is maintained for these bytes and a particular count is set when the data is filled in retransmission queue. If transport layer receives acknowledgement for the byte before the corresponding counter reaches zero, that byte is erased from queue. If not, the byte is retransmitted.

TCB contains remote and local socket numbers and all the send and receive pointers discussed earlier. From these details and incoming header information, we check the validity and reliability of incoming packet and generate the reply header. Certain details like total data length from IP header are also used at the transport layer which are transmitted to transport layer as an internal header.

6. STATE MACHINE REALIZATION OF NETWORK PROTOCOLS

The SMTP mail transfer sequence is shown in Figure-3. The remote machine can interact with this SMTP machine only if the handshaking is completed at the TCP level. Transport layer will notify SMTP VEH the completion of handshaking by passing a dummy data to it. Then SMTP sends its code to notify the remote machine its readiness to accept data. The end of data is indicated by a '.' in one line from the client. And by the QUIT message it stops communicating with VEH.

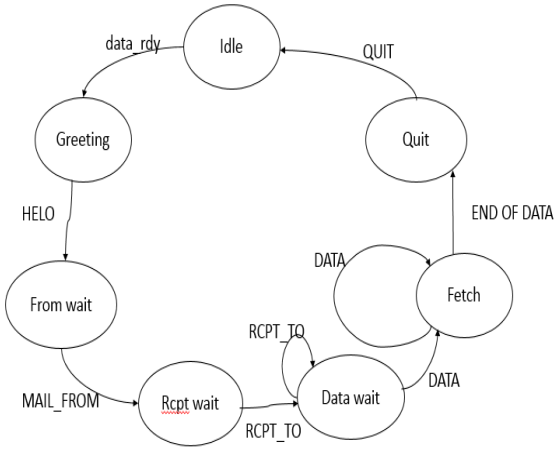

Figure-3. State machine for SMTP VEH.

Now the job of transport layer is to go for connection termination handshaking and once it is completed, the communication gets disconnected and all the records in TCB and retransmission queue gets erased. The state machine realization of TCP is as shown in Figure-4.

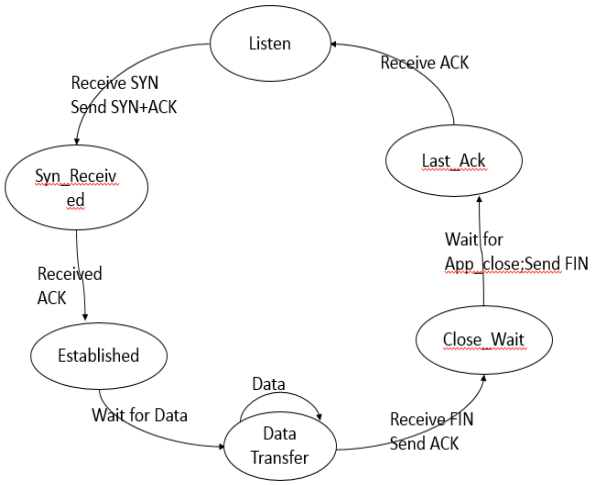

Figure-4. State machine for TCP.

7. RESULTS AND DISCUSSIONS

The designs were synthesized using Vivado 2014.4 with Virtex-7 VC709 evaluation platform (xc7vx690tffg1761-2) as the target device at a clock frequency of 200 MHz. The overall latency is found to be 23 cycles. Resource utilization comparisons are as shown in Table-1.

Table-1. Resource utilization comparison between stateful and stateless TCP approaches.

Site type	Utilization in stateless design	Utilization in stateful design	% increase in utilization from stateless to stateful design
Slice LUTs	2485	11451	360.8
Slice Registers	3680	7388	100.76
BRAM	38	56	47.37
F7 Muxes	0	32	-

The increase in resource utilization is three-fold for the stateful approach. This is expected since there is an appreciable increase in storage requirements due to the need to maintain additional table information pertaining to the sessions. However, the increase is justified since the stateful approach provides a complete emulation without loss of session information and retransmission.

8. CONCLUSIONS

Hardware Honey pots have tremendous advantages in terms of speed, reliability and security. Here we implemented a TCP based e-mail hardware honey pot which emulates an SMTP server. The stateless TCP approach in hardware honey pot which has been already implemented in previous works aims to save hardware resources. But it suffers from the inability to handle unnecessarily retransmitted packets and out of order packets. Hence a stateful TCP approach for hardware e-mail honey pot is proposed and a comparison is made with the stateless approach. Future work will include incorporating other packet types in the TCP emulator and reducing the area overheads. Also the power consumption of the designs will be analyzed.

REFERENCES

[1] Muhlbach Sascha and Andreas Koch. 2010. An fpga-based scalable platform for high-speed malware collection in large ip networks. Field-Programmable Technology (FPT), 2010 International Conference on. IEEE.

[2] MühlbachSascha and Andreas Koch. 2012. A dynamically reconfigured multi-fpga network platform for high-speed malware collection. International Journal of Reconfigurable Computing. 2012: 4.

[3] Muhlbach, Sascha, and Andreas Koch. 2011. A novel network platform for secure and efficient malware collection based on reconfigurable hardware logic.Internet Security (WorldCIS), 2011 World Congress on. IEEE.

[4] Muhlbach, Sascha and Andreas Koch. 2010. A dynamically reconfigured network platform for high-speed malware collection. Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on. IEEE.

[5] MuhlbachSascha, *et al*. 2010.Malcobox: Designing a 10 gb/s malware collection honey pot using reconfigurable technology. Field Programmable Logic and Applications (FPL), 2010 International Conference on. IEEE.

[6] Pejovic, Vukasin, *et al.* 2007. Migrating a Honeypot to Hardware. Emerging Security Information, Systems, and Technologies, 2007. SecureWare 2007. The International Conference on. IEEE.

[7] MairhAbhishek, *et al*. 2011. Honeypot in network security: a survey. Proceedings of the 2011 International Conference on Communication, Computing and Security. ACM.

[8] MokubeIyatiti and Michele Adams. 2007.Honey pots: concepts, approaches, and challenges. Proceedings of the 45th annual southeast regional conference. ACM.

[9] Egele Manuel, et al. 2012. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR) 44.2: 6.

[10] Muhlbach, Sascha, and Andreas Koch. 2014. A Reconfigurable Platform and Programming Tools for High-Level Network Applications Demonstrated as a Hardware Honeypot. Selected Areas in Communications. IEEE Journal on. 32.10: 1919-1932.

[11] Muhlbach Sascha and Andreas Koch. 2011.NetStage/DPR: a self-adaptable FPGA platform for application-level network security. Reconfigurable Computing: Architectures, Tools and Applications. Springer Berlin Heidelberg. pp. 328-339.

[12] Dhanesh. P, Harish Ram. D. S. 2015. A fast and scalable pattern matching scheme for NIDS using Z algorithm. International Journal of Applied Engineering Research. 10(16).

[13] Singh, AbhayNath and R. X. Joshi. 2011. A honey pot system for efficient capture and analysis of network attack traffic. Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on. IEEE, 2011.

[14] Sadasivam, Karthik, Banuprasad Samudrala and T. Andrew Yang. 2005. Design of network security projects using honeypots. Journal of Computing Sciences in Colleges. 20.4: 282-293.