A Static Code and Dynamic Data Attestation based Intrusion Detection System for Wireless Sensor Networks

Neelam A Surti
Department of Computer Engineering
C. K. Pithawalla College of Engg. and Technology,
Surat, India

Devesh C Jinwala
Department of Computer Engineering
S. V. National Institute of Technology,
Surat, India

ABSTRACT
The Wireless Sensor Networks (WSNs) have applications typically in ubiquitous and pervasive environments that make ensuring the security therein critical. Despite deployment with utmost stringent security measures, the intrusions and the adversarial attacks like node compromise and node tampering cannot be prevented. Hence, there is a need for devising an intrusion detection and prevention system that can withstand the resource constraints and work feasibly within the same. One such intrusion detection technique is code attestation which is useful for verifying the program integrity of nodes in such networks. Our focus here is on software based remote code attestation. The static code attestation techniques published in the literature only check the integrity of the static code embedded within sensor nodes whereas the dynamic data attestation techniques check the structural integrity of dynamically created data. We believe that an integrated approach that uses both the static and dynamic code attestation techniques can leverage the effectiveness of an intrusion detection system. In this paper, we propose our integrated approach for countering attacks based on code attestation. As we demonstrate using our experimental simulation studies, with the marginal increase in memory and computational overhead, our approach ensures improved overall security. To the best of our knowledge ours is the first attempt in following such an approach.

General Terms
Networks, Security, Algorithms, Attacks.

Keywords
Wireless Sensor Networks, Intrusion Detection System, Code Attestation, Pseudo Random number Generator.

1. INTRODUCTION
Wireless sensor Networks (WSNs) consists of small and low cost sensor nodes logically interconnected to each other through a wireless radio to form a network that senses and processes a physical parameter in the real world. Even with the limited resource availability in general and scarce computational power in particular, the WSNs are deployed in versatile applications in diverse areas ranging from *environment monitoring, defense, industrial process monitoring and control, homeland securities and many more* [1]. However, being deployed ubiquitously, the communication security issues in WSNs are critical and have to be carefully examined.

Devising the security protocols for WSNs is as such challenging and non-trivial. This is so because one hand the traditional security protocols entail heavy resource overhead whereas, on the other hand, the scarce resource availability on the sensor nodes makes it difficult to implement the conventional security protocols on the network nodes. Nevertheless, even if security protocols tailored to the sensor nodes were devised and implemented, as is true with security implementations in general, there is a need to monitor the network for any unwanted intrusions. Intrusion detection is a set of actions that analyses and reports unauthorized activities [30]. Detecting a physical layer attack of node compromise is very crucial as the upper layer attacks can be introduced, with the node's memory contents being tampered.

There are numerous approaches proposed in classical network security literature that implement network intrusion detection [31]. One of these is a signature based approach wherein the patterns for known attacks are compared with the current events for intrusion detection. As compared in anomaly based approach, the regular network behavior is studied and any deviation from the regular behavior is used for detecting the intrusion. In general, signature based [2],[4] approaches like rule based traffic analysis, pattern matching are used for intrusion detection. As compared, the anomaly based [3] approaches use Finite State Machines [5], machine learning techniques [6]. Few of the approaches have indeed been adapted for the WSNs which are further discussed in section 2.1.

However, amongst all of these, *code attestation based intrusion detection* [8],[9],[20] has been finding significant attention. Using code attestation, a compromised node with embedded malicious code within its memory can be detected. Thus, code attestation can be used to verify the integrity of memory within a sensor node. Here we focus on software based code attestation techniques.

Software based code attestation can be categorized into static code based and dynamic data based attestation. Static code based attestation [8],[18],[21] checks the integrity violation of the static code embedded within sensor node using cryptographic checksum technique. As compared in dynamic attestation [9],[20] the dynamically created data at runtime is considered for attestation. This enables verification of the structural integrity violating parameters of these data.

Using static code attestation techniques integrity of program code within sensor node can be attested but such sensor node are still vulnerable to attacks which are created at run time. There are some attacks on sensor nodes which are created by utilizing the existing program code of it, like return-into-libc [38] and ROP [35] attacks.

In return-into-libc [38] attack the existing code in the program memory is reused by manipulating the stack to call an existing library function. This technique violates the normal flow of the program by calling library function, but it is not useful to create a specific task.

While in ROP [35] attack the vulnerabilities within the program code is utilized to create gadgets (pieces of codes within the program memory ending with ret instructions). Multiple gadgets are combined in various patterns to create specific operations as specified in [35] which are useful for creating attacks. Some of the techniques through which ROP attacks are introduced in network are stack smashing [36], buffer overflow [39] and mal-packet injection [37] within network.

We observe from the literature that there is as yet an attempt lacking that demonstrates the feasibility of an integrated approach for code attestation. Therefore, in this paper we propose our approach that combines the static code and dynamic data attestation techniques. We show that this approach can withstand various attacks like pre computation of checksum, return oriented programming attack, memory copy attack and buffer overflow attack. This advantage can be realized at a small increase in overhead in memory and energy consumption as compared to the same without using any attestation. To the best of our knowledge ours is the first approach in combining the static code and dynamic data attestation techniques.

The rest of the paper is organized as follows. In section 2 we introduce the intrusion detection system in wireless sensor network as well as literature survey related to code attestation in the area. In section 3 we cover our assumptions and threat model for the proposed algorithm. In section 4, we discuss the approach for combining static code and dynamic data techniques. Section 5 covers the methodology of implementation. Section 6 covers analysis and performance results. Lastly in section 7, we conclude with future scope of the work ahead in the area.

2. THEORITICAL BACKGROUND

WSNs are vulnerable to many security attacks as it uses open-to-all wireless communication for information exchange which is an unsecure transmission media. Furthermore, WSNs are deployed in intimidating environments where it is difficult to provide physical protection. Hence, security at different layers is desired in protocol stack design for communication.

2.1 Intrusion Detection System for Wireless Sensor Networks

Intrusion detection systems (IDS) are processes that check the normal flow of a system or a network and notifies if some violations are there due to unwanted activities within it [30]. Few of the IDS adapted in WSNs are cited here, signature based approach using spontaneous watch-dog [11] and the Received Signal Strength Indicator value used to detect intruder as in [5]. In [6] node impersonation attack and route depletion attacks are detected using an intrusion detection algorithm with sliding window and packet buffering. Intrusion detection algorithm based on estimation of the network flow information [10] in the attacked area is used for intruder identification. Game theory based approaches [15] and machine learning based approach using automata based learning [16] is used for intruder detection. Code Attestation based IDS [8], [9], [18] is used at physical layer for node compromise detection.

2.2 Code Attestation Techniques

WSNs are having applications in ubiquitous and unattended environments for han-dling various events like temperature monitoring etc. Security is crucial in such envi-ronments as a node within the network can be tampered which lead to an attack, may on the whole network by disclosing the node's confidential information. To detect compromised nodes, code attestation techniques are used for verifying the program integrity on the sensor nodes in WSNs. There are two approaches proposed for the same – Hardware based code attestation and software based code attestation. Memory traversal for checksum computation can be pseudo random cell based, block based or sequential.

2.2.1 Hardware Based Code Attestation.

Hardware based code attestation protocols includes a tamper resistant hardware which uses Trusted Platform Module (TPM) [11] to validate the system integrity. Periodic Broadcast Attestation Protocol(PBAP) and Individual Attestation protocol(IAP) [12] are hardware based code attestation protocols in which cluster nodes verify the system integrity of cluster head at regular interval in PBAP and at any time in IAP. TRAP [13] justifies that TPM can be included in all the sensors and not just in the cluster heads.

2.2.2 Software Based Code Attestation.

Instead of adding extra hardware like TPM chips, in software based code attestation memory checksum is computed to verify the system integrity. All the existing soft-ware based attestation techniques are based on a challenge-response protocol where the verifier challenges a prover to compute a checksum of its program memory.

Software attestation can be categorized as static code attestation [8],[15],[16],[17],[18,[19] which checks the static code embedded within the program address space of a sensor node and dynamic data attestation [9],[20] which checks the dynamically generated data while in program execution. Multiple nodes are involved in attesting a single node in distributed attestation [21]. From the various techniques available for attestation our focus is on software based remote code and data attestation.

2.3 Related Work

SWATT (Software-based ATTestation) [8] is software based static code attestation protocol which relies on the time bound of the attestation response. Software based remote code attestations techniques as proposed in [15],[16],[17],[18] includes the static memory traversal or pseudo random cell based memory traversal for computing the checksum of only static memory contents. Checksum computation is not useful to check the integrity of memory locations with dynamic data like stack and heap storage. In DataGuard [9] data are protected by guard values and on corrupting the guard values an intruder can be detected. While ReDAS [20] uses the structure integrity violation of the system properties of dynamically created data for intrusion detection.

3. ASSUMPTIONS AND THREAT MODEL

3.1 Assumptions

We assume that the Base station working as verifier is aware of hardware con-figuration, static memory image and other

vital information of sensor node to be verified. We assume that sensor node are equipped with microcontrollers with Harvard architecture were only program memory is executable and data memory is non-executable. Free memory space within program memory is filled with pre-deployment random noise. We have taken non-optimized application codes for experimentation.

3.2 Threat Model

We assume that the attacker has full control over the sensor node and can read and write any memory location of the sensor node. Extending or modification of hardware is not allowed to the attacker. Also the attacker cannot insert powerful laptop class machine within the network.

We have created Return Oriented Programming (ROP) attack using buffer overflow vulnerability on sensor motes for testing our approach. In [32] one attack is described for hacking the sensor motes using existing functions of the program code. In our attack model we used inline assembly codes for getting the stack pointer values and frame pointer values. These values are used to insert into system stack at the time of buffer overflow so that the control flow of the program is changed. Detail description is specified in [33] and for experimentation we changed the control flow of the program to call an existing routine code while sustaining the program execution.

4. THE PROPOSED APPROACH

When some new updated modules are to be inserted within WSNs, then code updating was essential to make it adaptable. Due to self-updating process included in AVR microcontrollers, malicious code may propagate from the writable data address space to executable program address space. Most of the code attestation techniques are attesting the program memory to verifying the program integrity of the embedded code. These techniques are failing to withstand against ROP [22] attacks were it alters the control flow of the running program without modifying the program memory. Hence it is needed that attestation routine include both the program memory and data memory as well at the time of attestation.

As shown in Table 1 the checksum is computed taking a randomly generated number m as challenge from the Base Station working as verifier. Using the challenge as a seed to the RC4 stream cipher used as a Pseudo Random number Generator (PRG), the prover travels the memory location and update the checksum vector C.

Dynamic objects which are changing frequently are not considered in static code attestation like the stack and heap contents. Hence runtime threats like buffer overflow attacks that modify the dynamic objects cannot be detected. Using dynamic data attestation, hash computation of dynamically changing objects is also incorporated. In one of the dynamic data attestation technique, DataGuards[9] runtime program data are tracked for any superfluous manipulation. Basically an executing statement only affects some fixed number of data objects, but by enforcing some of the attacks like stack buffer overflow which can change the return address and can modify the function pointers, runtime data objects are modified. In Data Guard the data boundary integrity is checked were a program variable should not affect anything outside its boundary limits. In data guard technique, from an input program which is about to execute the number of data objects are categorized into local, global and heap data objects. Additional data guard variables are created around the data objects to check their boundary integrity. Detail description of generating guard elements is provided in DataGuards[9].

Table 1. Algorithm for static code attestation using Pseudo random memory traversal

Algorithm Compute_Checksum(n)

//Input: n initial nonce value is the challenge sent by the verifier

//Output: Checksum of memory

Let C be the checksum vector and j be the current index into the checksum vector, m is the maximum memory location available, mm = m*ln(m)[due to Coupon collector's problem]

for i=1 to mm do

Ai = PRG(n)[1-m]

//Update checksum byte

Cj = Cj + (Mem[Ai] (exor) C((j−2) mod 8))

Cj = rotate left one bit(Cj)

//Update checksum index

j = (j + 1) mod 8

return C

Here it is assumed that the data guard variables are already inserted within the program using the criteria specified in [9]. Table 2 shows the hashing computed on the data guards using the nonce n sent by the verifier for attestation.

Checksum generated using static code attestation is appended with hash value generated over data guard values as described in the combined algorithm in Table 3. The reason behind combining both these techniques is to provide protection against static code and dynamic data attacks. The verifier needs to get both the combined value as response as the values are generated atomically so there is no way to change the code for in-between computation of both the techniques.

Table 2. Algorithm for Data Guard Assignment [9]

```
Algorithm data_guard_assignment(int seq)
//seq the sequence to generate the data guard and nonce is the
change given by the verifier
if(seq == 0) then
  {e,nonce} = get_secret_data_from_verifier();
  Seq++;
  V = hash(e,nonce,seq);
  Erase(e,nonce);
  Set_data_guard_value(v);
End
Else
  V = readout_last_data_guard_value();
  Seq++;
  Update_last_data_guard_value(hash(v,seq));
  Set_data_guard_value(hash(v,-seq));
End
```

As far as data memory is concerned the data lying on that memory contents are unpredictable as they are dynamic data created and collected by running application in program memory. SMARTIES [24] uses the technique of filling the empty data space with randomness at the time of attestation, but the data overhead increases largely between the challenger and the prover to withstand the attack like Time Of Use to

Time Of Attestation (TOUTOA). Hence, the dynamically created data can be guarded using DATAGUARD [9] to provide protection against buffer overflow attack. Other mechanisms used to provide protection against ROP attack is using STACKGUARD [25] and StackShield [34] but even these tools are not secured against ROP attack. One of the vulnerability with the STACKGUARD [25] as specified by [26] is double memory corruption through which the canary words are not corrupted but the return address is manipulated without detection.

Thus we can say that program memory and the data memory both are to be checked at the time of attestation. As data memory contents are not predefined we cannot use checksum method to attest it. But by checking the control flow integrity or by guarding the data by canary words we can provide the attestation of data memory.

Table 3. Algorithm for Combined Static Code and Dynamic Data Attestation

Algorithm combine_static_code_dynamic_data(challenge C)
//Challenge C sent by the verifier
Checksum = Compute_Checksum(C);
Hash_val = hash(data_guard_assignment(C);
Result =
First_Four_most_significant_bytes(Checksum)+First_Four_least_significant_bytes(Hash_val);
return Result

5. SECURITY ANALYSIS

In this section we consider the resistance of our attestation protocol against pre-computation of checksum, TOUTOA, Memory copy, buffer overflow and ROP attacks.

Pre-computation of checksum: Pre-computation of checksum attack is created if the attacker compromises the sensor node and stores all the possible values of checksum before inserting the malicious code. As shown in Figure 1 it uses Pseudo random memory traversal were RC4 stream cipher is used as PRG (Pseudo Random Generator) which utilizes the challenge sent from verifier as a random seed value. Hence, computing checksum in advance is not possible.

TOUTOA attack: In this attack, the attack uses the sensor node as a compromised node and at the time of attestation, the attack may reset the setting so that node is working as a genuine node. Here we have considered data guard values with each data elements as shown in Figure 2 in which one way hash function SHA-1 is used to create guard elements which are once corrupted, cannot generate the correct answer at later attestation time. Hence, TOUTOA attack can be prevented by our attestation protocol.

Memory copy attack: In this attack, the original program code is copied at another location in memory and the malicious code is copied at the original code. Otherwise the original code is at its place only and malicious code is copied into program memory. This attack is prevented as we are filling the empty space in program memory with pre-deployment random noise and if these values are over written by malicious code, checksum computed would be incorrect.

Buffer Overflow attack: In this attack, any of the data or stack elements are extended beyond their storage capacity and malicious routines are inserted in the extended space. In our dynamic data guard algorithm as in [9] we are protecting all the data from overflow by assigning guards around it so, if buffer overflow occurs the guard elements are corrupted and the checksum generated by data guard values would be incorrect.

ROP attack: ROP [22] attack can be created using buffer overflow vulnerability or stack smashing[36] technique in both these technique the data boundary of local data is changed beyond its limit. Using dynamic data guard technique we can always detect the attack as guarding elements are manipulated which cannot be recovered again as described in [9].

Most static code attestation techniques withstand against attacks like pre-computation of checksum, proxy attack, memory copy attack and TOUTOA if strictly time bounding is included. But these techniques are lacking resistance against buffer overflow and ROP attacks which can be provided using dynamic data attestation as combining both proposed in our attestation technique.

6. METHODOLOGY OF EVALUATION AND RESULT ANALYSIS

In this section we describe the evaluation methodology and the brief analysis of the result we got from our proposed algorithm.

6.1 Methodology of Evaluation

6.1.1 Platform/Tools used

To develop our application we are using tinyos-2.1.0 version of TinyOS [27] which is a free operating system basically designed for WSNs applications. For our experimentation we are using mica2 motes with MIB510 programming board and avrora [29] emulator for computing the energy consumption.

6.1.2 Test Application

Sense application from TinyOS simulator is used as a base application for implementing the code attestation algorithms. Sense application periodically collects environment monitoring data and displays the value on LEDs. Attestation algorithms are truly applicable in such environment monitoring applications.

6.1.3 Metrics of Evaluation

For evaluating the algorithms memory overhead and computation overhead are considered for analysis.

6.2 Result Analysis

Memory required in terms of RAM and ROM usage in bytes and energy consumption in joules are considered for static code attestation, dynamic data attestation and combined algorithms.

Within the Sense application (appl) of TinyOS simulator we have added combined static code and dynamic data attestation routines and evaluated these based on different metrics viz. Storage requirement and energy consumption.

6.2.1 Storage Requirement

Storage requirement of RAM and ROM in joules are computed for mica2 mote in TinyOS. Considering the total capacity of mica2 mote as 4k Byte of RAM and 128 kByte of ROM limit for the Atmega128- microcontroller % utilization of RAM and ROM in mica2 are computed for result analysis.

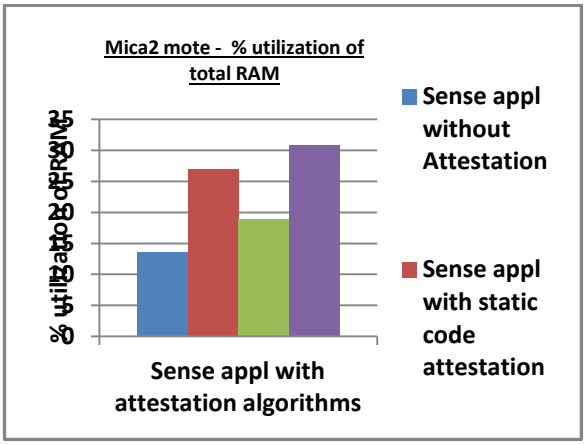

Fig 1: % utilization of RAM on mica2 mote

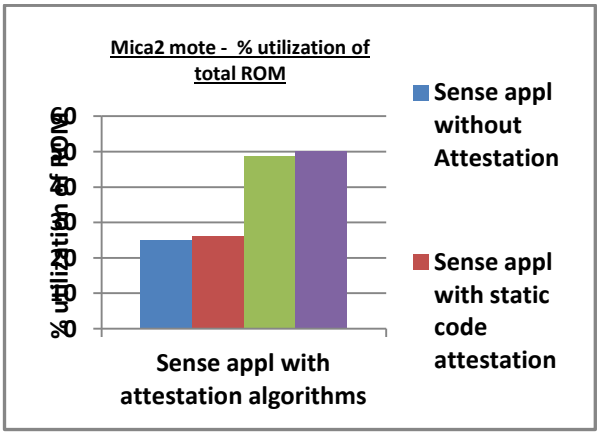

Fig 2: % utilization of ROM on mica2 mote

From the graphs shown in figure 1 and figure 2 for the memory storage in RAM and ROM we can conclude that memory required for combined technique of static code and dynamic data attestation is highest compared to Sense application with only static code attestation and with only dynamic code attestation. With only 17.23% RAM and 25.19% ROM overhead is needed in combined attestation technique compared to code without attestation, we gain resistance of the application against various attack. RAM memory is very crucial in sensor node; we are sustaining it as shown in the results.

6.2.2 Energy consumption

Figure 3 shows the energy consumption in % increase over Sense application without attestation. From the figure we can say that with only increase of 4.56% energy consumption compared to Sense application without attestation we gain security against various attacks using the combined code attestation technique.

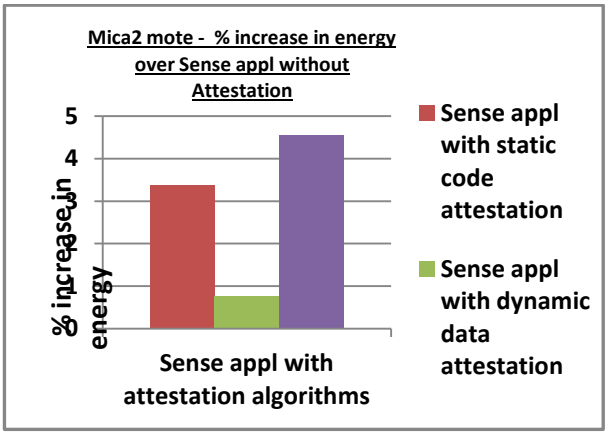

Fig 3: % increase in energy consumption over Sense appl without attestation

7. CONCLUSION AND FUTURE WORK

Using static code attestation techniques, it is possible to verify the program integrity of static code embedded within a sensor node. In addition, using dynamic data attestation, dynamically changing system properties of running application like function data can be guarded from vulnerabilities like buffer overflow. We observe that the existing static code attestation techniques are time bounded and are vulnerable to attacks like return-oriented programming attack and buffer overflow attacks, while dynamic data attestation techniques are vulnerable to attacks violating integrity of complex programming structures. In this paper, using an integrated approach we demonstrate that the static and dynamic code attestation techniques can be combinedly exploited to an advantage. To the best of our knowledge, ours is the first attempt to propose an integrated approach that realizes code attestation on motes with AVR microcontroller. From the experimental results that we obtained, using combined technique of static code and dynamic data attestation nearly 17.23% increase in RAM usage while 4.56% increase in energy consumption entails in WSNs while we get protection from attacks like buffer overflow, pre-computation of checksum and root-kit attacks.

In future work we would like to concentrate on other structural integrity violating parameters of running application which may lead to attacks in WSNs.

8. REFERENCES

[1] Y. Zhou, Y. Fang and Y. Zhang: Securing wireless sensor networks: A Survey. In: IEEE communications Surveys, vol. 10, no. 3, pp. 6-28. (2008)

[2] M. Roesch : Snort – Lightweight Intrusion Detection for Networks. In: Proceedings of USENIX LISA'99, November (1999)

[3] P. García-Teodoroa, J. Díaz-Verdejoa, G. Maciá-Fernándeza, and E. Vázquez : Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges. Computers & Security, vol. 28, no. 1–2, pp. 18–28, (2009)

[4] Sekar, R., Guang, Y., Verma, S., and Shanbhag, T.: A high-performance network intrusion detection system. In: Proceedings of the 6th ACM conference on Computer and communi- cations security. ACM Press, 8-17. (1999)

[5] V. Bhuse, A. Gupta: Anomaly intrusion detection in wireless sensor network. Journal of High Speed Networks, Volume 15, Issue 1, pp 33-51, Jan 2006.

[6] I. Onat, A. Miri.: An intrusion detection system for wireless sensor networks. In: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, pp. 253–259, Montreal, Canada (2005)

[7] Krontiris Ioannis, Tassos Dimitriou and Felix C. Freiling: Towards Intrusion Detection in Wireless Sensor Networks. In :13th European Wireless Conference, Paris, France (2007)

[8] Seshadri, A., Perrig, A., van Doorn, L., and Khosla, P. K.: SWATT: SoftWare-based ATTestation for embedded devices. In: IEEE Symposium on Security and Privacy (2004)

[9] Dazhi Zhang and Donggang Li u: DataGuard: Dynamic Data Attestation in Wireless Sensor Networks. In: IEEE/IFIP International Conference on Dependable Systems & Networks (DSN) (2010)

[10] E. Ngai, J. Liu, M. Lyu.: An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks. In: Proc. of IEEE International Conference on Communications (ICC'06) (2006)

[11] Technical report Group, T.C.:Trusted Platform Module (TPM) specifications. http://www.trustedcomputinggroup.org/specs/tpm.

[12] Christoph Krauf, Frederic Stumpf, and Claudia M. Eckert: Detecting node compromise in hybrid wireless sensor networks using attestation techniques. In: ESAS, pp 203–217, (2007)

[13] Hailun Tan, Wen hu, Sanjay Jha: A TPM-enabled remote attestation protocol (TRAP) in wireless sensor networks. In: Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks (2011)

[14] Elaine Shi, Adrian Perrig, and Leendert van Doorn: Bind: A fine-grained attestation service for secure distributed systems. In: IEEE Symposium on Security and Privacy, pages 154–168 (2005)

[15] Seshadri, A., Luk, M., Shi, E., Perrig, A., van Doorn, L., and Khosla, P.: Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In: SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles (2005)

[16] A.Seshadri, M.Luk, A.Perrig, L. van Doorn, and P.Khosla: Using FIRE and ICE for detecting and recovering compromised nodes in sensor networks. Technical Report CMU-CS-04-187, School of Computer Science, Carnegie Mellon University, (2004)

[17] T. Park, K.G. Shin: Soft tamper-proofing via program integrity verification in wireless sensor networks. In: IEEE Transaction Mobile Computing pp. 297–309 (2005)

[18] M. Shaneck, K. Mahadevan, V. Kher, Y. Kim: Remote software-based attestation for wireless sensors. In: ESAS, LNCS, vol. 3818, pp. 27–41 (2005)

[19] Seshadri, A., Luk, M., and Perrig, A.: SAKE: Software attestation for key establishment in sensor networks. In: DCOSS '08: Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems (2008).

[20] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang: Remote attestation to dynamic system properties: Towards providing complete system integrity evidence. In: Proceedings of the 39th Annual IEEE International Conference on Dependable Systems and Networks (DSN) (2009)

[21] Y. Yang, X. Wang, S. Zhu, G. Cao: Distributed software-based attestation for node compromise detection in sensor networks. In: 26th IEEE International Symposium on Reliable Distributed Systems, pp. 219–230. (2007)

[22] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente: On the difficulty of software-based attestation of embedded devices. In: Proceedings of ACM Conference on Computer and communications Security (CCS) (2009)

[23] M.J.B. Robshaw: MD2, MD4, MD5, SHA and Other Hash Functions. Technical Report TR-101, version 4.0, RSA Laboratories, July 1995.

[24] Aurélien Francillon and Claude Castelluccia: Code injection attacks on Harvard architecture devices. In: CCS '08: Proceedings of the 15th ACM conference on Computer and Communications Security, October 2008. ACM.

[25] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton: Stack-Guard: automatic adaptive detection and prevention of buffer-overflow attacks. In: Proceedings of the 7th USENIX Security Symposium, January 1998.

[26] A. Francillon, Attacking and Protecting Constrained Embedded Systems from Control Flow Attacks, Ph.D. dissertation, Institut Polytechnique de Grenoble (2009)

[27] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler: TinyOS: An Operating System for Sensor Networks. Springer-Verlag (2004)

[28] TinyOS tutorial, http://www.tinyos.net/tinyos-1.x/doc/tutorial

[29] Ben L. Titzer: Avrora: Scalable sensor Network simulation with precise Timing. In: Proceeding of 4th IPSN (2005)

[30] John McHugh, Alan Christie, and Julia Allen: Defending yourself: The Role of Intrusion Detection Systems. IEEE software (2000)

[31] F. Schepers: Network- vs. Host-based Intrusion Detection: A Guide to Intrusion Detection Technology. Information Security Technical Report (1998)

[32] Qijun Gu and Rizwan Noorani: Towards self-propagate mal-packets in sensor networks. In WiSec '08: Proceedings of the first ACM conference on Wireless network security, pages 172–182, New York, NY, USA. (2008)

[33] http://fuxi.cs.txstate.edu/~download/attack/report4demo1.pdf

[34] StackShield. http://www.angelfire.com/sk/stackshield

[35] H. Shacham," The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86)", In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), 2007

[36] A. One,"Smashing the stack for fun and profit", Phrack, 7(49), Nov. 1996.

[37] Ferguson, C., Gu, Q., and Shi. H.,"Self-healing control flow protection in sensor applications". In WiSec '09 (2009), ACM.

[38] Tran, Minh, Mark Etheridge, Tyler Bletsch, Xuxian Jiang, Vincent Freeh, and Peng Ning. "On the expressiveness of return-into-libc attacks." In Recent Advances in Intrusion Detection, pp. 121-141. Springer Berlin Heidelberg, 2011.

[39] Steven Alexander. "Defeating compiler-level buffer overflow protection." Usenix LOGIN;, 30(3), June 2005.