A Testbed for the Security Issues of Limited-resource Internet Appliances

S. Vorapojpisut

Department of Electrical Engineering, Thammasat University (Rangsit Campus)
Paholyothin Rd., KlongLuang, Pathumthani, THAILAND 12121
Tel: 66-2564-3001, Fax: 66-2564-3010, E-mail: vsupacha@engr.tu.ac.th

Abstract: This paper introduces a testbed which is suitable for the study of security issues arising in applications involving internet appliances. The testbed implements secure door locks by utilizing the intranet in the building and is composed of two main parts, namely a database server and door locks each of which equipped with a custom-made embedded system. The main objective is to provide a platform for teaching the conflict among real-time specifications, security requirements, and limited-resource constraints. After definitions of threat, vulnerability, and attack are given, we discuss how the testbed can be applied as an education tool for these concepts. Finally, the effects of sequential and multitasking operations are given as a case study.

Keywords: Security, embedded system

1. Introduction

Recently, the entry of cheap internet-enabled microcontrollers [1], [2] in the market has shown a good promise for the coming age of ubiquitous internet appliances. Since more than 90% of processor market is dedicated to embedded systems, more and more applications of this type of microcontrollers can be expected in a variety of contexts including critical services such as banking, health, and defense. As a consequence, many researchers have been conducting in-depth studies on this subject under different titles, e.g. ubiquitous computing [16], networked embedded systems [15]. The usage of such equipment in daily life raises many concerns [9] particularly on the issues of security.

The current processor trend used in appliances relies on families of 8-bit microcontrollers due to their simplicity and cost/performance. Compared to 32-bit and 64-bit processors used in modern computers, these microcontrollers have been known for their restrictions including low computational power (the order of $10^{-3} \sim 10^{-6}$), limited memory ($<$ 128 KB), and the lack of hardware-based features (e.g., cache, MMU, FPU). To design products based on these microcontrollers, the popular encryption/decryption schemes tend to consume unacceptable amount of resources whenever real-time and memory specifications are of major concern. As an example, it was reported in [12] that the 128-bit AES algorithm [7] (28-KB code size) was the fastest scheme for an 8-bit processor running at 36 MHz with the processing time of 365 milliseconds for 1024-byte data. With such level of processing time and code size, it is very hard to implement well-known secure protocols such as SSL [8] or IPSec [17] in order to comply with the requirement of response time.

A possible approach to overcome such difficulties is to design a custom protocol optimized for the nature of target products, e.g. the SPINS protocol [11] proposed for sensor network applications. Another alternative is to ease the implementation of cryptographic algorithms by using the crypto co-processor provided in secure microcontrollers [3]. In order to implement a secure protocol on an embedded system platform, the team of engineers and programmers must have enough background in the subject of computer security. An obvious obstacle is that this discipline is quite foreign to many engineers who play the main role in embedded system projects. Except for those from the field of computer engineering, just about every engineers tend to have weak background in discrete mathematics which substantially affect the acquisition of computer security fundamentals.

Hence some educating tools are needed for developing the understanding about various concepts and terminologies underlying the subject of computer security. As the basis, many computer security educators have found that the concept of threats, vulnerabilities, and attacks can be illustrated by appealing to real-life scenarios. Moreover, the interrelation among real-time specifications, security requirements, and limited-resource constraints should be exploited in order to reflect on the practical implementation of secure embedded systems. Thus a real-world testbed seems to offer an advantage for teaching and experiencing tasks of identification and prevention of malicious and potentially harmful behaviors. Searching existing testbeds [5], [16], [19] on the subject, it is amazing that the number of education-oriented platforms is quite small in spite of strong concerns from society. Besides, these education platforms [18], [22], [23] have been targeted towards the audience in the field of information technology, hence focused on existing technologies of computer network.

This paper proposes an education-oriented platform consisting of a number of limited-resource internet appliances and a database server of which they belong to the cloud of network-capable objects constituted by the internet topology. We organize the paper into three parts. The first part is the description about the structure of our testbed. After definitions are given, we discuss how to apply the testbed in order to educate the subject of security in embedded systems. Finally, we give an example to be used as a study topic.

2. The Embedded System Testbed

Our testbed was initially an undergraduate project [20] which aimed to develop a physical security system for locations in the faculty buildings. The testbed consists of a

Linux server running PHP/MySQL database services and internet-enabled embedded systems performing access control and intruder detection operations.

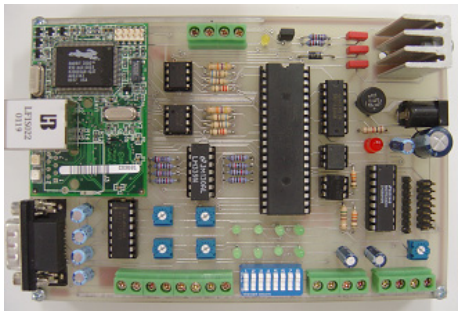

Fig. 1. The embedded system entity

Depicted in Fig. 1, each embedded system entity utilizes a Z80-variant RabbitCore RCM2200 module [1] and our custom-built interface board equipped with a barcode reader, a keypad, an electronic strike, a door-open detector, and PIR detectors. The communication between the server and these entities relies on a 10/100 Mbps Ethernet network. The integration of these components constitutes the security management service for buildings with a large number of involved people. It is worth mentioning that our platform shares the similar framework as many internet appliances expected to be found in home to business organizations. That is, the design has employed the HTTP over TCP/IP protocol, a dedicated server, and the existing intranet network.

The development of software was separated into two portions, database services on the Linux server and the firmware on each embedded system entity. The development of database services was written in PHP scripting language [13] of which process HTTP-based requests from embedded system entities, then manipulate the MySQL database engine [14] accordingly. The firmware was developed using the C compiler supplied with the RCM2200 development kit. In order to guarantee of being functional, the firmware development made use of real-time services provided by the kernel of μC/OS II [6]. Based on preemptive-scheduling, access control/intruder detection operations were decomposed into 9 prioritized tasks as listed in Fig. 2.

Priority	Task detail
2	Read barcode
3	Read PIN from keypad
4-7	Read the status of PIRs
8	Submit request to the server
9	Receive & process command from the server
10	Display the status through web page

Fig. 2. Tasks on the embedded system entity

During the development stage, it was found that our platform has exhibited several features of which security may be violated. An obvious one is that the coexistence of embedded system entities and regular computers within the same organization network leads to the potential of being observed by unauthorized staffs. As a more serious illustration, the task of access control and/or intruder detection may be suspended by attacks from computers with considerably higher-performance. Examining these situations, the platform has turned out to be an appropriate testbed for studying the issues of security and privacy involving internet appliances. Another encouraging reason is that it is easy to relate security concepts with real-life scenarios.

3. The Instruction of Security Concepts

Emerging from the field of electronics, the discipline of embedded system engineering involves the design and implementation of products which utilize microprocessor-based hardware with associated software as their cores. While the discipline of security system engineering involves the determination of the optimal security approach for a particular system based on an identification of all relevant factors and deterrents. This section discusses the application of our testbed as an education tool on the subject which correlates between embedded system engineering and security system engineering.

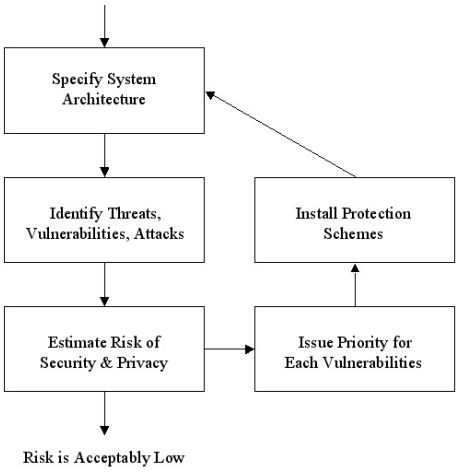

Fig. 3. The typical process of security system engineering

Fig. 3 provide details for the security system engineering process which depends upon iterative discussions on threat, vulnerability, and attack identification as well as available protections. That is, the savvy of these security concepts is an essential background knowledge on the determination and mitigation of security problems. Here we provide definitions of threats, vulnerabilities, and attacks of which we keep to those given in a computer security textbook [4].

Definition 1: A *threat* to an embedded system is any potentially malicious situation that can have undesirable effect on the assets and resources associated with the embedded system.

Definition 2: A *vulnerability* of an embedded system is some characteristic that makes it possible for a threat to occur.

Definition 3: An *attack* on an embedded system is some action taken by malicious intruder that involves the exploitation of certain vulnerabilities in order to cause an existing threat to occur.

In order to appeal with real-life scenarios, the operation of physical access control is used as the case study whose details are given as follows. In idle state, the access point waits until it can read barcode and PIN from a user's card. Then the access point submits the barcode and PIN to the server through intranet network, and waits for a responding command. If the sever authorizes the access request, the access point will unlock door by releasing its electronic strike, otherwise it will raise a warning by using its siren. Trimmed for experimentations, the firmware of embedded system entity and the PHP/MySQL script running on the server are provided in the form of pseudocodes in Fig. 4 and 5.

```
main_loop():
   while no input
      idle()
   [barcode,PIN] = read_input()
   submit(server URL,barcode,PIN)
   while no new message
      idle()
   command = get_message()
   if command == 'ok'
      unlock_door()
      record(date,time,barcode,'pass')
   else
      warning()
      record(date,time,barcode,'fail')

idle():
   process_html_request()
```

Fig. 4. Pseudocode of the firmware

```
[IP,barcode,PIN] = extract(HTTP-based request)
Initiate database connection
if user(barcode,PIN) has access rights
   submit(IP,'ok')
else
   submit(IP,'not allow')
```

Fig. 5. Pseudocode of the PHP/MySQL script

The following experiments are outlined according to three major classes of threats, namely disclosure, integrity, and denial of service threats. These experiments have been arranged such that student will learn how to identify threats, vulnerabilities, and attacks along the procedure of experiments. The minimum requirements for a classroom demonstration include a server running PHP, MySQL, and httpd services, one set of the embedded system entity, an Internet browser, and a packet sniffer software, e.g. Sniffer Pro [10] or Ethereal [21]. Although, it should be better if student can do experiments by running the packet sniffer software on their client machines with a separated embedded system entity as illustrated in Fig. 6.

3.1. Representation of Disclosure Threats

The disclosure threat denotes any potential situation of which stored or in-transit information expose to one or more person who should not know. Being arranged as the first topic, the scenario of disclosure threats involves the leakage of barcode and PIN during the access control process and the leakage of operation log with respect to the feature of on-line report. The experiment procedure is outlined as follows.

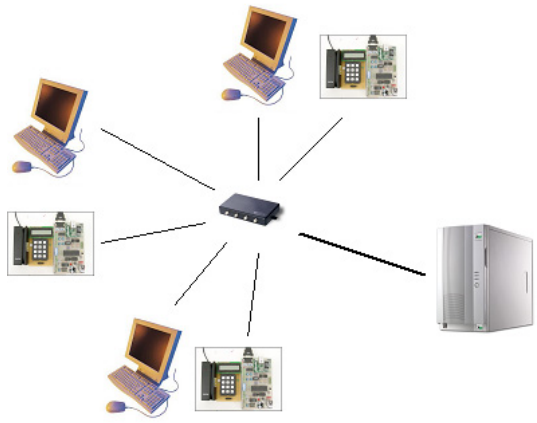

Fig. 6. Configuration of Testbed Experiments

a) Set up the packet sniffer software to capture every packet delivered between the server and the associated embedded system entity.
b) Use an ID card and then enter PIN.
c) Discuss about the criticality of information acquired from captured packets.
d) Use an Internet browser to access the URL of embedded system entity.
e) Discuss about the criticality of information seen in the browser.

The experiment objective is to give an idea of how insecure scenarios may simply exist in an embedded system by performing tapping attacks and also accesses through carelessly unrestricted channel.

3.2. Representation of Integrity Threats

The integrity threat involves any unauthorized change to stored or in-transit information or command. For the sake of easiness, the PHP script in Fig. 5 has to be modified by adding a time-delay operation at the beginning. The experiment procedure is outlined as follows. We also provide a PHP script which composes every text in a form into an HTTP request.

a) Perform tapping attacks on the embedded system entity to acquire registered barcode and PIN.
b) Use the provided PHP script to submit structured packet containing acquired barcode and PIN to the server.
c) Evaluate and discuss about the result.
d) Perform tapping attacks on the server to acquire the structure of command message.
e) Use the provided PHP script to submit packet containing unlock command to the embedded system entity.
f) Evaluate and discuss about the result.

The experiment procedure makes use of impersonation at-

tacks to clarify the vulnerability of nonencrypted packets which results in the presence of integrity threats.

3.3. Representation of Denial of Service Threats

The denial of service threat arises whenever access to some resource is intentionally blocked as a result of malicious actions.

a) Open multiple windows of Internet browser
b) Try to use every instances of Internet browser to access the URL of embedded system entity.
c) During the attempt, use an ID card and then enter PIN.
d) Evaluate and discuss about the result.

The experiment procedure employs denial of service attacks similar to the tactic of recent viruses which can block the Internet access of some web sites by using infected computers from around the world. This illustrates a vulnerability corresponding to immensely different processing power between modern computer and embedded system.

4. A Case Study on The Security Issue of Embedded Systems

Section 3. addressed the representation of threats and attacks by physical scenarios and also how to observe them within a classroom. The objective of such experimentations is to aid the instruction related to the step of identifying threats, vulnerabilities, and attacks. This section provides a case study for a discussion about how to protect a system. Note that the case study is tailored for the platform of embedded systems with respect to our assumption of limited-resource constraints.

4.1. Relationship between Security and Processing Period

In contrast to computer software, an operating system is not the necessity component for the development of embedded system firmware. There are many embedded systems which perform tasks sequentially without any form of scheduled and/or preemptive operations. That is, the capability of multitasking may not be assumed for every embedded systems. The following concept states the benefit of being sequential process in the sense of security.

Concept 1: Consider a singleton firmware which consists of strict sequential tasks. With the assumption of independent weaknesses for each task, the single-tasking firmware is less or equally prone to attack when compared to its decomposed, multitasking counterpart.

Proof: Let $(\tau_1, t_1), \ldots, (\tau_n, t_n)$ be pairs of tasks and their corresponding processing periods. Let $E_{\tau,t}$ be an event in which there is a successful attack against τ in time period t and $P(E_{\tau,t})$ be the corresponding probability of successful attack. First, it follows from the assumption that the probability of successful attack for a sequential process is $\max_n P(E_{\tau_i,t_i})$. Assume the independence of tasks, the probability of successful attack for a multitasking process is $\max_n P(E_{\tau_i,T})$ where $T = \sum_{i=1}^{n} t_i$ is the total processing time. Since $t_i \leq T$ for any t_i, we can conclude that $\max_n P(E_{\tau_i,t_i}) \leq \max_n P(E_{\tau_i,T})$. Then it is straightforward to show that $\max_n P(E_{\tau_i,T})$ is the upperbound and $\max_n P(E_{\tau_i,t_i})$ is the lowerbound for the probability of successful attack for a multitasking process with dependent tasks. These conclude the proof. ■

In the case of embedded system, the multitasking capability relies on the mechanism of context switching which schedules machine instructions of each tasks into the execution of processor. As a consequence, the processing time of each task does not directly correspond to the actual elapsed time. For instance, if there are 10 tasks each of which is processed in 1 second, then every task will end at about 10 seconds in the case of round-robin scheduling. That is, the multitasking process may extend the elapsed time of each task until the total time frame of process, even the processing time for each task is fixed. Since it is natural to say that the probability of successful attack increases if you provide more time to the attacker, we can conclude that the extension of elapsed time causes the overall system to be less or equally prone to attack.

4.2. Security of Sequential and Multitasking Processes

Using the same problem as in Section 3., the pseudocode of firmware in Fig. 4 is obviously an implementation based on sequential process. Decomposing into a multitasking process, the following pseudocode is implemented without any synchronization among tasks.

```
main():
   Initiate barcode_task()
   Initiate server_task()
   while true
      idle()

barcode_task():
   while true
      wait_for_input()
      [barcode,PIN] = read_input()
      submit(server URL,barcode,PIN)

server_task():
   while true
      wait_for_message()
      if command is 'ok'
         unlock_door()
      else
         warning()
```

Fig. 7. Implementation of firmware with parallel tasks

Even with the assumption of strong authentication associated with server_task(), the system is more prone to impersonation attacks due to the persistence of running server_task(). The following pseudocode makes use of the synchronization between barcode_task() and server_task() in order to limit the period of server_task() within the overall time frame.

```
main():
   Initiate barcode_task
   while true
      idle()

barcode_task():
   while true
      wait_for_input()
      [barcode,PIN] = read_input()
      submit(server URL,barcode,PIN)
      Initiate server_task()
      Wait until server_task() ends
      Kill server_task

server_task():
   wait_for_message()
   if command is 'ok'
      unlock_door()
   else
      warning()
```

Fig. 8. Implementation of firmware with synchronized tasks

5. Conclusion

This paper describes an internet appliance testbed to be used as an education tool for the subject which correlates between embedded system engineering and security system engineering. Based on the operation of physical access control, we provide three experiment procedures as a guideline for the instruction of threat, vulnerability, and attack identification. As a case study, the discussion on the topic of firmware design indicates that a sequential process is better or equal than its multitasking counterpart whenever the requirement of security is the major concern.

Acknowledgements

The author would like to gratefully thank Dr.Chanathip Namprempre for many helpful comments and suggestions during the manuscript preparation.

References

[1] Rabbit Semiconductor, *RabbitCore RCM2200 User's Manual*: http://www.rabbitsemiconductor.com.

[2] Atmel Corp., *@Web Hardwired TCP/IP Ethernet Solutions*: http://www.atmel.com.

[3] Atmel Corp., *AT90SC SecureAVR High-performance Secure 8-/16-bit RISC Architecture*: http://www.atmel.com/products/secureAVR/.

[4] E. Amoroso, *Fundamentals of Computer Security Technology*, Prentice-Hall, 1994.

[5] C. Christian, *Breaking the Barriers to the NII*, Presentation at the Council on Competitiveness, The National Information Infrastructure Testbed Consortium, September 1994.

[6] J. Labrosse ,*MicroC/OS-II, The Real-Time Kernel*, CMP Books, 1998.

[7] J. Daemen and V. Rijmen, *AES proposal: Rijndael*, 1999.

[8] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, 2000.

[9] Mudge, "Security Devices That Might Not Be", *;login*, vol.25, No.7, pp.49-51, November 2000.

[10] R.J. Shimonski, W. Eaton, U. Khan, and Y. Gordienko, *Sniffer Pro Network Optimization and Troubleshooting Handbook*, Syngress Pub., 2002.

[11] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D. E. CULLER, "SPINS: Security Protocols for Sensor Networks", *Wireless Networks*, vol.8, pp.521-534, 2002.

[12] N. Okabe, *Security Requirements and Impacts for Embedded Systems*, Workshop Presentation at Global IPv6 Summit in Japan, 2002.

[13] L. Welling and L. Thomson, *PHP and MySQL Web Development, 2nd edition*, Sams Pub., 2003.

[14] P. DuBois, *MySQL, 2nd edition*, Sams Pub., 2003.

[15] H. Saidi, *Security Codesign for NES*, Presentation at Trustworthiness in Embedded Systems Event, Center for Information Technology Research in the Interest of Society, 2003.

[16] D. Tygar, *Privacy and Security in Ubiquitous Computing*, Presentation at Trustworthiness in Embedded Systems Event, Center for Information Technology Research in the Interest of Society, 2003.

[17] N. Doraswamy and D. Harkins, *IPSec, 2nd edition*, Pearson Education, 2003.

[18] N. Memon, *Overview of Security Research at Polytechnic University*, Presentation at New York State Cybersecurity Symposium, Griffis Institute, February 2003.

[19] N. Breems, Y. Perzov, and R. Iyer, *Fault Tolerance and Security Testbed for Ad Hoc Networks*, Semi-annual Meeting's Slide, Quality of Service in Surveillance and Control Project, June 2003.

[20] T. Agganapanya and K. Nganpodoung, *Security system for large organization based on internet technology*, Undergraduate report, Thammasat University, 2004.

[21] A.D. Orebaugh and G. Ramirez, *Ethereal Packet Sniffing*, Syngress Pub., 2004.

[22] J. Hu, C. Meinel, and M. Schmitt, *Tele-Lab IT Security: An Architecture for Interactive Lessons for Security Education*, SIGCSE2004, Villanova University, USA, March 2004.

[23] P. Wagner and J.M. Wudi, *Designing and Implementing a Cyberwar Laboratory Exercise for a Computer Security Course*, SIGCSE2004, Villanova University, USA, March 2004.