Additional Security Mechanism in Single Packet Authorization

Co-authored by

Sukhraj Singh Brar
Student Id 141882
sbrar3@student.concordia.ab.ca

Sergey Butakov
sergey.butakov@concordia.ab.ca

Project Report

Submitted to the Faculty of Graduate Studies
Concordia University of Edmonton

in Partial Fulfillment of the
Requirements for the Final
Research Project for the Degree

MASTER OF INFORMATION SYSTEM SECURITY MANAGEMENT

Department of Information Systems Security & Assurance Management
Concordia University of Edmonton,
Edmonton, Alberta, Canada

April 2021

Abstract

Single Packet Authorization (SPA) is the technique used to guard camouflaged network services against unauthorized users. SPA hides specific ports from the external world to reduce attacks at TLS ports until a cryptographically protected packet is received and authorized by the transport layer. Although SPA helps to overcome significant flaws in conventional Port Knocking mechanisms, its fragility related to potential key leakage makes it susceptible to various attacks. Additional security measures like Segregation of Duties help to reduce the potential of these attacks. This paper demonstrates the architecture and practical implementation of Segregation of Duties and discusses the impact of the additional security mechanisms on SPA usability.

Keywords: Cryptographically, Single Packet Authorization (SPA), TLS, Port Knocking (PK), Segregation of Duties, Network services

Table of Contents

List of Tables

List of Figures

Introduction

The internet's fundamental goal is to connect people, and therefore, it was built as an open platform. When people began sharing vital information over the internet, security became a significant concern. Therefore, to isolate a network with sensitive information from the internet, network administrators started deploying firewalls: devices that can differentiate between legitimate and prohibited traffic. Over time firewalls evolved, but despite all the developments, they were still susceptible to various attacks. Therefore, to add an extra layer of protection on top of firewall technologies, a security mechanism known as Port Knocking (PK) was proposed. The PK's primary approach was to open services for authorized users while maintaining the protected system virtually hibernated for unauthorized users. PK presented an authentication system through which closed ports are opened to authorized users to establish communication on fortified network services (deGraaf, 2007).

The port knocking server requires the client to knock the ports in a predefined sequence. When the server receives a correct order of ports, it opens a specific port to allow the client to make communication.

PK provided the defense-in-depth and offered some practical applications. Still, its vulnerabilities like out-of-order delivery and replay attacks embodied significant flaws in a generic PK scheme. However, the improvement of PK - Single Packet Authorization (SPA) fixed various Port Knocking flaws such as out-of-order delivery and replayed attacks without remodeling PK's functionality. The significant difference between SPA and Port Knocking is the layer of data transference. SPA moved data transmission to the application layer, allowing SPA to transmit packets of the size comparable to network MTU. Holding easy access to a large amount of packet data opened up the vast range of possibilities. Therefore, instead of port numbers, the SPA uses an encrypted packet for authentication, which solved the problems like out-of-order delivery and replay attacks. Given the fact that SPA still relies on the keys stored on the client and server sides; therefore, it naturally inherited PK's vulnerability related to stolen keys or key leakage. This paper proposes a way to disgrace the attack surface by adding Segregation of Duties in key administration. (Rash, Fwknop-Single Packet Authorization, 2016)

Review of Related Work

The discussion below looks at the available PK and SPA services and reviews them against potential vulnerabilities like:

Replay Attack

When the attacker captures the traffic between two communicating systems and retransmits it to gain access to the protected system breaching the system's integrity.

Man-in-the-Middle Attack

When the attacker can listen or modify traffic between two communicating nodes, it breaches the communication's confidentiality or integrity.

Privilege Escalation

When an attacker exploits a bug, design flaw, or configuration oversight to gain elevated access to resources.

Administrator Collusion Attacks

When an administrator deliberately or accidentally helps malicious actors exploit the system or leak critical information.

Compatibility with Additional Mechanisms

The system's ability to blend with features added on top of its original architecture.

Key Leakages

It is an event in which keys are voluntarily or involuntarily leaked to an unauthorized person.

Compromised Host

Any system under the control of the malicious user. A compromised host can leave a backdoor to the environment to allow the attackers to retain their foothold within the organization's network.

Brute Force Attacks

A brute force attack uses trial-and-error to guess credentials for the authorization mechanism. This method tries all possible combinations.

Denial of Service attacks

A Denial-of-Service (DoS) attack shuts down a machine or network, making it inaccessible to its intended users. Typically, attackers flood the victim with traffic or send it data that triggers a crash.

	Fwknop	**Web-Knocking**	**BridgeSPA**	**Knockknock**	**WebSPA**
Packet Used	UDP packet with an encrypted payload	Authorization Web knock containing encrypted payload (OTP, IP address, port, protocol TCP or UDP)	TCP SYN request	Encoded keys in TCP header, I.e., TCP SYN	HTTP Base64 URL-Safe encoded URL

Table 1 The Architecture of Different SPA mechanisms

Name/ Risks	**Fwknop**	**BridgeSPA**	**WebKnocking**	**KnocKKnock**	**WebSPA**
Race Attack	Possible, if the attacker has access to traffic	Possible, but each packet contains a timestamp	Not Possible, as each OTP used once only	Possible, if the attacker knows the architecture of TCP request	Possible within 60 Seconds
Reused Key	Possible, as the Same key used	Key is effective for a limited time only	Not Effective, as OTP is used instead of keys	Possible, as same MAC key and Cipher key used	Instead of a key, Username and Password combination used
Compromised Client	Can request access to any machine, leak keys	Leak information to the attacker	A compromised client can request access for the attacker	Key leakage	Compromised client can leak credentials
Denial of	Possible, the server becomes	Possible, the server becomes unresponsive	Possible, the server becomes unresponsive	Possible, the server becomes unresponsive	Possible, the server becomes

Service Attack	unresponsive when overloaded with data.	when overloaded with data	when overloaded with data	when overloaded with data	unresponsive when overloaded with data
Man-in-the-middle attack	Possible, an attacker can capture traffic by sitting between client and server.	Possible, an attacker can capture traffic by sitting between client and server.	Possible, an attacker can capture traffic by sitting between client and server.	Possible, an attacker can capture traffic by sitting between client and server.	Possible, an attacker can capture traffic by sitting between client and server.
Privilege Escalation	Possible	Possible	Possible	Not possible	Possible, but can be prevented
Brute-Force	Possible, as the same key used	Infeasible, as SYN flood mitigated by firewall	Not possible because OTP used	Possible, as same keys used	Possible, for credentials
Administrator Collision Attack	Possible	Possible	Possible, the administrator can email OTP to an attacker	Possible	Possible
Proper Documentation	Available till version 2.6.8	Only paper, No documentation	No Documentation	A brief tutorial on GitHub	Official documentation available
Compatibility with Additional Mechanisms	Yes	Yes	No	Yes	Yes

Table 2 Some Of these risks are assumptions based upon theoretical models of different SPA tools.

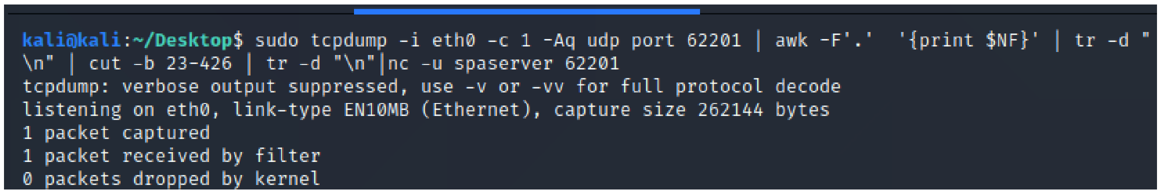
Examples of attacks on various PK/SPA mechanisms

Fwknop

Fwknop is a documented Single Packet Authorization mechanism. Nevertheless, Official documentation is available for version 2.6.8, released on December 23, 2015, while the latest available version, 2.6.10 released on August 06, 2018. Fwknop is a secure mechanism but still has some vulnerabilities. If the attacker and a client are on the same network, the attacker can perform an ARP cache spoofing attack and force a client to send all their traffic through the attacker. Since, by default, the server is listening at port 62201, the attacker can redirect traffic at port 62201 to any other port. After extracting encrypted UDP data from the packet, the attacker can resend it by encapsulating it in a new packet. The attacker is required to spoof its IP address as the client's because the client's IP address is encoded in the UDP data packet to perform a successful attack. Performing such an attack is a tricky task but not impossible.

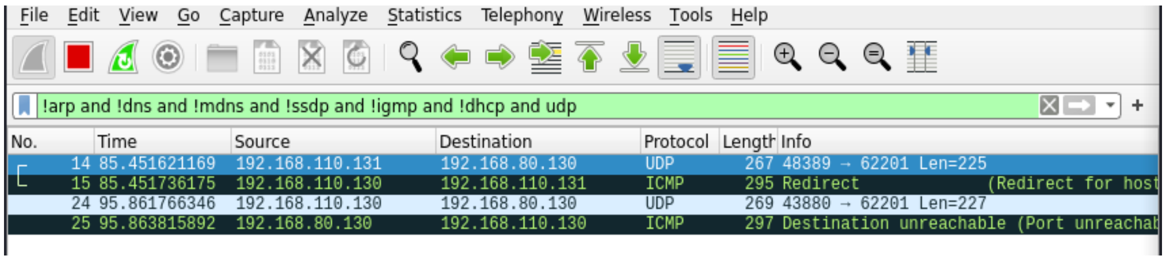
The proposed attack is executed successfully on a local setup where the attacker and a client were on the same network while the server was on a different network. Firstly, the attacker performed an arp-cache poisoning attack, forcing the client to transmit all traffic through the attacker. Then the attacker captured the client's traffic and filtered out the specific UDP packet on the ethernet wire. Lately, using the "nc" command, regenerated a packet for the server.

Figures 1 and 2 display ongoing race attacks.



```
kali@kali:~/Desktop$ sudo tcpdump -i eth0 -c 1 -Aq udp port 62201 | awk -F'.'  '{print $NF}' | tr -d "
\n" | cut -b 23-426 | tr -d "\n"|nc -u spaserver 62201
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
1 packet captured
1 packet received by filter
0 packets dropped by kernel
```

Figure 1. Attacker Capturing Data and Replaying It.



```
!arp and !dns and !mdns and !ssdp and !igmp and !dhcp and udp
```

```
No.   Time          Source           Destination      Protocol  Length Info
   14 85.451621169  192.168.110.131  192.168.80.130   UDP       267 48389 → 62201 Len=225
   15 85.451736175  192.168.110.130  192.168.110.131  ICMP      295 Redirect            (Redirect for host
   24 95.861766346  192.168.110.130  192.168.80.130   UDP       269 43880 → 62201 Len=227
   25 95.863815892  192.168.80.130   192.168.110.130  ICMP      297 Destination unreachable (Port unreachab
```

Figure 2. Wireshark shows retransmitted traffic by the attacker.

Implementation with NAT

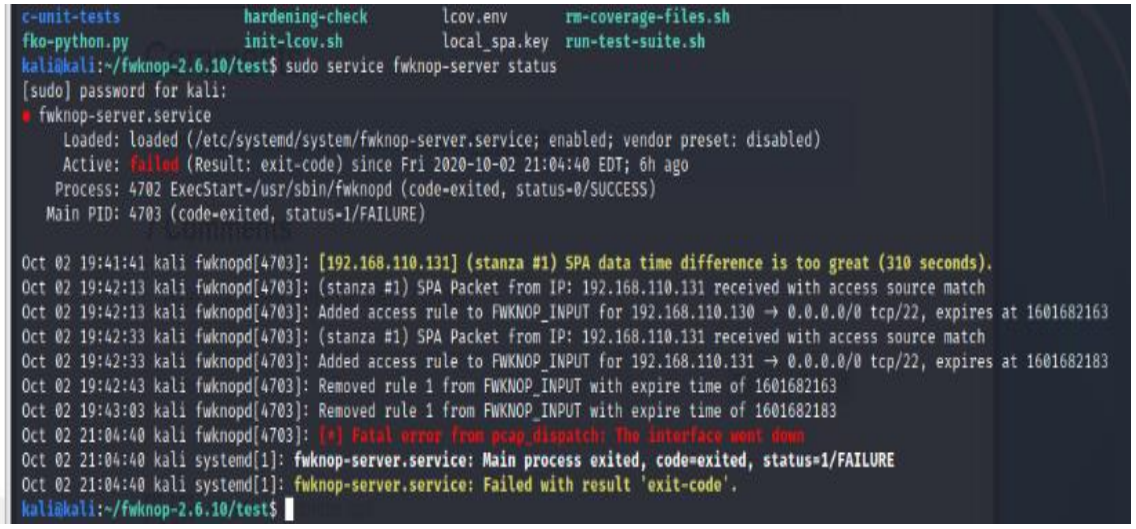
In fwknop, the client's IP address is encoded in the UDP packet. However, during the NAT, IP addresses in the header are altered by Nat-router. When the fwknop server receives the packet with different IP addresses in the packet's header and body, the server will ignore the packet. The solution for this is to provide the IP address of the gateway when generating requests. However, this approach also has some limitations. Supplying the gateway's address will give access to all devices behind the NAT interface, and rogue insiders can become a significant threat. (Rash, Single Packet Authorization, 2007)

Compromised Client

Fwknop allows the client to request any device's access by providing its IP address. However, with this feature, the attacker can compromise the client and then send a request to access any device. After getting authenticated, the attacker can perform an SSH connection with the server from its IP address. If the attacker gets physical access to a client, the attacker can generate a SPA request for any machine directly by giving own IP address and then access the SPA server.

Unexpected behavior

While testing, it is observed that the packet replayed repeatedly enhances the potential for a Denial-of-the-Service attack on the fwknop server. Assume an attacker has captured traffic through the Man-in-the-middle attack and replaying the captured packet; if the time duration between captured packet and its retransmission time is high, the server can crash unexpectedly. Therefore, attackers can willingly or unwillingly force the server to shut down and result in a Denial-of-the-Service attack.



```
c-unit-tests             hardening-check        lcov.env       rm-coverage-files.sh
fko-python.py            init-lcov.sh           local_spa.key  run-test-suite.sh
kali@kali:~/fwknop-2.6.10/test$ sudo service fwknop-server status
[sudo] password for kali:
● fwknop-server.service
     Loaded: loaded (/etc/systemd/system/fwknop-server.service; enabled; vendor preset: disabled)
     Active: failed (Result: exit-code) since Fri 2020-10-02 21:04:40 EDT; 6h ago
    Process: 4702 ExecStart=/usr/sbin/fwknopd (code=exited, status=0/SUCCESS)
   Main PID: 4703 (code=exited, status=1/FAILURE)

Oct 02 19:41:41 kali fwknopd[4703]: [192.168.110.131] (stanza #1) SPA data time difference is too great (310 seconds).
Oct 02 19:42:13 kali fwknopd[4703]: (stanza #1) SPA Packet from IP: 192.168.110.131 received with access source match
Oct 02 19:42:13 kali fwknopd[4703]: Added access rule to FWKNOP_INPUT for 192.168.110.130 → 0.0.0.0/0 tcp/22, expires at 1601682163
Oct 02 19:42:33 kali fwknopd[4703]: (stanza #1) SPA Packet from IP: 192.168.110.131 received with access source match
Oct 02 19:42:33 kali fwknopd[4703]: Added access rule to FWKNOP_INPUT for 192.168.110.131 → 0.0.0.0/0 tcp/22, expires at 1601682183
Oct 02 19:42:43 kali fwknopd[4703]: Removed rule 1 from FWKNOP_INPUT with expire time of 1601682163
Oct 02 19:43:03 kali fwknopd[4703]: Removed rule 1 from FWKNOP_INPUT with expire time of 1601682183
Oct 02 21:04:40 kali fwknopd[4703]: [*] Fatal error from pcap_dispatch: The interface went down
Oct 02 21:04:40 kali systemd[1]: fwknop-server.service: Main process exited, code=exited, status=1/FAILURE
Oct 02 21:04:40 kali systemd[1]: fwknop-server.service: Failed with result 'exit-code'.
kali@kali:~/fwknop-2.6.10/test$
```

Figure 3. Unexpected behavior of the fwknop server.

KnockKnock

KnockKnock is not a complex system as compared to Fwknop. It is an open-source tool available on GitHub. The installation for KnockKnock is a smooth process. However, it has one problem with its documentation. In the documentation, the first argument required for creating profiles is the port number, while the profile name is the second argument. However, in actuality, it is vice versa which can be a headache while generating profiles. It generates a profile named the port number when giving the arguments according to the documentation. Knockknock is not covert to the intruder in the Man-in-the-Middle attack. It allows raw sockets to respond to and initialize the connection after reading a valid logged TCP SYN on superuser privileges. (Marlinspike, 2011)

Compromised Keys

The discredited host can be a prominent intricacy for both fwknop and knockknock; a discredited host can leak the keys to the attacker, making the whole system vulnerable due to the absence of a supplementary protection mechanism.

Administrator Collusion Attack

Both fwknop and knockknock depend upon the administrator to create keys and share the keys with the client. If due to any reason the administrator goes rogue, it causes a single point of failure as the administrator can share these keys with potential intruders. This responsibility should divide between two or more persons, i.e., Segregation of Duties to eliminate the possibility of administrator collusion attack. With the implementation of Segregation of Duties, one person solely does not have the complete information, which results in a preventive measure. Segregation of Duties is the concept of internal controls. These controls must be implemented at the SPA daemon to prevent administrator collusion attacks. To implement proposed controls, one person must be accountable for generating keys but should not know which key is deployed. The other person should issue keys but must not have actual keys.

Improvements Made in knockknock

Changed argument According to Documentation

The first step was to modify the sequence of arguments required during profile generation and make it according to the official documentation to make knockknock better and efficient. For this modification, changes were requisite in the python code in the knockknock-genprofile.py file. It will make it convenient for users to work with knockknock with the help of documentation.

Implementation of Segregation of duties

Segregation of Duties helps overcome administrator collusion attacks in the knockknock by separating network administrator duties into two parts. Rather than making a single person responsible for creating, storing, and delivering keys, these jobs will split between two individuals after implementing Segregation of Duties. One person will be responsible for creating keys, known as Key Creator, responsible for generating multiple keys. Key Creator will have all actual keys but will have no idea among all these keys which key is valid. The second administrator will be the honey-checker administrator, responsible for managing the honey-checker. The second person will know which key is valid but will not have the actual key. After implementing Segregation of Duties, no single person going to have full access to the system results in mitigating administrator collusion attack and key leakage attack on the server-side. Following are the changes made in knockknock architecture:

Key Storage

On the server-side, instead of storing keys in files, keys are now stored in MySQL Database. While on the client-side, still file system is used for storing keys.

Multiple Keys

In the original version, on the server-side, only one key for each profile was stored. However, in the new version, rather than storing just single key multiple keys are stored for each profile.

Honey-checker

In the new version of knockknock, a honey-checker is deployed, which checks whether a key is valid or not. For a single client, server-side database stores multiple keys. Out of these keys, just one is a valid key. In a possible scenario, if an attacker stole the keys database from the server, it will still not send an authentic knock to the server unless it discovers which is a valid key. The server does not know which key is the actual key. It has to rely on the honey-checker for authorization. In the future, a honey-checker can also serve as a honeypot to eliminate brute force by setting traps for unsuccessful attempts.

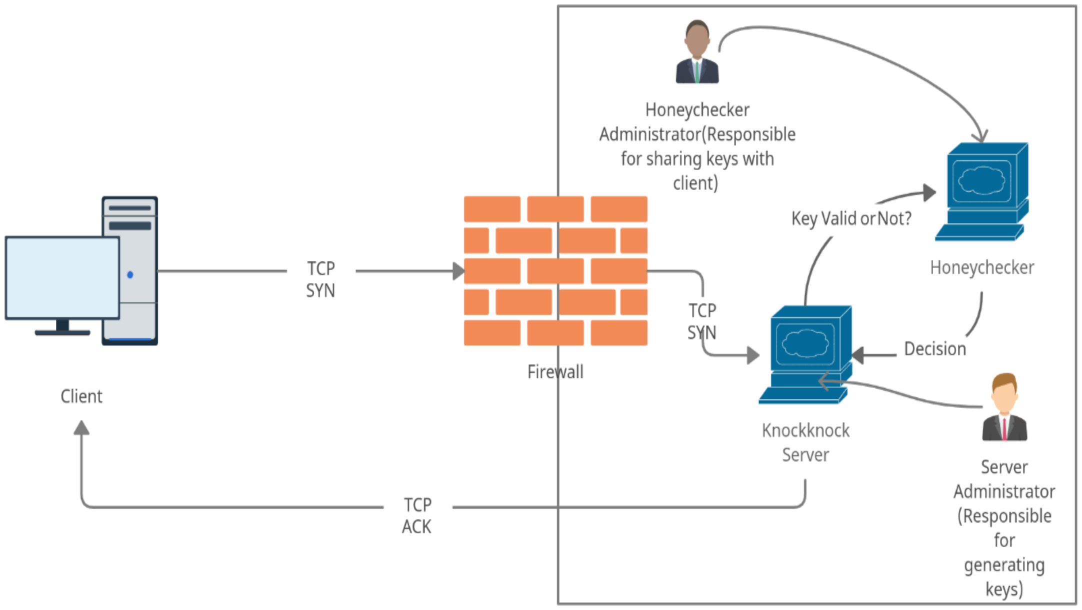

Honeychecker Administrator(Responsible for sharing keys with client)
Key Valid or Not?
Honeychecker
TCP SYN
Client
Firewall
TCP SYN
Decision
Knockknock Server
Server Administrator (Responsible for generating keys)
TCP ACK

Figure 4. The architecture of SPA after implementing Segregation of Duties.

Knockknock Architecture After changes

The new version of knockknock consists of two parts server and a client. However, the server-side is partitioned into two parts: the server itself and the other is a honey checker. Firstly, a client sends a single SYN request to the server on a specific port. Upon receiving that request, the server extracts the key from information sent by the client. After extracting a key, the server looks for it in the keys database. If a key is present in the database, the server sends it to the honey checker to identify whether it is a valid key or a honey key. If it is an original key, the honey checker responds to the server, and the server opens the port requested by the client. The rest of the mechanism kept the same as the original knockknock. With these changes in mechanism, a Role-based Access control system has been designed. One person - server administrator, is responsible for generating keys and managing the keys database on the server-side. The server administrator will have access to all keys but will not have any information about the valid key.

The other person - the honey checker administrator, will be responsible for managing the honey checker and sharing keys with the client. The honey checker administrator knows which key is valid but will not have the actual key. The architecture after implementing Segregation of duties and roles of server administrator and honey checker administrator explained in Figure 4. Since no single person has access to the entire network; therefore, the goal to implement Segregation of Duties achieved. Moreover, if an attacker successfully steals the entire database from the SPA server, that person will still have to try all keys to send a valid request; this provides an extra defense layer.

Experimental results

The implementation may affect usability through the extra time required for authentication. Therefore, ran various tests to calculate the time between sending KnockKnock requests from the client-side and opening port on the server-side upon receiving a valid knock authenticated by the honey-checker. The whole process repeated 30 times using: 10 keys, 50 keys, 100 keys, 250 keys, 500 keys, and 1000 keys. The following are the results:

(i) The average time for 10 keys is 0.2265 seconds
(ii) The average time for 50 keys is 0.2215 seconds

(iii) The average time for 100 keys is 0.2533 seconds
(iv) The average time for 250 keys is 0.3886 seconds
(v) The average time for 500 keys is 0.6003 seconds
(vi) The average time for 1000 keys is 1.0170 seconds

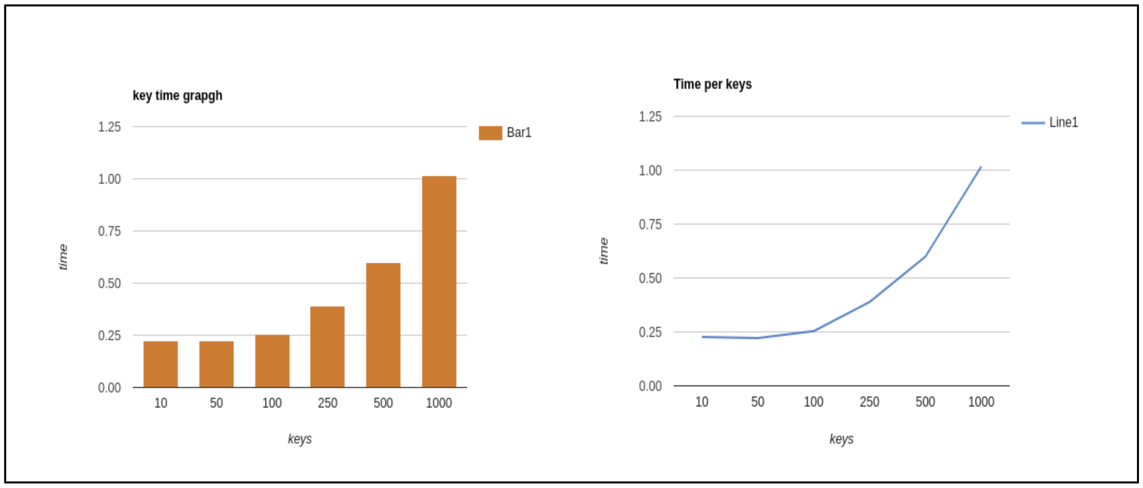


Figure 5. Graphs are demonstrating the key time ratio.

As results indicate, the increasing number of keys can increase the server's time for validating the request. However, for 1000 keys, the average time to validate the request is just touching the 1-second mark, which is quite acceptable. Because of the human perception, 1-second is about the limit for the user's flow of thought to stay uninterrupted, would usually not catch a user's attention, and will not hinder the user experience. (Nielsen, 1993)

Conclusion

The project described in the paper looked at the number of vulnerabilities in the port knocking and single packet authorizations techniques. It proposed to use Segregation of Duties – an additional security mechanism for Single Packet Authorization systems that utilize decoy keys to set up a role-based access control system. Such extension will help to mitigate Administrator collusion attacks. The system's performance was analyzed after implementing the Segregation of Duties, and it was shown that implementing the Segregation of Duties expectingly added a slight delay into the authentication process. However, the lag was below one second, and therefore it did not significantly impact the user experience while interacting with the system. Additional improvements in the proposed mechanism can be added by extending the system with additional features to send multiple keys at a time to deceive man-in-the-middle attackers.

Bibliography

Almeshekah, M., & Spafford, E. (2014). Planning and integrating deception into computer security defenses. Proceedings of the 2014 New Security Paradigms Workshop (pp. 127–138). Victoria, British Columbia, Canada: Association for Computing Machinery. doi:10.1145/2683467.2683482

Butakov, S., Zavarsky, P., & Mirheydari, S. (December 15, 2019). Honeykeys: deception mechanisms in single packet authorization . Proceedings of the 14th Pre-ICIS Workshop on Information Security and Privacy. Munich, Germany.

deGraaf, R. N. (2007). Enhancing firewalls: conveying user and application identification to network firewalls. doi:10.11575/PRISM/1377

Jeanquier, S. (2006). https://www.securitygeneration.com. Retrieved October 27, 2020, from https://www.securitygeneration.com/single-packet-authorization/

Juels, A., & Rivest, R. L. (2013). Honeywords: making password-cracking detectable. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13), 145-160. Retrieved from http://hdl.handle.net/1721.1/90627

Liew, J.-H., Lee, S., Ong, I., Lee, H.-J., & Lim, H. (22-24 June 2010). One-Time Knocking framework using SPA and IPsec. 2010 2nd International Conference on Education Technology and Computer, 5, pp. 209-213. Shanghai. doi:10.1109/ICETC.2010.5529780

Manzanares, A. I., & Márquez, J. T. (May 9-12, 2005). Attacks on Port Knocking Authentication Mechanism. International Conference on Computational Science and Its Applications,ICCSA 2005. 3483, pp. 1292-1300. Singapore: Springer, Berlin, Heidelberg. doi:https://doi.org/10.1007/b136266

Marlinspike, M. (2011, September 15). KnockKnock. (Github) Retrieved October 26, 2020, from https://github.com/moxie0/knockknock

Merki, O., Pavlosoglou, Y., Golen, P., Rouiller, J., & Arciszewski, P. (2014, November 21). WebSpa Single HTTP/S Request Authorisation Web Knocking. Retrieved October 26, 2020, from https://phoenixnap.dl.sourceforge.net/project/webspa/webspa-08.zip

Miedaner, D. M., & Pavlosoglou, D. (2012, November 15). Single Packet Authorization on the WEB -- WEB-SPA. Retrieved october 26, 2020, from https://owasp.org/www-pdf-archive/OWASP_Presentation_WEB_SPA.pdf

Nielsen, J. (1993). Usability Engineering. In J. Nielsen, Usability Engineering (p. Chapter 5). Cambridge: AP Professional.

Pourvahab, M., Atani, R. E., & Hoss, F. (2014/3). Enhanced Secure Web-Knocking Using Single Packet Authorization (SPA) and One-Time Password (OTP). International Journal of Current Life Sciences. 4, pp. 504-511. India: Bret Research Journals. doi:10.2437/ijcls

Rash, M. (2007, August 27). Protecting SSH Servers with Single Packet Authorization. Retrieved October 26, 2020, from https://www.ee.ryerson.ca/~courses/coe518/LinuxJournal/elj2007-157-protectSSH-singlepkt.pdf

Rash, M. (2007, April 1). Single Packet Authorization. (Linux Journal) Retrieved Octobet 26, 2020, from https://linuxjournal.com/article/9565

Rash, M. (2016, January). Fwknop- Single Packet Authorization. (Cipherdyne) Retrieved October 26, 2020, from https://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html

Smits, R. (2011). BridgeSPA: A Single Packet Authorization System for Tor Bridges. WPES '11: Proceedings of the 10th annual ACM workshop on Privacy in the electronic society. Chicago Illinois USA: UWSpace. Retrieved from http://hdl.handle.net/10012/6446