A SECURITY FRAMEWORK FOR ETHERNET BASED EMBEDDED WEB SERVER

Ravi Kiran Varma.P[1]and V.Valli Kumari[2]

[1] MVGR College of Engineering,Vizianagaram, AP, INDIA.
ravikiranvarmap@gmail.com
[2]Andhra UniversityCollege of Engineering, Visakhapatnam, AP, INDIA.
vallikumari@gmail.com

ABSTRACT

The enormous growth of the internet and its foray into every corner of our life makes it an indispensable tool to work with. The integration of the ubiquitous internet with embedded devices brings about a plethora of applications. It is a better choice for these devices to be embedded a web server. The embedded web server technology is a combination of embedded device and internet technology. As there is something like internet involved, there are enough chances that the web server might be attacked. This paper proposes a 3 layer ACA security framework which provides Authentication, Confidentiality and Availability for a tamper proof secured access to the internet enabled embedded web server. We have demonstrated the capability of a low cost rabbit 3710 based embedded web server module by implementing SSL protocol stack and also the capability to detect DOS attacks on Embedded Web Server using synfindiff and finite state machine algorithm. We found that TCP finite state machine algorithm performed well with faster detection time that the synfindiff algorithm.

KEY WORDS

Embedded Web Server, Authentication, SSL, Intrusion Detection.

1. INTRODUCTION

Integration of internet with embedded devices opens up a whole new dimension of applications for embedded systems where we will be able to remotely access and use electrical and electronic devices from anywhere in the world. These involve a wide variety of confidential or personalized applications which when subjected to unauthorized access may lead to unacceptable informational and economic losses for the organization or individual. So there is a need to create a secured embedded web server.In order to create a secured embedded server we use the RABBIT 3710 Ethernet based microprocessor shown in figure 1. This provides an efficient, quick and economical method to create a server for the application.This paper describesthedesign and implementation of an Embedded Web Server to remotely access electrical appliancesor any input/output devices using Rabbit Core Module RCM 3710 microprocessor and Dynamic C platform, the main focus being the two tier security provided through secured login authentication facility of Dynamic C and Secured Socket Layer (SSL) implementation in Dynamic C. Figure.2 shows the topological location of the device EWS which means Embedded Web Server.In a typical networked environment the EWS is connected in the intranet or in the internet, hence there is a possibility of internal and external attacks. The attacks can be normally one of the following categories, namely Denial of Service attacks (DOS), Probe attacks, Remote to Local attacks and

DOI : 10.5121/ijesa.2012.2203

User to Root attacks. DOS attacks cause the target to be consumed with bogus or malicious requests from the attackers and thereby unavailable to the legitimate users. SYN flood is one example of DOS attack. Probe attacks try to gather information about the target which may be useful for further launch an active attack. Ping sweeps, Port scans etc fall into this category of attacks. Remote to Local (R2L) are a class of attacks where the attacker tries to attain local privileges on the target machine from a remote machine, brute force attack, password guessing etc. come under this category. User to Root (U2R) attacks are those in which the attackers tries to attain root privileges having user level access, using some escalation techniques.

The fact that majority of the attacks in the network enabled devices are DOS attacks, motivated us to take up this issue regarding Embedded Web Servers. Network enabled devices are vulnerable to attacks,which makes no exception for even embedded devices. There is a lot of need for electrical, electronics, consumer, industrial equipment be provided with internet connectivity for them to be enabled to control form remote location and access without the physical presence at their location. At the same time securing the internet enabled embedded devices with the available limited hardware and software on board is a major challenge and very important in today's malicious environment. The basic security mechanism is to provide authentication service, confidentiality and data integrity should also be taken care at the same time, SSL can provide the same. Further the availability of the device is also of prime importance. With the help of DOS attack simulation on the EWS device in our lab, we found that the device is flooded with malicious requests and very soon the connection queue is full and it was no more able to serve the real users. In this work we have demonstrated the capability of EWS device to protect itself from DOS attack by detecting SYN flood attacks. Rest of the paper is organized as follows: Section 2 consists of the related work, section 3 consists of discussion on basic authentication and session based encryption using SSL. Section 4 tells about DoS attack detection in EWS and section 5 gives the conclusion and results.

Figure 1: Rabbit RCM3710 Module.

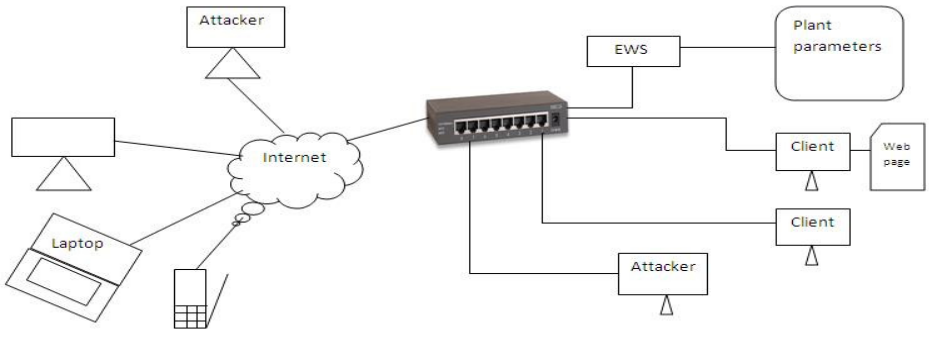

Figure 2: Topological representation of the EWS.

2. RELATED WORK

Networking of embedded systems has become has become more prevalent than stand-alone systems. Generally it is difficult for us to get information from remote equipment. From the very first we usually adopt RS232, RS485 or CAN, but these manners are too limited in distance nowadays. In a phenomenal work Tao Lin et al [1] proposed their "Webit" a EWS which gives a uniform Internet interface to traditional equipment. In another great paper by Guangjie Han et al. [2] showed that the embedded system can be utilized to serve the embedded web documents, including static and dynamic information about embedded device, to web browsers. A good work by Xiang Yang et al [3] *used AT91SAM9260* as the hostcontroller designed a Home Intelligent System which has several merits such as credible, flexible, easy maintenance, low cost and so on, and used Javalanguageto develop Embedded Web Server. There is a considerable study on development of remote connectivity for embedded applications but most of them did not mention about the security issues in protecting the server from unauthorized and malicious attackers. These can also be used to monitor the working of devices as said by Yong et al [11]. Apart from these, these can also be used to protect the devices from being tampered [12]. Few good works [13], [14], [15] shows how network can be embedded onto devices using the TCP/IP stack implementation. In another work [16] the authors demonstrated the implementation of digest authentication and symmetric cryptography methods to provide security for web-based access to embedded devices. This article provides the approach that we followed to implement security measures on the EWS.

3. AUTHENTICATION

Authentication is the processes of establishing the true identity of the person trying to access a web server. The identity is verified mostly by creating a mechanism to verify the user name and corresponding password.

3.1 Authentication using Dynamic C

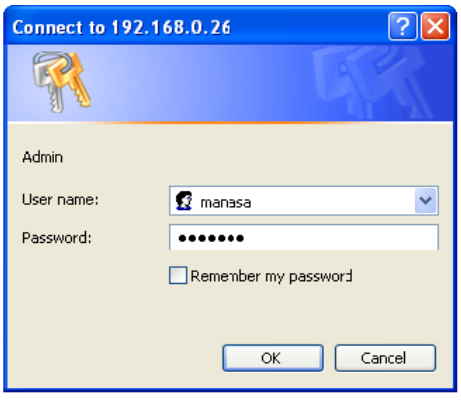

Figure 3: Basic Authentication.

The mechanism used is the predefined Dynamic C Functions which are invoked when the web page is accessed to modify the contents of the page. The user has to specify his user name and his password.The functions verify whether the user name and password are valid and then proceeds to the next page.

The following are the functions that are used:

1.sspec_addrule("/admin",Admin",admin,admin,SERVER_ANY,SERVER_AUTH_BASIC, NULL);
2.sauth_adduser("username","project",SERVER_ANY);
3. sauth_setusermask(userid, admin, NULL);

The Logic authentication window is shown in figure 3. Basic authentication does not provide confidentiality. Data protection and confidentiality are provided using the Secure Socket Layer

3.2 Secured Access / SSL

SSL is the ubiquitous security protocol used in almost 100% of secure Internet transactions. SSL is relatively new to the world of embedded systems because it was out of scope of the older days' embedded devices to handle it.However, starting with Rev. A of the Rabbit 3000 microprocessor, hardware assistance has been added to speed up some of the more complex SSL cryptography operations, making SSL a viable solution in a market where standard (usually complex) security protocols have not traditionally been supported.SSL is designed to run over TCP/IP. The following steps shows how the SSL protocol fits into the overall TCP/IP reference model.

3.2.1 Steps to Set Up and execute SSL on rabbit device

- Creating a digital certificate
- Importing the certificate
- Setting up TCP/IP for the sample application
- Setting up the application to use SSL
- Setting up the web browser
- Running the application

3.2.1.1 Creating a digital certificate

Using the Rabbit SSL certificate utility we can create a digital certificate for the SSL enable Rabbit web server. This process involves creating our own Certifying Authority (CA) along with its root CA, and then we need to create a server certificate signed by the root CA. The server certificate is created using the certificate creation wizard as shown in figure 4.

Figure 4.Certificate creation wizard.

3.2.1.2 Importing the Certificate.

Dynamic C *#import* is used to import the certificate into the project.

3.2.1.3 Setting up TCP/IP for the sample application.

TCP/IP protocol stack must be enabled for the device to use by configuring the file TCP_CONFIG.LIB library. The HTTPS port number 443 has to be enabled.

3.2.1.4 Set up the application to use SSL

There are several macros that have to be configured for HTTPS server namely USE_HTTP_SSL which is used to enable SSL for HTTP, HTTP_MAXSERVERS and TTP_SSL_SOCKETS are used to specify the number of HTTP and HTTPS servers.

This would be done by using the following code in program:

```
#define HTTP_MAXSERVERS 3 // Total number of  servers
#define USE_HTTP_SSL // Use SSL
// Tell HTTP.LIB Reserve 1 server for HTTPS out of 3 servers.
#define HTTP_SSL_SOCKETS 1
```

3.2.1.5 Set up the browser

This step involves setting up of TLS1.0 (optional) for more security and supports some browsers, installing our root CA certificate and using the *https://* prefix.

3.2.1.6 Running the application.

To run the application we need to type “https://<IP address OR URL of the embedded web server device>” in the browser.

3.2.2 SSL Performance

The SSL performance is based on cryptography and hashing. Rev. A Rabbit 3000 chips contain special instructions that speed up RSA operations significantly. Here each operation takes just more than 2 seconds for a 512bit key on a 44MHz Rabbit 3200. With RSA operation, the entire handshake takes approximately 2.5 to 3 seconds. The SSL implementation here uses RC4 stream cipher which is simple and fast for a cipher at 15KBps. RC4, being the smallest and fastest of all the cipher algorithms, is ideal for embedded processors. The packet format after implementing SSL is shown in figure 5 which is captured using Wireshark sniffer.

3.3 Establishing a Socket Connection

This is the basic step in the usage of an embedded web server. We use the Client-Server Paradigm and establish a Socket Connection between the embedded web server and the clients. A socket is an interface which can be both from the client side and the server side. The most important point here is that we have to choose the type of the socket connection we are going to establish. It can be either TCP socket or UDP socket.Applications need to make the *socket()* call first to create the socket. If the server is creating a socket, it must then use thebind()function to attach the socket to an IP address and port number.

If using TCP, [4] the server can then get into a listen state and listen for incoming connections. Otherwise, if using UDP, the server can block until it receives a UDP datagram. If using TCP, the client tries to connect to a server using *connect()* and can then *read()* and *write()* data to that socket. The client does not need to call *bind()*; it gets an arbitrary local port number, which for TCP clients is usually just fine. In case of UDP, the client can simply use *sendto()* and *recvfrom()* to talk to a UDP server.

In order to set up an IP Address to the device, we make some changes to the library code. The board by default has private Class A address of "10.10.6.100". Assuming we are going to make the device work in a class C private address space, we will define a static IP address of "192.168.0.26". So, we define a custom library and set up the TCPCONFIG macro accordingly.

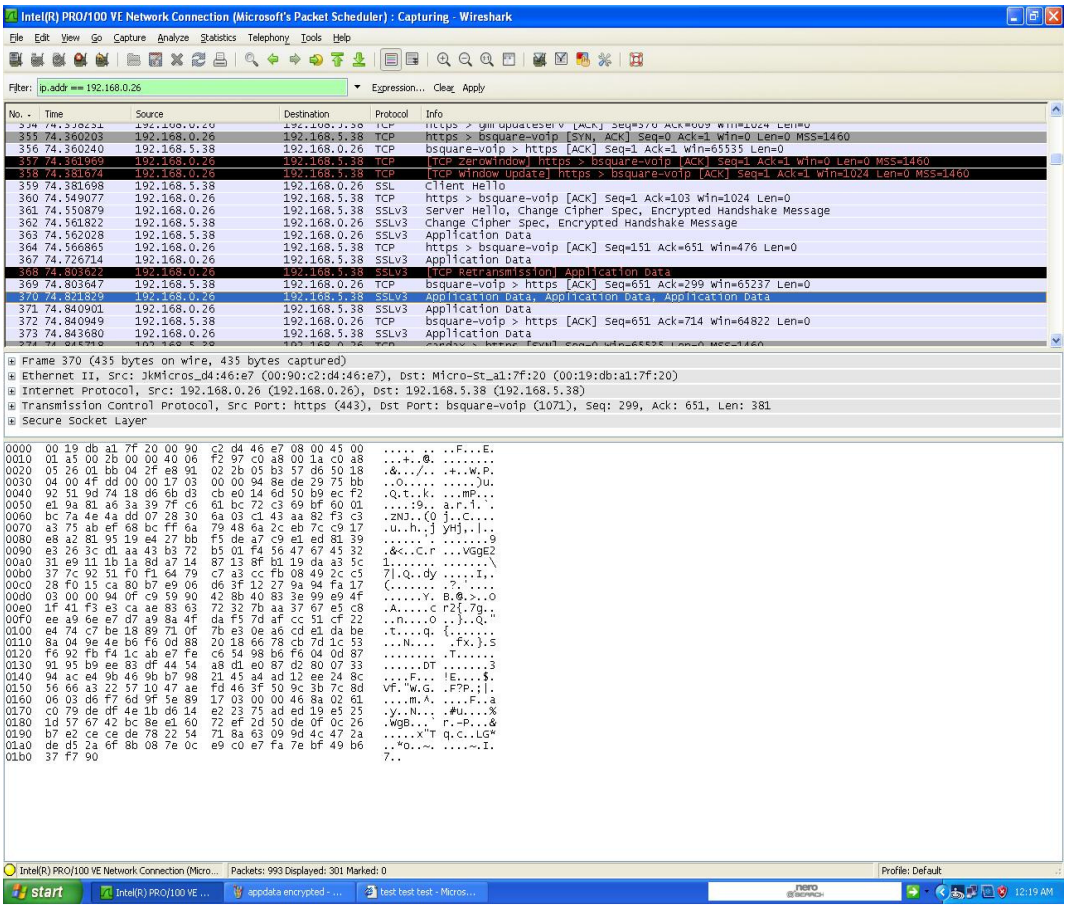

Figure 5: The Encrypted form of packets after implementing SSL captured in Wireshark sniffer.

We create a custom configuration library, CUSTOM_CONFIG.LIB, and will store it in the same folder as TCP_CONFIG.LIB. We have the following code in the CUSTOM_CONFIG.LIB file:

```
#define _PRIMARY_STATIC_IP "192.168.0.26"
#define _PRIMARY_NETMASK "255.255.255.0"
#ifndef MY_NAMESERVER
#define MY_NAMESERVER "192.168.5.12"
#endif
#ifndef MY_GATEWAY
```

```
#define MY_GATEWAY "0.0.0.0"
#endif
```

Next, since we are defining our own configuration, we will define a custom value for the TCPCONFIG macro. Since values above 100 are read from CUSTOM_CONFIG.LIB instead of TCP_CONFIG.LIB, we will use a value of 100; this is consistent with static IP configuration as shown earlier. Except for the line checking for the value of TCPCONFIG, everything else is just copied from TCP_CONFIG.LIB:

```
#if TCPCONFIG == 100
#define USE_ETHERNET 1
#define IFCONFIG_ETH0 \
IFS_IPADDR,aton(_PRIMARY_STATIC_IP), \
IFS_NETMASK,aton(_PRIMARY_NETMASK), \
IFS_UP
#endif
```

Finally, we need to make sure that the master library file MYLIBS.DIR has an entry for CUSTOM_CONFIG.LIB. Thus, the top two lines of MYLIBS.DIR contain the two libraries CUSTOMLIBS\MYLIB.LIB and LIB\TCPIP\CUSTOM_CONFIG.LIB

4. DETECTING DOS ATTACKS

A denial of service (DoS) attack is the one which prevents the victim form being able to use all or part of their network connection. A denial of service attack may target a user, to prevent them from making outgoing connections on the network. [5]A denial of service may also target an entire organization, to either prevent outgoing traffic or to prevent incoming traffic to certain network services, such as the organizations web page. These are much easier to accomplish than remotely gaining access to administrative access to a target system. Due to this, these attacks have become most common on the internet. We have tested our EWS with a DoS attack called smurf which is a SYN flood attack (shown in figure 6) and found it is vulnerable to the same. Here we used the detection algorithms for detecting the smurf attack.

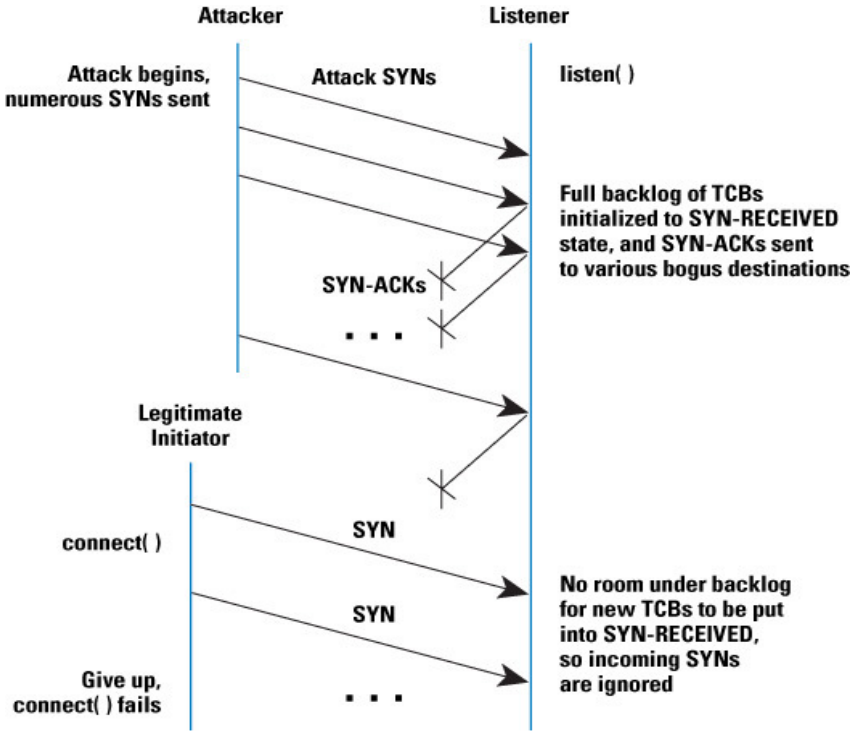

Figure 6: SYN Flood attack

4.1 Detection of DOS Attacks

There are various algorithms which are used to detect the Denial of Service attacks. The algorithms which we used in our current scenario are the Synfindiff [6] and using a finite state machineto detect the Syn Flood attack.

4.1.1 Synfindiff Algorithm

The algorithm provided by Wang, Zhang and Shin in [7] is called the Synfindiff. This algorithm is mainly based onthe CUSUM method described in [8], which belongs to the category of sequential change point detection method.

The difference between the count of incoming SYN's and that of outgoing FIN's, Δ_n, over an observation period of length t_0 is used by Wang et al. The collection of the FIN's begins after a period of t_d after the collection period for SYN's to allow the TCP connections to last longer. $\Delta_n$is normalized by an exponentially weighted moving average (EWMA) of the number of FINs seen in past observation periods $\bar{F}$. They define$\tilde{X} = \Delta_n / \bar{F}$-a where a is the constant chosen to be the mean of $\bar{X}_n$ negative during normal operation. They then define $y_0 = 0$ and $y_n = (y_{n-1} + \bar{X}_n)^+$ (for n > 0; x^+ is x if x > 0 and 0 if otherwise). The algorithm reports an error if $y_n > N$ for some threshold value N.

The other algorithm which we used here against the TCP SYN Flooding Attacks is the TCP Finite State Machine [9].

4.1.2 TCP Finite State Machine Method

This method overcomes the problems that we encounter when the system is flooded with SYN requests. The basic architecture of the TCP protocol is modified here. Here the amount of information stored about each in-progress connection is reduced. During the SYN Flood attack, we get multiple initialization requests. So, a finite state machine has been built to give states. When a synchronization request is arrived, the connection state is SYN_RCVD. A large number on this state could possibly determine a SYN Flood attack. Based on RFC 793 i.e., TCP State Transition Diagram, a new finite state machine for the NEMESI [10] is designed. The TCP state transition diagram is based on two end applications, the sender and the receiver sides and each state in the diagram the sender, the receiver and the NEMESI states.

The design of the defense algorithm is in such a way that it is strongly bound with the previous levels. The number of connections in various states can be obtained by measuring the FSM output and here it is the number of connections in the SYN_RCVD state. The rate of SYN packets per 2ms is calculated at the monitoring level and the values are examined on that level for any anomaly.

The detection algorithm has the following pseudo code:

A. *CompareΔSYN=compare(Current_ΔSYNs, Previous_ΔSYNs)*
B. *CompareΔTCP=compare(Current_ΔTCPpackets, Precious_ΔTCPpackets)*
C. *If (CompareΔSYN AND CompareΔTCP increasing)*
D. *If (SYN_ACK packets are not increasing with the same rate)*
E. *Check the number of connections on SYN_RCVD state*
F. *Check the number of connection on ESTABLISHED state.*

The above is an overview of the detection algorithm. Whenever we detect an attack, we should know from where the attack is coming from in order to counter it. This location of the origin of the attack may be internal or external. The direction of the attack is found out using the tables which are configured to bridge the gap between the external and internal interfaces. Whenever an attack is detected, then the action algorithm makes decision to send the reset packets to reallocate the service.

The detection level which we discussed here is one of the four levels of the NEMESI. The other states are Monitoring, Connection modeling and Action levels. The TCP Finite State Machine Method which we discussed here is a part of the Connection Modeling level.

5. RESULTS AND DISCUSSION

In this work we designed a secured embedded web server to remotely access any electronic input output devices, using Rabbit Core Module RCM 3710 microprocessor. We have designed and implemented an embedded web server for web users or administrators to access their equipment remotely and many other things such as maintenance requests. We have accomplished this task in an efficient and cost-effective manner. Also, since security is a critical issue, we have ensured the authentication and confidentiality by implementing the dynamic C and SSL respectively. We have also provided protection to the system against unauthorized attacks.

Also, here we concentrated on the DoS attacks and tried to implement two remedial methods SynFinDiff and TCP Finite State Machine Method, which proved to work efficiently in detecting the attacks on the system and provided a substantial reliability to the system. The performance of

each in detecting the attacks is as shown below in figure 7. We found that TCP finite state machine algorithm performed well with faster detection time than the synfindiff algorithm.

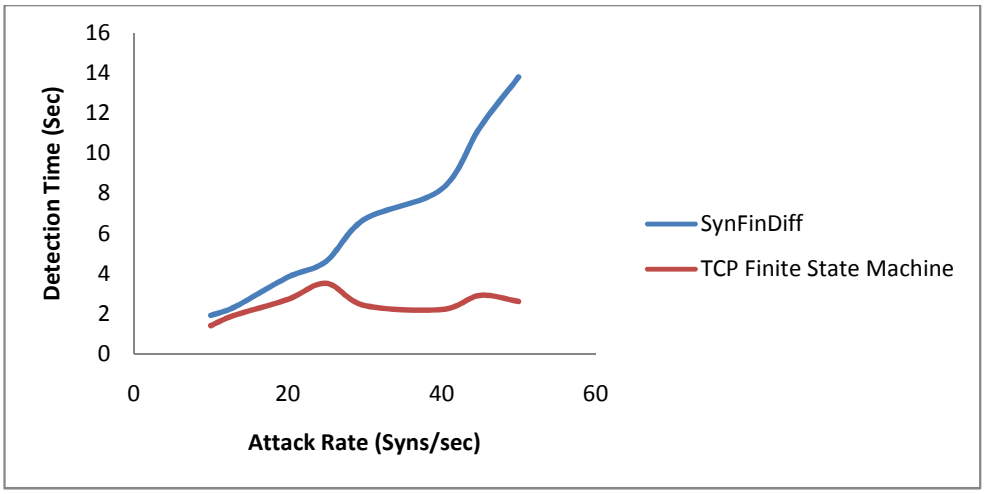

Figure 7: Comparison of SynFinDiff and TCP FSM algorithms for Intrusion Detection.

6. REFERENCES

[1] Guang-jie Han, Hai Zhao, Jin-dong Wang; Tao Lin, Ji-yong Wang, (2003)Webit: A minimum and efficient Internet server for non-PC devices. Global Telecommunications Conference, 2003.GLOBECOM '03. IEEE, pp 2938-2931

[2] Tao Lin, Hai Zhao, Jiyong Wang, Guangjie Han, Jindong Wang, (2004) "An Embedded Web Server for Equipments", IEEE Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, pp 345-350

[3] Xiang Yang,YuanyiZhang,Rongyang Zhao, (2008) "Study and design of home intelligent system based on embedded internet", IEEE International Conference on Embedded Software and Systems Symposia, pp 344-349

[4] Agranat I.(1998) "Embedded Web Servers in Network devices". Communication Systems Design, pp. 30-36.

[5] Mika J. Saaranen, JussiRoivainen and Juha-PekkaSoininen, (2001). Providing Network Connectivity for Small Appliances: A functionally Minimized Embedded Web Server. IEEE Communications Magazine 0163-6804/01 pp74-79

[6] Matt Beaumont-Gay, (2007), A Comparison of Syn Flood Detection Algorithms.Second International Conference on Internet Monitoring and Protection (ICIMP 2007) 0-7695-2911-9/07 IEEE Computer Society.pp 9

[7] Haining Wang, Danlu Zhang and Kang. G. Shin.(2002),"Detecting SYN Flodding attacks", In Proc. of INFOCOM IEEE Communications Society., pp 1530-1539

[8] B E Brodsky and B. S. Darkhovsky. (1994) Non Parametric Methods in Change-Point Problems.Kluwer Academic Publishers.

[9] A. Gemona, I. Duncan, A. Miller, NEMESI: Using a TCP Finite State Machine against TCP SYN Flooding Attacks.

[10] Gemona Anastasia, A.M.R., Ishbel Duncan, Colin Allison, (2004) END TO END DEFENCE AGAINST DDOS ATTACKS. 2004, University of St. Andrews, School of Computer Science: IADIS in Spain.,pp 325-333

[11] Yong-tao Zhou, Xiao-hu Chen, Xu-ping Wang, Chun-jiang YAO, (2008), "Design of EquipmentRemote MonitoringSystem Based on Embedded Web", IEEE International Conference on Embedded Software and Systems Symposia, pp73-78

[12] Srivaths Ravi, AnandRaghunathan and SrimatChakradhar, (2004) "Tamper resistance mechanisms for secure embedded systems", IEEE, Proceedings of the 17th International Conference on VLSI Design, pp 605-611.

[13] Quinma Kang, Hong HE, Hongrun Wang, (2006), "Study on Embedded Web Server and Realization", IEEE 1st International Symposium on Pervasive Computing and Applications, pp675-678.

[14] Wei Chen, Shu-Bo-Qiu, Ying-Chun Zhang, (2008), "The Porting and Implementation of Light-Weight TCP/IP for Embedded Web Server", IEEE 4th International Conference on Wireless Communications, Networking and Mobile Computing, pp1-4.

[15] Guangjie Han, Mo Guan, Hai Zhao,(2004) "EWS: Providing Internet Connectivity for non-PC Devices", Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control", pp 349-354

[16] ChaninMaharak and BoonchaiSowanwanichakul, (2004), "Security Methods For Web-Based Applications on Embedded System", Proceedings of the IEEE Region 10 Conference, pp 56-59

Authors

[1]Mr. Ravi KiranVarma.P is Associate Professor at MVGR College of Engineering, Vizianagaram, with an experience of over 10 years in teaching and 1 year in Industry. He is a Wipro Mission 10X certified faculty. He has achieved CEH and CEI certifications from EC-Council. He has published three papers in international journals and one in international conference. He is a member of IEEE Computer Society, ACM, CSI and ISTE. His areas of research include Network Security, Intrusion Detection Systems and Embedded Systems. He is an active volunteer of IEEE.

[2]Dr. V. ValliKumari is a Professor in Computer Science and Systems Engineering department of Andhra University with over twenty years of teaching experience. She was awarded a gold medal for best research in 2008 in AU. Her areas of research interest include Web, Data and Security Engineering and have 70 publications in conferences/journals of international and national repute. She has eight PhD students working under her guidance and one is awarded. She is currently executing two projects with DST, Government of India.
She is an active consultant to several government/public sector/private organizations. She has successfully managed the Integrated Common Entrance Test for MCA/ MBA in AP State, on behalf of Convener and APSCHE for icet2009.net, icet2010.net and icet2011.net. She is a member of IEEE, ACM, CRSI and CSI.