Harden Single Packet Authentication (HSPA)

Haythem Zorkta and Basel Almutlaq, *Member, IACSIT*

***Abstract*—A new single packet authentication method HSPA is presented in this paper. HSPA works transparently for authenticating remote clients and in its design solved two main problems, resource starvation attack and the lack of association between the authentication process and establishing process. Authentication data of HSPA packet is maintained secure by encrypting it using Rijndael, in cipher block chaining (CBC) mode, with a block size of 192 bits and a key length of 192 bits. HSPA evaluation study, in accordance to processing overhead, buffering, and communication overhead, shows that HSPA overhead is marginal as compared to its improvements in authentication.**

***Index Terms*—Firewall, passive authentication, port knocking, single packet authorization.**

I. Introduction

Many machines now run some form of Firewall to help prevent unauthorized connections to open ports, however, if the machine is needed to run services accessible to the Internet, then this is not always an option. There are countless ways to protect information flowing over a network, but if the software running those services has bugs, then protecting that machine against attacks becomes a much bigger problem [1]. Most software has vulnerabilities and now system administrators cannot rely on the security provided by software manufacturers. Thousands of bugs that could be exploited by malicious attackers have been found in all kinds of network services. Those bugs are often fixed weeks, months or even years after they were made public so the window of exposure for some vulnerabilities can be very high. Critical systems need additional layers of security to prevent zero-day exploit attacks against running services and this is when Passive Authentication Techniques come in handy [2].

Passive Authentication Techniques are Port knocking PK and Single Packet Authentication SPA. They are methods to keep a machine hidden from would-be attackers, and yet allow legitimate users to connect to services running on that machine. In broad terms, they are methods for transmitting information across closed ports, with the aim of authenticating users before allowing them, and only them, to access a protected service [1].

These can be achieved in many ways but in general a client sends a specific sequence of connection attempts to a listening server. The server detects this sequence and opens one of its ports so the client machine can connect to it. This prevents attackers from scanning your network for open ports or attacking network services with 0-day exploits because the protected ports will appear to be closed [2].

Although that port knocking provides some real benefits that enhance security, still has some serious limitations [3], [4]. Single Packet Authentication [5], [4] is a relatively newer protocol that retains all of the benefits of port knocking, but fixes many PK limitations [5].

II. SPA Mechanism

In general a client prepares the authentication data that is put in the payload of a single UDP/TCP packet (SPA packet) then sends the packet to the listening server. The server detects, validates SPA packet and opens one of its ports, which is required. So the client machine can connect to it. For a graphical representation of SPA in action, see Fig. 1.

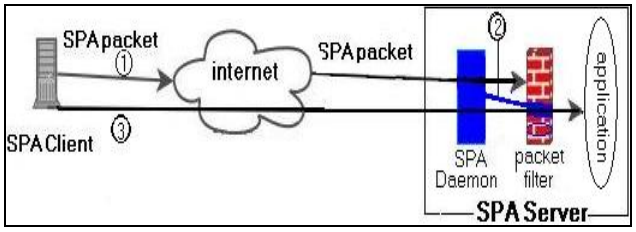

Fig. 1. Single packet authentication basics.

Fig. 1 shows the main steps of SPA:

1) Client send SPA packet
2) After the Server daemon detect validation SPA packet, it reconfigure packet filter to allow Client access to a desired service.
3) SPA Client initiate session with the desired service.

Although that SPA fixes many PK limitations, it stills suffer from many problems. One problem is the lack of association between the authentication process and the follow-on TCP connection to be established. This problem enables the attacker to connect to a protected service on behalf of a valid client, after the client has successfully authenticated with the firewall but before he establishes a TCP connection with this service. Another problem is Resource Starvation Attacks against the server daemon through replaying SPA packet, which make the server daemon consuming resources, and make the authentication service unavailable.

A new proposal HSPA for authentication purposes, which tackles these problems, is presented in this paper.

III. Proposed Design

As any SPA implementation, HSPA relies on a packet sniffer and a completely closed firewall (entire TCP ports are

Manuscript received June 19, 2012; revised August 5, 2012. This paper was accepted by 4th IEEE International Conference on Computer Science and Information Technology (IEEE ICCSIT 2011)

H. Zorkta and B. Almutlaq are with the Aleppo University- Syria (e-mail: drzorkta@hotmail.com, basel_almutlaq@hotmail.com).

closed), which is set to drop all packets that arrive over TCP. That makes firewall to be more silent in its denial (packets that arrive are dropped without sending an ICMP PORT UNREACHABLE back to the initiator) [6], [7]. Two separate software components are required in HSPA, one at client side and the other at the server. Packet filters will be reconfigured after SPA packet is received and validated by server daemon.

HSPA defines his packet format as: (see Fig. 2).

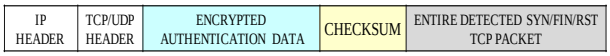

Fig. 2. HSPA packet formate.

Encrypted authentication data: This filed holds encrypted Authentication Data. Rijndael, in Cipher Block Chaining (CBC) mode, with a block size of 192-bits and a key length of 192-bits was used for encryption. *Authentication Data* consist of five fields as shown in Fig. 3.

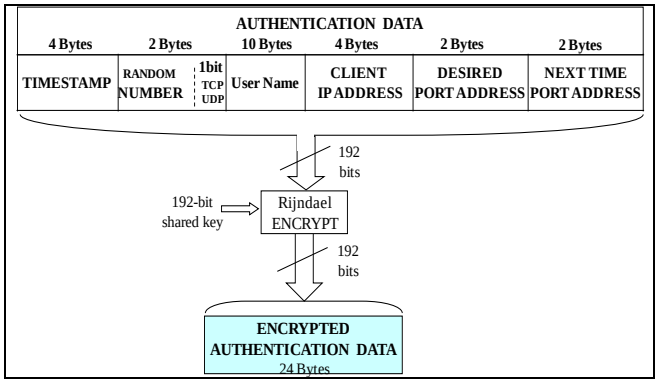

Fig. 3. Encrypted authentication data format.

- *Timestamp* (4bytes) and *random number* (2 bytes) are used to solve Replay Attack Problem. Server daemon compares these 6 bytes with corresponding 6 bytes of last SPA packet received from this client. If it is equal or smaller, then it assumes that there is a replay attack, and takes no action and writes a warning message to WarnLog. Hence, any SPA packet that is intercepted by a third party cannot be replayed on the network in an effort to get access through the default-drop packet filter.
- *TCP/UDP* bit, which is the lest important bit of *random number*, defines the transport protocol (TCP or UDP) of next SPA packet, (1 for TCP, 0 for UDP).
- *User* name (10 bytes) is used to maintain different authorization levels for remote users by the server daemon.
- *Client IP address* (4 bytes) is the IP address of the client.
- *Desired port address* (2 bytes) is the port address that which the client wants to access to it.
- *Next time port address* (2 bytes) represents a port address, which next SPA packet will send over it. It used to solve the DoS attacks to server daemon. Since the destination port address of SPA packet, is different from previous SPA packet, if a third party intercepts any SPA packet, and replays it, there is no listening to this port address by server daemon.

Entire detected SYN/fin/RST TCP packet: is the SYN/FIN/RST TCP packet that the client sends it to server for initiate/finalize/reset TCP session. The client daemon detects this packet and used it entirely in HSPA packet.

Checksum: is the CRC32 digest of both *authentication data* and *entire detected SYN/fin/RST TCP packet*. It is used by server to verify SPA packet integrity, see Fig. 4.

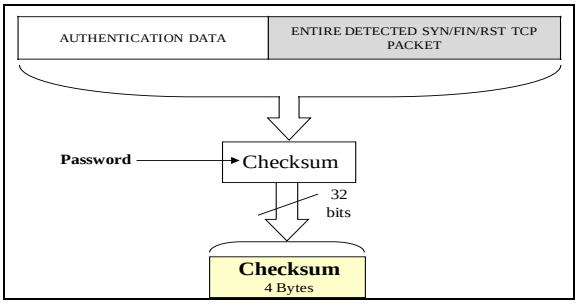

Fig. 4. Checksum calculation.

The following sequence gives overview of how HSPA works:

1) 1- *Server* daemon listens to a number of specific TCP/UDP ports for incoming SPA packet from specific clients, where it specify a TCP or UDP port for each client (see Fig. 5).
2) 2- A client send SYN/FIN/RST TCP packet to the server to initeiate/finalize/reset TCP session. This packet will be droped when it has been arrived, by server packet filter, because all TCP ports are closed (see Fig. 6).
3) 3- Client daemon detects SYN/FIN/RST TCP packet *transmitted* by the client, generates a SPA pcaket and sends it to the server over an agreed TCP or UDP port address. This packet is used to authenticate the client for the server, and it contains the transmitted SYN/FIN/RST TCP packet (see Fig. 7).
4) 4- Server daemon *receives* the SPA packet, decrypts *encrypted authentication data*, validates *user name*, *checksum*, *timestamp* and *random number*, passes the *entire detected SYN/fin/RST TCP packet* into stack and drop it. Server daemon reconfigures the Firewall, allowing client to access the desired port. Finally, server daemon starts listening to the new *next time port address* for next SPA packet from this client.

The first step of TCP three way hand shacking, has been completed now (see Fig. 8). The other two steps will run normally without any intervention by the client and server daemons.

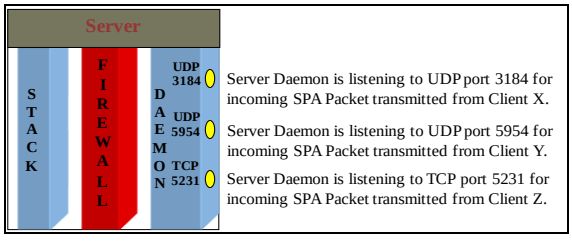

Fig. 5. Primitive (orginal) state of sersver daemon.

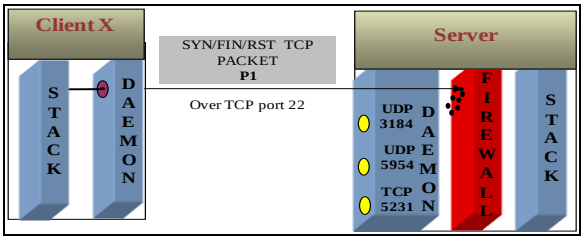

Fig. 6. Transmitting SYN TCP packet by Client.

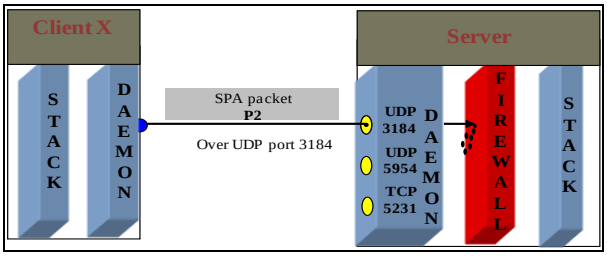

Fig. 7. Transmitting SPA packet by the client daemon to the server daemon.

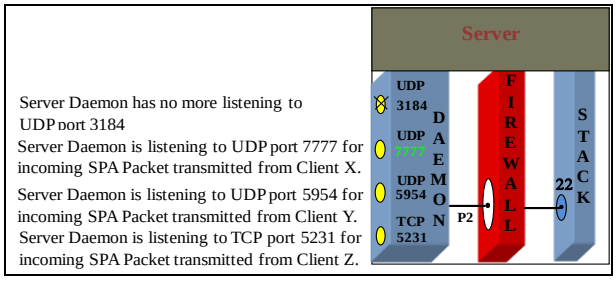

Fig. 8. Server daemon status after detecting valid SPA packet.

where:

- Client Daemon Detect transmitted SYN Packet.
- Client Daemon formats SPA Packet and send it. It's destination port is Next Time Port Address field of previous SPA Packet.
- Server Daemon listening to this port (Next Time Port Address filed of previous SPA Packet of the specific Client) for detect SPA Packet
- No listening.
- Allow role on packet filter to allow all incoming packets exception SYN TCP packet
- Presenting droping pcket by packet filter.

IV. HSPA Evaluation

The most critical parameter for SPA analytical study is the performance factor. Thus, three basic analysis parameters were used to evaluate the HSPA performance: processing overhead, communication overhead and buffering. Moreover strength analysis attempt is presented also.

A. Processing Overhead

Processing overhead is the time amount which added to the main time work of the machines in both sides of the authentication system. While HSPA provide improvements for authentication, it burdens the processor. HSPA Processing Overhead was measured through one second for client and server sides, using Intel Pentium 4 machine running at 3.2 GHz with 4 MHz bus, 2 MB L2 Cache and 3 GB of RAM. Table I list these results (see also Fig. 9 and Fig. 10).

HSPA processing overhead is increased proportionally with the number of HSPA packets sent in time unit. There is 144 µsec and 181 µsec are added to the actual time at client and server sides respectively. HSPA processing overhead is marginal as compared to the improvements in authentication.

B. Communication Overhead (bytes)

Communication overhead is the data amount in bytes, which added to the main transmission traffic volume. HSPA packet format suggests that, the sizes of its fields are (see Fig. 2):

- Authentication data: 24 bytes
- Checksum: 4 bytes
- Entired SYN/fin/RST TCP packet: 40 to 120 bytes (see table II for the headers required for calculations)

Table II illustrates all possible combinations for the minimum and maximum HSPA Communications Overhead.

TABLE I: HSPA Processing Overhead and Percentage Processing Overhead

SPA packets per second	Delay on client side (ms)	Percentage overhead on client side	Delay on Server side (ms)	Percentage overhead on Server side
1	0.293	0.0293%	0.363	0.0363%
2	0.582	0.0582%	0.731	0.0731%
3	0.885	0.0885%	1.101	0.1101%
4	1.163	0.1163%	1.517	0.1517%
5	1.480	0.1480%	1.889	0.1889%
6	1.801	0.1801%	2.263	0.2263%
7	2.090	0.2090%	2.633	0.2633%
8	2.397	0.2397%	3.024	0.3024%
9	2.711	0.2711%	3.410	0.3410%
10	3.007	0.3007%	3.792	0.3792%

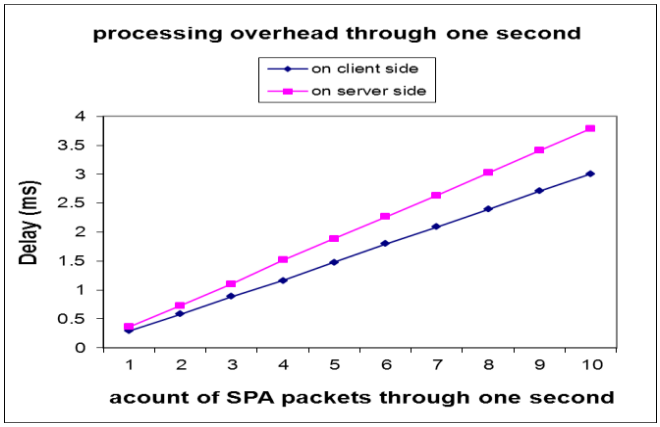

Fig. 9. HSPA processing overhead through one second.

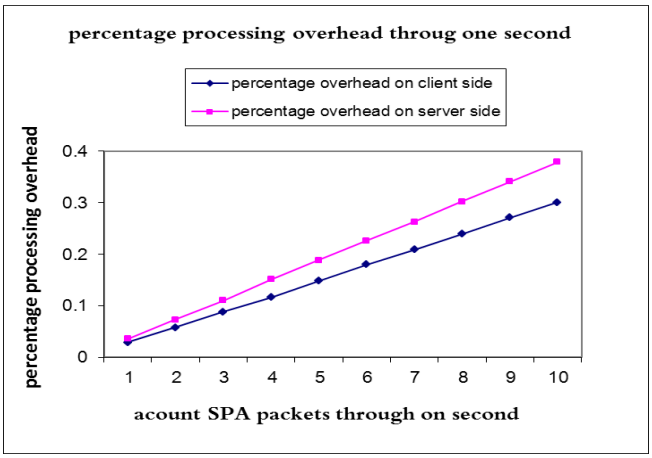

Fig. 10. Percentage processing overhead of the proposed design through one second.

TABLE II: Different Protocols Headers (Min, Max)

Header	Min(bytes)	Max(bytes)
TCP	20	60
UDP	8	8
IP	20	60
ETHERNET	14	14

TABLE III: HSPA COMMUNICATIONS OVERHEAD

state	initiate TCP session over	finalize TCP session over	IP and TCP header size	overhead (Bytes) for each SPA packet to initiate session	overhead (Bytes) for each SPA packet to finalize session	Total overhead for session
1	TCP	TCP	40	122	122	244
2	TCP	UDP	40	122	110	232
3	UDP	TCP	40	110	122	232
4	UDP	UDP	40	110	110	220
5	TCP	TCP	64	146	146	292
6	TCP	UDP	64	146	134	280
7	UDP	TCP	64	134	146	280
8	UDP	UDP	64	134	134	268
9	TCP	TCP	120	202	202	404
10	TCP	UDP	120	202	190	392
11	UDP	TCP	120	190	202	392
12	UDP	UDP	120	190	190	380

Example scenario. suppose, there are 100 clients, each client initiate and finalize 1 session on the server through 1 minute, bandwidth of connection is 64 kbps (65536 bps) and TCP header of SYN/FIN/RST packet have 24 bytes options. Three cases were used to estimate HSPA bandwidth consumption:

1) *The worst case*, all SPA packets that used for initiate and finalize the sessions are transmitted over TCP. In this case, size of 200 SPA packets is 200 * 146 B = 29200 Bytes. This size of bytes will consume 5.94% from bandwidth see Fig. 11.
2) *The intermediate case*, half of SPA packets that used for initiate and finalize the sessions are transmitted over UDP and the other half over TCP. In this case, size of 200 packets is (100 * 134 B) + (100 * 146 B) = 28000 bytes. This size of bytes will consume 5.69% from bandwidth see Fig. 11.
3) *The better case*, all SPA packets that used for initiate and finalize the sessions are transmitted over UDP. In this case size of 200 packets is 200 * 134B = 26800 bytes. This size of bytes will consume 5.45% from bandwidth see Fig. 11.

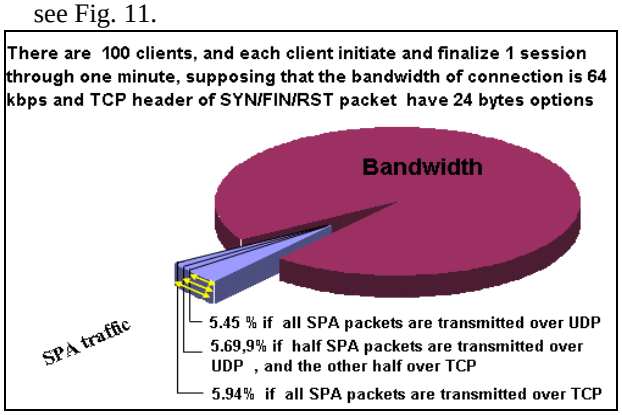

Fig. 11. HSPA bandwidth consumption in the example scenario.

Results in previous scenario, shows that HSPA communications overhead is marginal as compared to the improvements in authentication.

C. Buffering

Since, both of client and server daemons buffering the entire HSPA packet, two suitable buffers are needed at both sides for caching it (see Table III).

With the Worst Case of the last scenario, only 14.3Kbytes is needed at each side, and these buffering requirements are marginal as compared to the HSPA improvements in authentication.

D. Strength Evaluation

In General, SPA can be used to construct authentication systems on firewalls with the goal of only allowing authorized users access to open ports. Although that SPA provides some real benefits that enhance security, it still has some serious limitations. HSPA fixes two of these limitations: Resources Starvation attacks and the lack of association between the authentication process and establishing process. By appending SYN, FIN, and RST TCP PACKET to SPA packet, HSPA solves the lack of association between authentication process and the follow-on TCP connection to be established, where the attacker can connect to a protected service on behalf of a valid client, after client has successfully authenticated with the firewall, but before he establishes a TCP connection with this service.

Moreover, with HSPA, the server listen to a new TCP or UDP port for each SPA packet. Therefore, Resource Starvation Attacks against the server daemon through replaying SPA packet, that makes the server daemon consuming resources, and makes the authentication service unavailable, was solved.

Authentication data of HSPA packet is maintained secure by encrypting it using already exist standard encryption algorithm (Rijndael, in Cipher Block Chaining (CBC) mode, with a block size of 192-bits and a key length of 192-bits).

V. COMPARATIVE STUDY

A comparative study between HSPA and other advanced SPA techniques (aldaba, Fwknop), shows that HSPA not only provides characteristics of other techniques like authorization and encryption (see Table IV), but it also provides solutions for these problems:

TABLE IV: SPA TECHNIQUES COMPARATIVE STUDY

State	Total packet length	Encryption type	Authentication status	a lack of association between authentication and TCP connections being established
HSPA	110 - 202 bytes	Rijndael	TCP Connections	non- exist
Aldaba	Unknown	Blowfish, Twofish Serpent, Rijndael	TCP, UDP Connections	exist
Fwknop	80 - 1600 bytes	Rijndael, ElGamal	TCP, UDP Connections	exist

State	Resource Starvation Attacks	Authorization	SPA packet over Protocol	SPA packet over Port number
HSPA	non- exist	exist	Dynamic UDP,TCP	DYNAMIC
Aldaba	exist	exist	Static TCP SYN	STATIC
Fwknop	exist	exist	Static UDP,TCP, ICMP	STATIC

- A lack of association between authentication and TCP connections being established.

- Resurce Starvation Attacks.

This means that HSPA is robust against attacks more than existing SPA techniques.

VI. Conclusions

A new SPA proposal (HSPA) for authentication purposes is presented in this paper. HSPA provides solutions for:

- A lack of association between authentication and TCP connections being established.
- Resurce Starvation Attacks.

Analysis study proved that HSPA is robust against attacks more than existing SPA techniques, and has marginal processing, communications, and buffering overhead, when compared to the improvements in authentication.

References

[1] S. Jeanquier, "An analysis of port knocking and single packet authorization," MSc Thesis, Royal Holloway College, University of London September 9, 2006.

[2] Security of course is an important thing. If this really bugs you then look into port knocking. [Online]. Available: http://www.aldabaknocking.com/faq

[3] A. Manzanares, J. Marquez, J. Tapiador, and J. Castro, "Attacks on port knocking authentication mechanism," *ICCSA LNCS-Springer Berlin/Heidelberg*, 2005, pp. 1292-1300.

[4] M. Rash, "Protecting SSH servers with single packet authorization," in *the Linux Journal*, May 2007.

[5] M. Rash, "Single packet authorization with fwknop," *Usenix; login: Magazine*, vol. 31, no. 1, February 2006, pp. 63–69.

[6] R. deGraaf, J. Aycock, and M. Jacobson, "Improved port knocking with strong authentication," Department of Computer Science, University of Calgary, Canada (fdegraaf, aycock, jacobsg@cpsc.ucalgary.ca).

[7] C. Borss 'DROP/DENY vs. REJECT'. 2001, Listserv post to Braunschweiger Linux User Group (lug-bs@lk.etc.tu-bs.de).

Assoc. Prof. Dr. Haythem Zorkta was bron in 1970, Damascus – Syria. He is a lecturer at Aleppo University- Syria. Master and PhD degree at computer networks security from MTC- Cairo- Egypt. A lot of international and local papers at the same field, and a technical reviewer for many international and local conferences. Supervisor for many Master and PhD thesis's.

Eng. Basel Almutlaq was bron in 1980, Damascus-Syria Master degree from Aleppo University (2011) at network security Many local papers and International participation at (IEEE ICCSIT 2011).