ATLANTIS PRESS
Advances in Computer Science Research, volume 63
International Conference on Artificial Intelligence and Engineering Applications (AIEA 2016)

Internet Key Exchange Protocol Fuzzing Test On Extended Peach Framework

Jiawei Zhang [a], Yijie Shi [b]
School of Beijing University of Posts and Telecommunications, Beijing 100876, China
[a]jwzhang2016@163.com, [b]yijieshi2000@bupt.edu.cn

Abstract. With the development of computer network technology, network security issues are also making more and more people pay attention. IPsec protocol as a viable solution to IP communications security was put forward, and many IPsec products were also launched. Therefore, to detect and mine possible vulnerabilities of IPsec-related products is important and necessary. Fuzzing test can help us find these vulnerabilities in these IPsec product implementations. In the current popular Fuzzing tools, Peach is a widely recognized open-source fuzzing framework. However, the current Peach is unable to support IKE protocol. The paper achieved fuzzing test for IKE protocol by extending the framework of the Peach.

Keywords: Peach; Fuzzing; IPsec; IKE; Frankencert.

1. Introduction

In modern time, Internet has been integrated into the people's daily life. At the same time, network security issues have been exposed. User privacy and sensitive information at any time are likely to be stolen and destroyed by computer viruses in network, resulting in heavy losses. In the network layer, IPsec is used to increase the security of IP communications [1]. Many large manufacturers have also launched IPsec products. The achievement of these IPsec product directly affect the size, stability and scalability of IPsec network. Therefore, it is important and necessary to detect and mine possible vulnerabilities for IPsec related products. It has been widely used in industry through the Fuzzing test for vulnerability mining. Fuzzing test has been an important safeguard for network security [2].

Peach Fuzzer is an open-source test tool that attempts to discover security vulnerabilities by sending random input to an application [3]. It is widely used to test for security bugs in input validation as well as in the application logic. However, Peach cannot fuzz encrypted protocols, such as IKE. Hence, we propose a novel Peach improvement on IKE for IPsec products vulnerability detection. We extended Peach and integrated some the latest research results such as Frankencert [4] in order to make peach more powerful.

The outline of this paper is as follows: in Section 2, we briefly describe the Peach Framework, Fuzzing test, IKE protocol and Frankencert; in Section 3, we present our new approach for Peach improvement; in Section 4, we demonstrate the experimental results by Pcap packet and analysis our data; Finally, we conclude in Section 5.

2. Related Works

Peach *Fuzzer*

Peach, developed by Michael Eddington and some others, is a fuzzy test framework. At first, it was written by Python in 2004, version 2 in 2007, and it was rewritten by C# in version 3 which is the newest version.

Peach uses Pit File to be a test script which is written in XML. The main parts of Peach include initial, engine, parser, state machine, state, operation, agent, monitor, strategy of mutant, mutant machine and etc.

Compared with SPIKE, Sulley, Codenomicon and other fuzzy test tools, Peach is the most flexible framework can be gotten. As for developers, using Peach can achieve code reusing to a maximum extent and reduce developing work hugely.

IKE protocol

In order to improve IP's safety in opening network, IETF (Internet Engineering Task Force) release protocol suits IPSec, which is based on cryptography. IPSec belongs to network layer protocol, and uses technologies as encryption, decryption, authentication and digital signature, in which achieves transparent and safety protection of above protocols and applications at the lowest layer of TCP/IP's communication. So that it hugely improves the safety of TCP/IP. Internet Key Exchange protocol (IKE), as an important part of IPSec, provides safety service for other two protocol---AH and ESP.

IKE: It belongs to a kind of hybrid key exchange protocol and consists of three parts [5]:

ISAKMP: It is a key exchange framework, independent of the specific key exchange protocol. It defines the architecture of message exchange

OAKLEY: A series of key exchange patterns are described, and the services provided by various modes are defined.

SKEME: A general key exchange technique is described. This technique provides anonymity, non-repudiation and fast refresh.

Frankencert

Certificate validation in SSL/TLS implementations is critical for Internet security. There is recent strong effort, namely frankencert, in automatically synthesizing certificates for testing certificate validation [6].

Frankencert——by generating a certificate from the real certificates in a random structure, you can ensure that the correct syntax, and covers a variety of common or very common extensions and constrains, and can form a X.509 certificate chain. The basic idea of Frankencerts is to use a stack of certificates as seeds, and to create a new test certificate using random variations and extensions in different fields. (Frankencerts).

Frankcert generation process:

It randomly extracts some certificates to form a part of the certificate, in addition to the RSA key and Issuer, which is another assignment. So the Frankencert can be used as an intermediate certificate. The Issuer domain of each Frankencert must be equal to the certificate object in the higher level of the current certificate chain [7].

3. Improvement on Peach Fuzzing Framework for Key Exchange in IPsec

As previously mentioned, our Fuzzing Framework is built on top of Peach (version 2.3.10) to facilitate development. However, a fuzzer for IKE protocol has several features that distinguish it from fuzzers targeting other more common network protocols.

3.1 Improvement of Peach Receive Action

In the original PeachPit of Peach, due to the lack of support for the length of the field, it cannot provide a good analysis of the contents of the message received. This leads to the IKE handshake process unable to be carried on. In order to better interact with the device being tested, we should modify the Peach's collection mechanism. In the original PeachPit, it has a valueType tag, which is used to represent the outgoing data type. Since we used Peach to test the network protocol in this test, and the data type is the sixteen string, so we should have a new definition of the valueType attribute. We use valueType to indicate the length of the field. In this way, we can analyze the data received. Data type defined is as follow:

Table 1: Peach Pit value type and realization

Type	Realization
Integer less than 1 byte(8 bits), such as Unsigned4, Unsigned5 and so on	Peach will splice several successively fields which is less than 1 byte into byte.
Unsigned8	unsigned 8 integer(1 byte)
Unsigned16	unsigned 16 integer(2 bytes)
Unsigned32	unsigned 32 integer(4 byte)
Unsigned64	unsigned 64 integer(8 byte)
String	store any length of byte

When Peach parses message, it is based on the length of the valueType in the Peach Pit to fill in the corresponding Blob field in the protocol data unit. For the type of string that cannot be determined for this length, Peach will first calculate the length of the entire Block field based on the value of the Number tag. Then, Peach will fill each Blob field according to the valueType of the Blob attribute in the Block. And the rest of the contents will be filled into the Blob field which type is string.

3.2 Extension of Peach for IKE Protocol Fuzzing

In this paper, we take the IPSec key exchange Progress based on the signature authentication in the main mode as an example, explaining the testing progress of the extended Peach framework for the IPSec key exchange phrase droved by scripts. The progresses of exchanging messages is shown as figure 1.

Fig. 1 The main mode exchange based on signature verification

As description before, HDR means ISAKMP header, and HDR* means the payload after header of ISAKMP was encrypted; SA means sending security association payload; KE means key exchange payload; Nonce means sending Nonce payload, which is necessary in generating the key of encryption and authentication. ID_i and ID_r mean the verified payload for sponsor and responder. Cert-Req is request certification payload. Cert is exchange certification payload.

Prepairation:

Before sending the message, sender and receiver must calculate their own cookies (to protect replay and DOS attack), these cookies are used to identify each negotiation in exchanging message. Because the cookies need to be calculate dynamically in real situation, we increase a new attribute ---*constraint* to define the constraint of this field. Since constraint is defined as the correct value of the field, we will perform the constraint operation on the field after the mutation operation. As for cookie, we invoke constraint="set_random_bytes (8);" in script, which means we can fill the current field by 8 bytes random numbers for each mutant. The set for cookie field in script is as follow:

```
<Blob name="initiator_cookie" value="b6 6c 42 10 12 25 4e c7" valueType="Unsigned64" constraint="
set_random_bytes(8)"/>
<Blob name="responder_cookie" value="00 00 00 00 00 00 00 00" valueType="Unsigned64" />
```

Message *1 and* message *2:*

Communication between the two sides will make a negotiation about security policy. Currently available proposals and transformations are shown in Table 2:

Table 2: IKE proposals and transformations in key negotiation

Proposal	Transformations
encryption algorithm	DES, 3DES, IDEA,Blowfish,etc
hash algorithm	MD5, SHA, Tiger
authentication algorithm	preshared secret key, digital signature , etc
Diffie-Hellman group parameters	Like prime number p and generator g,etc
SA's lifecycle	

For the main mode exchange based on signature verification, we set the following in Peach Pit:

```
<Blob name="encryption_algorithm" value="80 01 00 01" valueType="Unsigned32" />
<Blob name="hash_algorithm" value="80 02 00 02" valueType="Unsigned32" />
<Blob name="authentication_method" value="80 03 00 01" valueType="Unsigned32" />
<Blob name="group_description" value="80 04 00 01" valueType="Unsigned32" />
<Blob name="life_duration" value="00 0C 00 04 08 f0 d1 80" valueType="String" />
```

This configuration shows that encryption algorithm is 3DES, hash algorithm is MD5, authentication algorithm is RSA public key signature, Diffie-Hellman group is 1 namely 768bit and SA's lifecycle is 150000000 seconds. We can use Wireshark [8] to catch the packet, and the result is shown in Figure 2.

```
> Transform IKE Attribute Type (t=1,l=2) Encryption-Algorithm : 3DES-CBC
> Transform IKE Attribute Type (t=2,l=2) Hash-Algorithm : MD5
> Transform IKE Attribute Type (t=3,l=2) Authentication-Method : RSA-SIG
> Transform IKE Attribute Type (t=4,l=2) Group-Description : Default 768-bit MODP group
> Transform IKE Attribute Type (t=11,l=2) Life-Type : Seconds
> Transform IKE Attribute Type (t=12,l=4) Life-Duration : 150000000
```

Fig. 2 Wireshark captured packet of strategy negotiation

Message 3 and message *4:*

Since in message 3 Peach needs to send the cookie provided by message 1 and message 2, there is a function "refer ()" invoked by constraint to quote the assigned field which is received by device under test before.

The configuration of Peach Pit is set as follows:

```
<Blob name="initiator_cookie" value="b6 6c 42 10 12 25 4e c7" valueType="Unsigned64"constraint="refer(
Identity_Protection_Security_Association_Exchange_Request.isakmp.initiator_cookie);" />
<Blob name="responder_cookie" value="a8 a1 be b2 6c 78 96 9e" valueType="Unsigned64"constraint="refer(
Identity_Protection_Security_Association_Exchange_Response.isakmp.responder_cookie);"/ >
```

Identity_Protection_Security_Association_Exchange_Request.isakmp.initiator_cookie and Identity_Protection_Security_Association_Exchange_Response.isakmp.responder_cookie means the the position of quoted field.

Message 5 and message 6:

The main mode based on signature authentication requires to use certificates to obtain public key and authentication. Therefore, we invoke Frankencert as mutation of the certificate field to test this part.

The format of initial certificate generated by Frankencert is PEM, and Peach can't send this kind of certificate. So it needs to make encapsulation on it. First, if the generated Frankencert is a certificate chain, we need to splits them to individual certificate. Then we use Openssl to change the PEM format into DER format which is based on binary code. Finally, we read the content of DER format certificate, change it to hex string and put it in corresponding field at Peach Pit.

4. Experiment and Analysis

To prove the validity of the improvement mentioned in the above section, we write the Peach Pit with XML, the generation based mutation mechanism is employed for the test case generating.

4.1 Experiment Design

As shown in Figure 3, a PC, a router and a PLC which supports IPSec protocol are used in this experiment. Using router is to simulate the real test environment and shield other flow in the net. PC has win7 system with Wireshark installed.

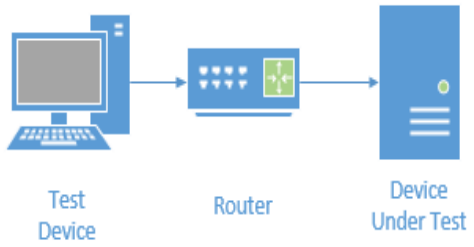

Fig. 3 experiment environment

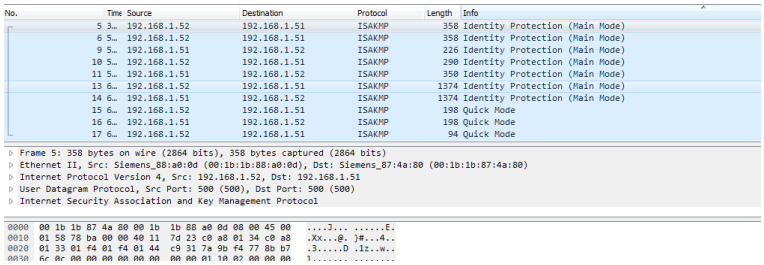

Fig. 4 Wireshark captured information in communication

4.2 Result Analysis

Focus on IKE test, we write some scripts based on different service of IKE. In the process of protocol testing. During the experiment process, a lot of test cases and log items are produced, If the extension of Peach framework for IKE protocol and Peach Pits are right, the console return Finished and the log file will be updated with the Fuzz testing. Also we can capture the communication traffic of the related series. Six service Peach Pit are tested by the extension of Peach framework. We use Wireshark to capture packet in the test. The captured information of CertKeyExchange service was

shown as an example in Figure 4. And the total result is shown in Table 5. The result of the experiment proves that our extension of Peach framework can satisfy the requirement of IKE testing.

Table 3: Test Result

Peach Pit	Cases	Target	Consequence
IKE.CertSAExchange	114973	PLC	Buffer Overflow
IKE.CertKeyExchange	91957		Normal
IKE.PskSAExchange	114973		Buffer Overflow
IKE.PskKeyExchange	40047		Normal
IKE.CertAggressiveSAExchange	198867		Normal
IKE.PskAggressiveSAExchange	146957		Normal

5. Conclusion

With the specific key exchange protocols such as IKE, the Peach Fuzzer cannot directly applied to them on Windows Platform, and some improvements are seriously needed. In this paper, we proposed a fuzzing approach for IKE and described the structure and the workflow. We also evaluated our approach for PLC fuzzing through experiment, and the experiment results proved that the Peach Fuzzer designed can satisfy the requirement of vulnerability detecting of IKE protocol.

References

[1] Doraswamy N, Harkins D. IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks[M]. Prentice Hall PTR, 1999.

[2] Banks G, Cova M, Felmetsger V, et al. SNOOZE: Toward a Stateful NetwOrk prOtocol fuzZEr[C]// Information Security, International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings. 2006:343—358.

[3] Information on:.http://www.peachfuzzer.com/.

[4] Information on:.https://github.com/sumanj/frankencert

[5] Perlman R, Kaufman C. Key Exchange in IPSec: Analysis of IKE[J]. IEEE Internet Computing, 2000, 4(6):50-56.

[6] Chen Y, Su Z. Guided differential testing of certificate validation in SSL/TLS implementations [C]// Joint Meeting. 2015: 793-804.

[7] Brubaker C, Jana S, Ray B, et al. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations [J]. IEEE Security & Privacy, 2014, 2014: 114-129.

[8] Information on: http://www.wireshark.org,2009-06.

[9] Biyani A, Sharma G, Aghav J, et al. Extension of SPIKE for Encrypted Protocol Fuzzing [C]// Multimedia Information Networking and Security (MINES), 2011 Third International Conference on. IEEE, 2011: 343-347.

[10] Dantas H, Erkin Z, Doerr C, et al. eFuzz: A Fuzzer for DLMS/COSEM Electricity Meters [C]// The Workshop on Smart Energy Grid Security. ACM, 2014: 31-38.

[11] Miller B P, Cooksey G, Moore F. An empirical study of the robustness of MacOS applications using random testing [J]. Proceedings of International Workshop on Random Testing, 2006, 41(41): 78-86.

[12] Meijer E, Manolescu D A, Dyer J W, et al. Fuzz testing of asynchronous program code: US, US9015667 [P]. 2015.