Course: ENPM 695 – Secure Operating Systems  
Semester: Spring 2017  
Day(s): Th  
Time: 4:00 – 6:40PM  
Location: JMP 2121  
Instructor: Ido Dubrawsky  
Phone:  
Email: idubraw1@umd.edu

Course Description

Prerequisite

ENPM 808 Programming in C for Cybersecurity Applications, CMSC 106 Introduction to C Programming, or permission of the instructor.

Catalog Description

Operating systems are the basic building block on which programmers build applications and on which security-minded professionals rely, whether they are monitoring activity on a computer, testing applications for security, or determining how malicious code affected their network. This course covers advanced topics in operating systems including process management and communication, remote procedure calls, memory management (including shared memory and virtual memory), checkpointing and recovery, file system, I/O subsystem and device management, distributed file systems and security. The course consists of reading and discussing current papers covering these and other topics as well as a course project. Please note: This course assumes knowledge of C programming and a previous operating systems class or knowledge in various issues such as process management, process synchronization, the critical section problem, CPU scheduling, memory management, secondary storage management.

Detailed Description

An operating system consists of a core kernel surrounded/supported by modules that provide basic operating services (file system management, memory management, etc.). Some parts of the kernel and modules are relevant to the security of the overall operating system. For example, the memory space allocated to a specific user may need to be protected from other (possibly malicious) users – the memory management module provides this security. Similarly, persistent storage space of a user may need privileged access – the file system provides this facility. Closely related to security policies is understanding the behavior of attackers and malicious users. For example, one common attacker behavior is to predict target memory addresses of programs and then use that information to redirect program execution to injected code. Because of this behavior, one security mechanism typically implemented in the memory management module randomizes address spaces, thereby hindering such types of attacks by making it more difficult for an attacker to predict target addresses. Application developers need to understand these behaviors so that they can assess the security risks of running their application on a particular operating system. Additionally, application developers need to have a firm understanding of how to secure the code used to write programs. Understanding and using secure operating systems thus requires the study and understanding of five (5) separate conceptual threads:

1. operating system fundamentals,
2. secure code fundamentals,
3. understanding security features of kernel and modules,
4. attacker behavior and attack patterns, and
5. practical security features provided by today’s operating systems.
While it is important to understand the theoretical foundations of secure operating systems, it is equally important to study how the theory is realized in a practical, modern, and widely used operating system. Discussion of each security feature studied in this course is therefore, tightly coupled with a discussion of how the feature is implemented in the Linux and Android operating systems. For example, the lecture on memory protection includes a discussion of Windows, Android and Linux memory management security enhancements that make common security issues harder to exploit.

This course presents four perspectives relevant to secure operating systems:

1. operating system designer,
2. programmer, who develops code that run on the operating system base,
3. end-user, who simply uses the operating system and applications, and
4. administrator, who manages and administers the operating system

The operating system designer creates and implements the security model that the operating system provides; the programmer uses the security model to develop software applications that rely on the implementation to provide security; the end-user expects the operating system and applications to work together seamlessly, providing a reasonable level of security; the administrator leverages the operating system features and functions to provide a secure platform for the running applications.

After a short "introduction to OS" thread, the course dives into content that is interwoven with 4 threads that cover theory as well as practice: (1) secure coding overview, (2) kernel and core modules of an operating system, (2) security aspects of the kernel and relevant core modules, (3) how attackers think, and (4) practical case studies of Unix/Linux which discuss the mechanisms these OS’es provide, and how programmers may leverage these mechanisms for secure software development and analysis.

Textbook(s)

Recommended (but not required) Textbooks:
1. Operating System Security (Synthesis Lectures on Information Security, Privacy and Trust) by Trent Jaeger

In addition to these texts we will cover excerpts from other texts as well including:
1. Building Secure Software by John Viega and Gary C. McGraw
2. Design and Implementation of the FreeBSD Operating System by Marshall Kirk McKusick, George Neville-Neil and Robert Watson

Course Outline

Grading: The grade of the course will be determined as follows: 25% assignments (we will have 5 assignments each worth 5% of the final grade), 25% midterm, 25% final exam and 25% semester project.

Code of Academic Integrity

The University of Maryland, College Park has a nationally recognized Code of Academic Integrity, administered by the Student Honor Council. This Code sets standards for academic integrity at Maryland.
for all undergraduate and graduate students. As a student you are responsible for upholding these standards for this course. It is very important for you to be aware of the consequences of cheating, fabrication, facilitation, and plagiarism. For more information on the Code of Academic Integrity of the Student Honor Council, please visit http://shc.umd.edu/SHC/HonorPledgeInformation.aspx.
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Purpose</th>
<th>Readings/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan. 28</td>
<td>Course Overview. What is a secure OS?</td>
<td>To introduce the course and basic terminology. This lecture will cover a history of a variety of secure operating systems focusing on the development security. By the end of this lecture, students will understand the need for a secure operating system and be familiar with the terms commonly used in the secure operating systems literature.</td>
<td>Various resources, textbooks, and Internet.</td>
</tr>
</tbody>
</table>
| Feb. 4  | Threats, Attacks and Exploits | This lecture is designed to provide students with an introduction to threats and attacks against operating system focusing on low-level exploitation and how they work internally. Discussion will focus on buffer overflows, format string attacks, heap overflows, integer overflows and how they work within the memory of an operating system. By the end of this lecture the students should understand how these attacks work and why they represent such a danger to operating systems. | Quiz 1 (in Class)  
Various lists of attacks from the Internet. |
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Lecture Details</th>
<th>Quiz 2 (in Class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb. 11</td>
<td>Parts of an OS</td>
<td>The significant modules that comprise an OS; the separation between spaces; how spaces interact in an OS. By the end of this lecture, students will understand the parts of an OS, the security requirements of each part, and how a vulnerability in one part may have an impact on another part.</td>
<td>kGuard: lightweight kernel protection against return-to-user attacks. By Vasileios P. Kemerlis, Georgios Portokalidis, and Angelos D. Keromytis. In Proceedings of the 21st USENIX conference on Security symposium (Security'12).</td>
</tr>
<tr>
<td>Feb. 18</td>
<td>Threat Modeling and Secure Coding</td>
<td>This lecture introduces students to evaluating the security of a system but building simplified threat models of components and the system as a whole. In addition the students will be introduced to building secure software and secure coding techniques to help mitigate common threats and attacks as covered in the second lecture.</td>
<td>Quiz 3 (in Class) Operating system textbook.</td>
</tr>
<tr>
<td>Feb. 25</td>
<td>Cryptography</td>
<td>This week is focused on cryptography and the role it plays in operating systems - from file systems to communications to applications. This lecture introduces the student to basic concepts in cryptography - hashes, symmetric key algorithms, public-key algorithms and how they are implemented in modern operating systems. Additionally the lecture covers the finer parts of these implementations such as protecting key material in memory, sufficient entropy generation, and various attacks against cryptographic systems.</td>
<td>Quiz 4 (in Class) ASIST: architectural support for instruction set randomization. By Antonis Papadogiannakis, Laertis Loutsis, Vassilis Papaefstathiou, and Sotiris Ioannidis. In Proceedings of the 2013 ACM SIGSAC conference on Computer &amp; communications security (CCS ’13). ACM, New York, NY, USA, 981-992. Even more patterns for secure operating systems. By Eduardo B. Fernandez, Tami Sorgente, and Maria M. LarrondoPetrie. In Proceedings of the 2006 conference on Pattern languages of programs (PLoP ’06). ACM, New York, NY, USA, Article 10, 9 pages. Additional material on concurrency, locks, mutexes, etc., those ensure security by separation of access to critical sections of program data.</td>
</tr>
<tr>
<td>Date</td>
<td>Topic</td>
<td>Description</td>
<td>Quiz/Reference</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Mar. 4 | Operating System Components and Architecture | This lecture reviews the various components of modern operating systems - file systems, the kernel, processes, memory, daemons and services. The focus will be on the complexity of modern operating systems which makes securing them more difficult. | First InClass Test  
Operating system textbook.                                                                 |
| Mar. 11| File Systems                                 | Overview of filesystem design and architecture with particular emphasis on security features within various file systems such as ZFS, Fast Secure File System (FSFS), Secure File System (SFS), as well as network file systems and their challenges such as NFSv4, OpenAFS, and CIFS. The focus will be on performance of secure file systems, implementation challenges and key management. | Quiz 5 (in Class)  
| Mar. 25| Memory Management                            | The focus of this lecture will be on improving the student’s appreciation of the intricacies of memory management under specific languages (both managed and unmanaged languages) and of the difficulty in properly securing memory in a system. By the end of this lecture the student will understand the need for strong memory management, the details of securing sensitive information (cryptographic keys, passwords, etc) in memory and how to accomplish these in an operating system. | Quiz 6 (in Class)  
Operating system textbook.                                                                 |
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Apr. 1 | In-Class Mid-Term | Quiz 7 (in Class)  
| Apr. 8 | Spring Break | Quiz 8 (in Class)  
| Apr. 15 | Processes and Threads | Two of the basic components of an operating system are processes and threads. Threads can be considered the units of execution within processes which themselves are abstractions of running programs. Processes include binary images and virtualized memory while threads can be a virtualized processor, a stack or a program state. The security of both of these components through isolation comprises the fundamental level of security in an operating system. |  
Second In-Class Test  
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Lecture Content</th>
<th>Quiz (in Class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr. 22</td>
<td>Secure Hardware</td>
<td>&quot;If you don't have physical control of the computer... it's not your computer&quot; - an axiom that still remains true today as it was 10 years ago. This lecture will introduce students to new hardware capabilities providing the promise that this axiom may not hold true forever. Technologies such as TPM, Full-Volume Encryption (FVE) and chip-level security and management features will be discussed with the focus on how they are changing the landscape of system security from a physical perspective. In addition we will look at hardware back-door insertion and how it may (or may not) be detected.</td>
<td>Mandatory access control with a multi-level reference monitor: PIGA-cluster. By Mathieu Blanc, Damien Gros, Jérémy Briffaut, and Christian Toinard. In Proceedings of the first workshop on Changing landscapes in HPC security (CLHS '13). ACM, New York, NY, USA, 1-8. Aspects of security that the end user sees. Various Internet resources.</td>
</tr>
<tr>
<td>Date</td>
<td>Topic</td>
<td>Details</td>
<td>References</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>May 6</td>
<td>Kernel Level Security</td>
<td>The kernel is the heart and brain of the operating system. As attackers have become more sophisticated attacks against the kernel have become more prevalent. This lecture will introduce students to kernel security features in the Linux and BSD kernels, the concept of protection rings, process level authentication, as well as...</td>
<td>Quiz 11 (in Class) From Android developers site; Android Security</td>
</tr>
<tr>
<td>May 13</td>
<td>Untrusted Operating Systems</td>
<td>This lecture focuses on the issue of handling a compromised operating system while still ensuring that a compromise of the operating system does not impact the ability of applications to run on the operating system and to protect the data of those applications</td>
<td>Towards Application Security on Untrusted Operating Systems, Ports, Dan R. K. and Tal Garfinkel, 3rd USENIX Workshop on Hot Topics in Security, 2008</td>
</tr>
<tr>
<td></td>
<td>Security Models: Linux, BSD, Qubes OS</td>
<td>This lecture will focus on the security models used in popular operating systems such as Linux (using Ubuntu Linux and CentOS as the two models) as well as the BSD variant of UNIX (using FreeBSD as the model). The intent is to have students learn about the differences in how these UNIX variants approach security and where they have similarities with a specific focus on the Qubes OS project and how they have hardened the operating system beyond the normal efforts conducted by many security professionals.</td>
<td>Implementing an Untrusted Operating System on Trusted Hardware, Lie, David, Thekkath, C. A. and Mark Horowitz, 19th ACM Symposium on Operating Systems Principles (SOSP), 2003</td>
</tr>
</tbody>
</table>