# The OpenWRT's Random Number Generator Designed Like /dev/urandom and Its Vulnerability

Dongchang Yoo[1] and Yongjin Yeom[1,2(✉)]

[1] Department of Mathematics, Kookmin University,
77 Jeongneung-Ro, Seongbuk-Gu, Seoul 136-702, Korea
{yoodc8l67, salt}@kookmin.ac.kr
[2] Department of Financial Information Security, Kookmin University,
77 Jeongneung-Ro, Seongbuk-Gu, Seoul 136-702, Korea

**Abstract.** OpenWRT is an open source router firmware based on embedded Linux that uses a dedicated random number generator for the WPA/WPA2 authentication protocol. Its main purpose is to generate a nonce that can be used in a WPA/WPA2 handshake. If the output of the random number generator can be predicted by an attacker, the relevant protocol will not be able to authenticate securely. According to previous studies on Linux, it is well known that a random number generator implemented in an embedded Linux environment does not collect sufficient entropy from noise sources. The lack of entropy increases the potential vulnerability for the random number generator and the Linux protocol. Therefore, we analyzed the WPA/WPA2 authentication protocol and its random number generator. In our results, we point out some potential cryptographic weaknesses and vulnerabilities of the OpenWRT random number generator.

**Keywords:** Embedded linux · OpenWRT · Random number generator · Entropy

## 1 Introduction

In today's world, wireless routers are commonly found in public places, such as Internet cafés and hotel lobbies. Many people join these networks as part of their daily lives. Because wireless routers are inexpensive and can be easily configured without any special knowledge, they are often installed by non-experts. However, an inexpensive device may not contain strong security functions, such as a random number generator. This becomes a major issue when the designer of a cryptographic protocol assumes that the target system is able to generate random numbers with full entropy. In an environment where the target system does not have adequate security functions, there can be significant security breaches, and are most likely caused by weak random numbers.

In this paper, we investigate the structure of the OpenWRT random number generator. Because it is generally implemented in a tiny embedded system, its random number generator lacks the typical noise sources of non-embedded systems. According to a previous research paper [1], the Linux random number generator, both /dev/random and /dev/urandom, may expose some vulnerabilities in an embedded environment like


J.J. (Jong Hyuk) Park et al. (eds.), *Advances in Computer Science and Ubiquitous Computing*, Lecture Notes in Electrical Engineering 421, DOI 10.1007/978-981-10-3023-9_127

the OpenWRT. Embedded systems are inherently vulnerable because of the lack of any human interfaces, such as key-press and mouse movement information. Even worse, embedded systems generally do not contain a spinning disk. These are all used in the Linux kernel random number generator as important noise sources.

We point out an inappropriate design of the random number generator used in the WPA/WPA2 protocol, specifically, entropy collection from noise sources. In addition, we propose a new noise source that can provide additional entropy and be used in a Linux environment.

This paper covers the following topics in the following sections:

- The security analysis of the WPA/WPA2 protocol is described in Sect. 2. In Sect. 2.1, we identify a potential weakness with an attack scenario originated by sending a nonce without encryption.
- The specification and security analysis of the random number generator used in the WPA/WPA2 protocol is covered in Sect. 3.

The preliminary work on this topic is presented in [2]; this paper is a comprehensive and extended version. We delve into more detail of this subject and extract and analyze experimental data harvested from the monitoring function we inserted in OpenWRT for logging the actual noise source of the random number generator.

## 2 WPA/WPA-PSK Authentication Protocol

When a smart device user requests a connection to the Internet via a wireless router, a mutual authentication is required before the router provides Internet service. The OpenWRT packaged daemon, hostapd, facilitates the cryptographic function including user authentication. In fact, the WPA/WPA2 protocol implemented in hostapd works this way. There are two versions of WPA/WPA2 - PSK, and Enterprise. The WPA/WPA2-Enterprise version includes an additional authentication server, a radius server, and provides a well-designed, strong authentication. Therefore, we will focus on the vulnerability of WPA/WPA2-PSK.

### 2.1 WPA/WPA-PSK

The authentication protocol WPA/WPA2-PSK [3] implemented in the hostapd enables a client and a router to authenticate each other through a 4-way handshake. Before handshaking, they must pre-share a single password created using English alphabet characters, numbers, or a combination of both. WPA/WPA-PSK performs mutual authentication by having both the router and client calculate a PMK (Pairwise Master Key) using a PSK (Pre-Shared Key). Next, a router and client send a message encrypted with the symmetric key that is shared using a 4-way handshake. The steps of the WPA/WPA2-PSK authentication are described in Fig. 1.

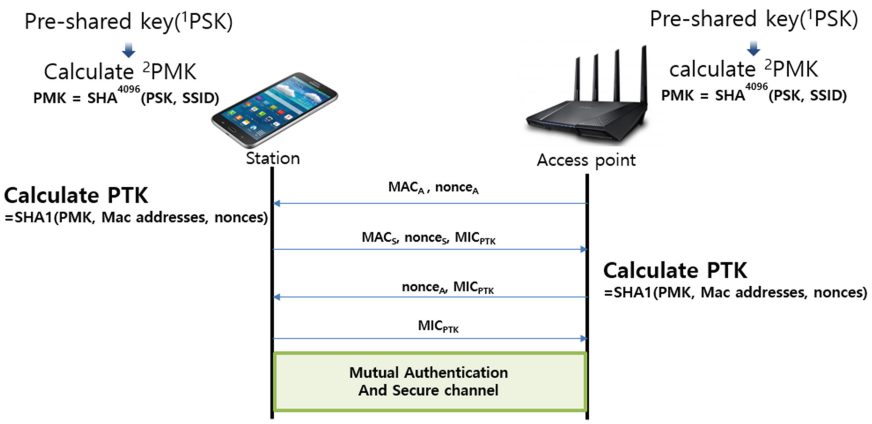


**Fig. 1.** The WPA/WPA2-PSK protocol with 4-way handshake. In this figure, the left side entity is called a station, or client. The right side entity is the wireless router.

## 2.2 Cryptographic Analysis of WPA/WPA2-PSK

The WPA/WPA2-PSK protocol has well-known cryptographic vulnerabilities. This protocol was designed under the assumption that all users having the password are honest. In Internet cafés, hotels, and public places, all the customers and visitors can obtain the password easily, including any potential attackers. Therefore, our attack model assumes that an attacker knows the correct password. Furthermore, this attacker can calculate the encryption keys of other users who have connected with the router because all the parameters for deriving encryption keys are public, except for the password. Unfortunately, it is infeasible to overcome this vulnerability. We can only propose that the WPA/WPA2-PSK protocol is no longer used in public places. Otherwise, the router administrator must check face-to-face to determine whether the user is an attacker (Fig. 2).


**Fig. 2.** An example of poor WPA/WPA2 password management in a public place. The password is posted openly on the table and is too simple.

Another security issue is that a nonce is sent openly in the WPA/WPA2-PSK protocol. In general, a nonce is not dealt with as a secret parameter; it only needs to vary over time. However, if the distribution of a nonce is biased, or is predictable, then the protocol cannot achieve the desired security. The WPA/WPA2-PSK protocol cannot avoid this issue when implemented in an embedded environment.

## 3 The Random Number Generator of the WPA/WPA2

The Linux environment contains a kernel level Random Number Generator (RNG), also known as /dev/random and /dev/urandom. However, research has shown that the Linux kernel RNG is vulnerable when used in an embedded Linux system [4]. The WPA random number generator was installed and used in the WPA/WPA2 protocol in order to provide more entropy and make up for the potential weakness of the Linux kernel random number generator. We analyzed the source code of hostapd-2016-01-15 in OpenWRT Chaos Calmer 15.05.1, the latest, stable version [5]. Consequently, we found this random number generator implemented in the path ../src/crypto/random.c.

How much more secure is the random number when using an additional RNG?

### 3.1 Specification of WPA's RNG

The WPA random number generator has three components - noise collection, number generation, and a single entropy pool. All components are similar to the corresponding parts of the Linux kernel random number generator.

Noise sources are mixed into the entropy pool using the mixing function. Notice that the noise source is independent of the Linux kernel random number generator. When a user or program asks to generate random numbers, the extraction function generates a random number using the entropy pool in secret. The following figure presents the structure of the WPA random number generator process (Fig. 3).

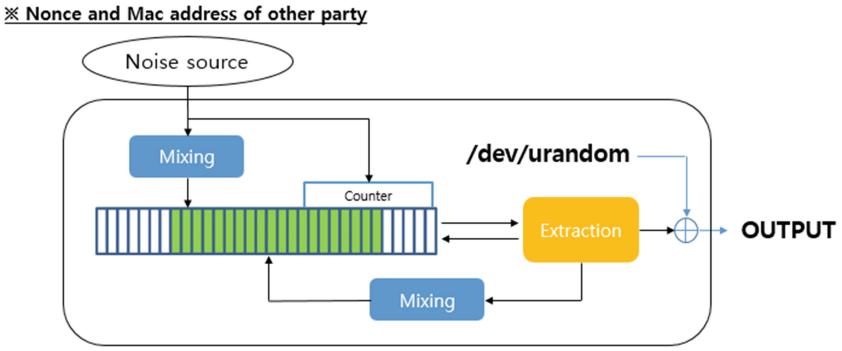


**Fig. 3.** WPA's random number generator structure. Notice that the random number generator provides the final random numbers based on an output of XOR operation of extraction function and the /dev/urandom device.

### 3.1.1 Structural Differences Between WPA's RNG and Linux Kernel RNG

The WPA's random number generator was implemented based on the source code of the Linux kernel random number generator [6]. After analyzing the source code, we discovered that the two source codes are almost identical. In this section, we explain the primary differences between the two RNGs.

The WPA's RNG has a single entropy pool while the Linux kernel RNG has 3 entropy pools - 1 input pool and 2 output pools. The output pool is called a blocking pool and is used when creating cryptographic random numbers, as long as the entropy of the input pool is sufficient. Otherwise, output is blocked until enough entropy is accumulated. The other output pool is connected to /dev/urandom and always produces a random number for the user. Therefore, when the entropy of the input pool is insufficient, the random numbers generated are potentially vulnerable. The WPA's random number generator works like /dev/urandom. Despite having little entropy, it creates random numbers without blocking.

A second point is a difference between primitives of each extraction function. A keyed hash function is used in the Linux kernel random number generator while using the SHA-1 cryptographic hash function [7] in the Linux kernel random number generator. We can see that the designers have tried to increase the security of the WPA protocol through the WPA random number generator.

## 3.2 Security Analysis of the WPA Random Number Generator

We observed the collection process of the WPA random number generator by adding a monitor function to its source code. We then checked the type and quantity of noise sources. The WPA random number generator uses a nonce that is generated by itself, or sent by the other party in the WPA/WPA2-PSK protocol with the opposing MAC address. After collecting data from a noise source, the noise and the time it is collected are mixed into the entropy pool. Because an attacker continuously observing the protocol session can check all entropy sources and the time when the nonce is sent, additional time complexity will increase only slightly. In conclusion, the WPA's random number generator does not increase the security strength of random number generation. That is, using the Linux kernel random number generator with an additional implementation, the WPA's random number generator is not significantly different cryptographically (Fig. 4).

```
   IEEE 802.1X: version=1 type=3 length=121
WPA: Received Key Nonce - hexdump(len=32): 2d 5f 6a ff 57 fa bf d1 00 b9 05 cd e8 cb b9 a4
cd 20 34 80 a4 07 6f e1 7f 08 07 bf 1e fc 08 1a
WPA: Received Replay Counter - hexdump(len=8): 00 00 00 00 00 00 00 01


Im in Adding Function
[Adding]Noise:  - hexdump(len=32): 2d 5f 6a ff 57 fa bf d1 00 b9 05 cd e8 cb b9 a4 cd 20 34
80 a4 07 6f e1 7f 08 07 bf 1e fc 08 1a
[Mixing]Input Data:  - hexdump(len=8): 16 c5 66 57 4d 8b 08 00
[Mixing]Input Data:  - hexdump(len=32): 2d 5f 6a ff 57 fa bf d1 00 b9 05 cd e8 cb b9 a4 cd
20 34 80 a4 07 6f e1 7f 08 07 bf 1e fc 08 1a
Entropy Counter : 1
```

**Fig. 4.** Partial log files collected show the noise source of the WPA/WPA2 random number generator (*underlined hex values*). After a router receives a nonce from a client, it accumulates the nonce in an entropy pool. This operation is invoked just once per client connection.

## 4 Conclusion

We analyzed the source code of the WPA random number generator working in the hostapd of OpenWRT to determine its structure. Modifying the program to add noise source monitoring, we could derive real noise sources and determine its collection frequency and quality.

The designer of the WPA/WPA2-PSK protocol tried to enhance the security of the random number generator, but it does not make it more difficult for an attacker to gather random number information than by using the Linux kernel random number generator exclusively. Even though the two environments are different, both were designed like the Linux kernel RNG. This is the reason the random number generator fails to enhance the security. Consequently, we propose that a random number generator be designed with consideration of the environment and recommend that it use more varied noise sources.

**Acknowledgments.** This research was supported by Next-Generation Information Computing Development Program Through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648).

## References

1. Gutterman, Z., Benny, P., Tzachy R.: Analysis of the linux random number generator. In: 2006 IEEE Symposium on Security and Privacy (S&P 2006). IEEE (2006)
2. Dongchang Y., Yongjin Y.: Security analysis of random number generator in OpenWRT access point. Korean Inst. Commun. Inf. Sci. (2016)
3. IEEE Std 802.11i-2004. http://standards.ieee.org/getieee802/download/802.11i-2004.pdf
4. Mowery, K., et al.: Welcome to the entropics: boot-time entropy in embedded devices. In: 2013 IEEE Symposium on Security and Privacy (SP). IEEE (2013)
5. OpenWRT Chaos Calmer 15.05.1. released: Mon, 16 March 2016. https://downloads.openwrt.org/
6. Lacharme, P., Rock, A., Strubel, V., Videau. M.: The Linux Pseudorandom Number Generator Revisited. depose sur Cryptology ePrint Archive (http://eprint.iacr.org/) (2012) < hal-01005441>
7. PUB, FIPS: Secure hash standard. Public Law **100,** 235 (1995). Distributed