LETTER

Implementing Signature Based IDS in IP-Based Sensor Networks with the Help of Signature-Codes**

Syed Obaid AMIN†, **Muhammad Shoaib SIDDIQUI**†, *Nonmembers*, **Choong Seon HONG**†*a), *Member*, *and* **Sungwon LEE**†, *Nonmember*

SUMMARY A dynamic coding mechanism is presented to implement distributed signature based IDS (Intrusion Detection System) in IP-USN (IP-based Ubiquitous Sensor Networks). The proposed scheme allows the creation of a lightweight IDS in terms of storage, messaging and energy consumption, which make it appropriate for resource constrained sensor devices.

key words: *IP-USN, IDS, sensor networks*

1. Introduction

Utilization of IP in USN (Ubiquitous Sensor Networks), called IP-USN, converges them to a unified and simple naming and addressing hierarchy. This also allows us to get benefits from variety of tools available already for configuring, managing, commissioning or accounting of the IP networks. However, this integration of IP and USN also combines weaknesses of both worlds. Along with conventional sensor network attacks, single packet attacks which were only possible in IP networks are now possible in sensor networks as well. For example, Ping of Death attack, in which a large ping-packet is sent to crash the receiver. For detecting single packet attacks signature based IDS (Intrusion Detection Systems) are often found useful; because they do not require several packets to detect an attack as compared to anomaly based IDS. In signature-based IDS, we first define intrusions ahead of time in terms of signatures or patterns and then observe the occurrence of these signatures in the system under observation. As the number of patterns could range up to thousands, pattern matching consumes not only the storage but also the most of the CPU cycles to execute the complex pattern matching algorithms [1]. Therefore, so far there is no signature based IDS which can work on resource constraint sensor nodes. In this paper, we present a coding mechanism so that signature-based IDS can be realized for IP-USN.

Manuscript received March 3, 2009.
Manuscript revised August 25, 2009.

†The authors are with the Department of Computer Engineering, Kyung Hee University, Korea.
*Dr. Choong Seon Hong is the corresponding author.

**This research was supported by the MKE, Korea, under the ITRC support program supervised by the IITA (IITA-2009-(C1090-0902-0016)).

a) E-mail: cshong@khu.ac.kr

DOI: 10.1587/transcom.E93.B.389

2. Relevant Work

The patterns or signature of an intrusion can be represented in many ways. In this paper we consider a signature-based IDS based on string matching algorithms [1]. Most of the commercial string-based IDS use automata theory or FSM (Finite State Machines) for signature matching. However, these solutions due to their high demand of resources are not practical for sensor networks. In [1] authors presented a fast and scalable pattern matching scheme using Bloom filters [2]. Bloom filter is a randomized data structure, represented in a form of bit vector. It is used for membership queries that whether an entity in question was part of the data set or not [1], however with controllable false positive probability.

Our scheme, also relies on Bloom filters and therefore has some similarities with this scheme. However, scheme in [1] was proposed for a single machine and all traffic passes through that machine. Moreover, in [1] authors used separate hash table for Bloom filter match verification, which is required due to false positive probability associated with Bloom filter [2]. However, in sensor networks many nodes are working as relay nodes. And due to resource limitations, it is unwise to place hash table at every relaying node. Therefore, this approach is not practical. On the other hand, if distributed approach is used to implement this method such that hash table and Bloom filters reside on separate machines (at the Sink and sensor node, respectively) then entire payload of the packet would have to be sent to the Sink as well. This condition lessens the efficiency of this approach because sensor needs more energy in transmission as compared to calculation [3].

3. Proposal

To overcome the aforementioned problems we propose the concept of signature-code, a dynamically created attack-signature identifier which allows us to represent an attack-signature in only a few bytes. Therefore, instead of sending a complete payload we only send a signature-code for Bloom filter match verification to the Sink. Consequently, we not only save the storage of sensor nodes but also the energy required for transmitting long bit patterns. Thus, with the help of signature-code a lightweight signature-based intrusion detection system can be realized.

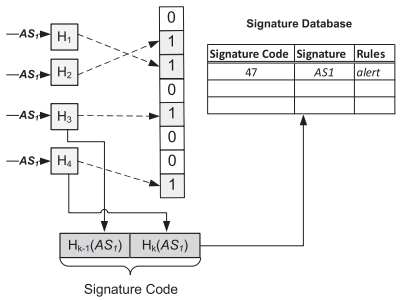


Fig. 1 Generation of signature-code for the signatures having same length.

3.1 Signature-Code

The signature-code is an $r * \theta - bit$ number which is formed by concatenating the output of the last θ hash functions. Where θ is the number of hash functions used to generate the signature-code. Let S_i denotes the signature-code for attack-signature i. Then according to the definition, for $\theta = 2$, S_i can be given by $S_i = H_{k-1}(i) \odot H_k(i)$, where $\odot$ represents string concatenation. Figure 1 depicts generation of signature-codes. In Fig. 1, the attack-signature AS_1 is passed through four hash functions which produce the $r-bit$ output and set the corresponding bit to 1. For example, the output of the hash function H_3 and H_4 is 100 and 111 (in terms of bits), respectively. Therefore, the bits on the 4th and 7th position (indexed from 0) are then set to 1. The outputs of the hash functions H_3 and H_4 are also used to generate the signature-code. In this case, the signature-code of AS_1 is 47.

We used signature-code to dynamically locate the attack-signatures and their associated rules in the signature database, located at the Sink. As shown in Fig. 1, the signature database contains attack-signatures and associated rules, which are indexed by signature-codes. It is possible that due to hash collision we may experience a signature-code collision. However, it is only possible when all of the hash functions, involved in a signature-code generation, experience a collision. Even if we have only one hash function which did not experience a hash collision, we would have unique signature-code.

Hash functions involved in signature-code generation must be lightweight. This requirement is not impractical because we do not require any cryptographic hardness properties. Therefore, simple integer hash functions can be used for this purpose. Examples of such hash functions can be found in [4].

3.2 Description

In first phase of our scheme, signatures of different length are passed from the Bloom filters. As a result, Bloom filters give us a bit array representation of the input signatures and signature-codes. The programmed Bloom filters are then placed at every sensor node. Once there is a Bloom filter match, sensor node sends the alert signal containing packet payload and signature-code to the Sink for the verification of it.

The Sink, in response lookup the received signature-code and payload in its signature database. If a match is found, it sends the rules associated with the signature such as drop, notify, log etc. to the sensor node, which applies it on the packet in question. The scheme can be optimized by storing the downloaded rules in a small cache for later use; so that extra messaging can be avoided for similar packets.

Since attack-signatures are rarely found in the packets, the swift check in Bloom filter avoids not only the unnecessary messaging but also the unneeded searching of attack-signature in whole signature database. Furthermore, in sensor network, if we do not identify the attack-signature exactly then there would be a huge overhead of messaging to avoid the compromised nodes. Therefore, with the help of signature-code we try to identify exact attack-signatures so that extra messaging overhead can be minimized. Sending of signature-code instead of entire payload also lessens the bits to be transmitted. This factor is mainly helpful in extending the lifetime of the sensor networks as shown in Evaluation section. In addition, the compressed representation of attack-signatures in a form of bit vectors also reduces the storage requirement on sensor nodes. Although this saving of space is inherited advantage of Bloom filters; however, application of Bloom filters in sensor network in the context of intrusion detection has never been addressed before.

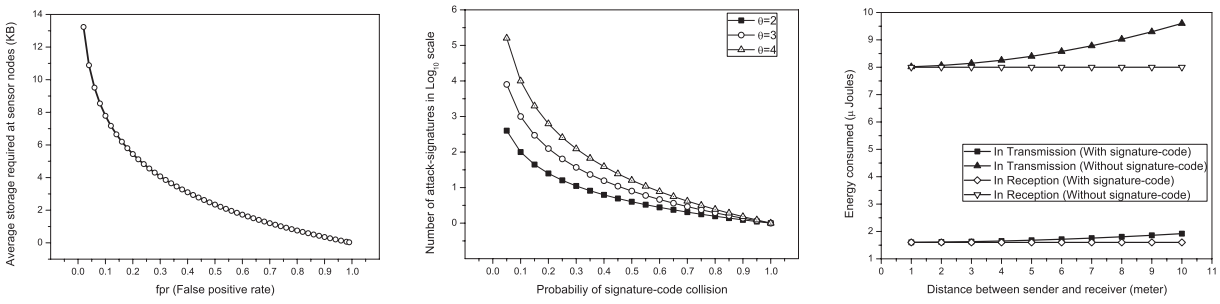
4. Performance Evaluation

Currently, IP-USN is in its evolutionary stage. Thus, at the moment it is hard to realize any attack-signature in IP-USN traffic. Therefore, to evaluate the performance of our scheme we use Snort signature-set [5]. In current signature-set of version 2.8 of Snort, there are 13,339 attack-signatures. Given the current limited set of applications for IP-USN, this signature-set can easily be regarded as the upper bound for number of signatures in IP-USN.

The performance of our proposal mainly depends upon the size of the bit array m, number of attack-signatures n and number of hash functions k. We start our evaluation with the analysis given in [2]. The *fpr (false positive rate)* of Bloom filter can be given by 1 as shown in [2].

$$fpr = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{\frac{-kn}{m}}\right)^k \quad (1)$$

Further simplification shows that Bloom filter uses $1.44 \log_2(1/fpr)$ bits of space per inserted attack-signature [2]. This means that the 13,339 attack patterns available in current release of Snort would require approximately 252 KB of storage without our scheme. Figure 2(a) shows the number of bits required to store current Snort signature-set by using our scheme. For example, with $fpr = 0.024$, our scheme on an average requires 13 KB ($\approx$ 8 bits/signature) as compared to 252 KB to represent the



(a) (b) (c)

Fig. 2 (a) Size of m for representing current Snort signature-set with respect to fpr, (b) Expected number of signatures that can be added without experiencing a collision, and (c) Comparison of energy consumption with and without signature-code.

whole signature-set. Approximately, we require 95% less space than normal storage requirements. To calculate the probability of signature-code collision, assume that p_{H^i} is the probability of experiencing a collision in hash function H_i. Then the probability that a signature-code can also experience a collision is $p_s = \prod_{i=1}^{\theta} (p_{H^{k-i}})$; where, p_{H^i} can be given by Eq. (2):

$$p_{H^i} = 1 - (1 - \varphi_i) \cdot (1 - 1/m)^n \tag{2}$$

Here, φ_i is the probability of hash collision due to its internal operations. Furthermore, if X is the number of signatures added up to and including the first signature that experience a collision then the probability $P_X(x)$ of experiencing a collision in a signature-set can be given by $P_X(x) = p_s \cdot (1 - p_s)^{x-1}$. Whereas, the expected number of signatures which can be added without experiencing a collision can be given by $E[X] = 1/p_s$. Figure 2(b) shows the expected number of signatures that can be added without experiencing a collision with increasing probability of signature-code collision and different values of θ in Log_{10} scale. For example, for $p_H^{k-1} = p_H^{k-2} = 0.01$ and $\theta = 2$ the probability that signature-code will also experience a collision is 0.0001. Theoretically, this shows that approximately 10,000 signatures can be added without experiencing a collision. However, when implemented, we did not observe even a single signature-code collision in 13,339 signatures long signature-set of Snort [5].

The maximum length of an attack-signature in current Snort's signature-set is 487 bytes; whereas, the average length of an attack-signature is 20 bytes. If we assume that the incoming packet only contains attack-signature of the length of 20 bytes then without our scheme 20 bytes are required to be sent to the Sink for searching a rule in signature database. On the other hand, our scheme requires only 4 bytes with $\theta = 2$ to search a rule in signature database, even for the biggest signature in Snort signature-set. Figure 2(c) shows the differences in energy consumption in single-hop transmission and reception of a message in both of these schemes according to the radio model given in [3]; with average attack-signature length of 20 bytes, $\theta = 2$ and $r = 16$.

5. Conclusions

In this paper we propose a way of implementing signature based IDS in IP-USN environment. We introduced the concept of signature-code to reduce the storage requirements and messaging overhead. In addition, with the help of signature-codes we can greatly enhance the lifetime of a sensor node.

References

[1] S. Dharmapurikar and J.W. Lockwood, "Fast and scalable pattern matching for network intrusion detection systems," IEEE J. Sel. Areas Commun., vol.24, no.10, p.1781, 2006.
[2] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet Mathematics, vol.1, no.4, pp.485–509, 2004.
[3] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," Proc. 33rd Annual Hawaii International Conference on System Sciences, p.10, 2000.
[4] D.E. Knuth, The art of computer programming, vol.3, Reading, MA, 1973.
[5] M. Roesch, "Snort-lightweight intrusion detection for networks," LISA'99: Proc. 13th USENIX Conference on System Administration, USENIX Association, pp.229–238, Berkeley, CA, USA, 1999.