A Security-centric Ring-based Software Architecture

Jay-Evan J. Tevis  
Department of Computer Science  
Western Illinois University  
Macomb, IL 61455  
jj-tevis@wiu.edu

John A. Hamilton, Jr.  
Department of Computer Science and Software Engineering  
Auburn University  
Auburn, Alabama 36849  
hamilton@eng.auburn.edu

Keywords: computer security, software architecture, Warden’s five-ring model

ABSTRACT

Software architectures that are centered on security historically contain layers to implement security controls, with the layered structure tending towards a series of rings. John Warden devised a five-ring model to assess enemy systems for strategic warfare attacks. We propose applying this same model to the domain of software in order to create a security-centric ring-based software architecture. This architecture would provide a foundation for software systems that would be highly resistant to malicious software.

1. INTRODUCTION

In today’s world, it is all too clear that software systems are vulnerable to many different forms of attack. This makes securing computer systems a difficult task to perform. In order to protect systems, one can view them from the perspective of the enemy. A system model for strategic warfare developed by Colonel John A. Warden, III provides an excellent way to model enemy systems in this manner. Using Warden’s model we can decompose a computer system into a five-ring model. Warden originally used his model to conduct strategic targeting against the most vulnerable points of a potential adversary. We apply this model to software architecture in a similar way, thus identifying what system components are essential and how these components can be better protected through better software design.

The rest of the paper is organized as follows. First, we summarize traditional approaches to a security-centric software architecture and highlight ring-based variations on this architecture. Then, we introduce Warden’s five-ring model and apply it to the domain of computer systems. Finally, we focus on the centers of gravity in a software system in an effort to create a security-centric ring-based software architecture.

2. SECURITY-CENTRIC SOFTWARE ARCHITECTURES

The ideas behind using a security-centric ring-based software architecture to ensure survivability and security of a system are not new. Neumann [1] outlines three critical concepts in the domain of security: multilevel security, multilevel integrity, and multilevel availability. Multilevel security restricts information from flowing from one entity to an entity with a lower security level. This is important in terms of confidentiality. An entity is not permitted to depend on an entity with a lower level of integrity in multilevel integrity. And similarly, multilevel availability restricts an entity from depending on an entity with a lower level of availability. Neumann [1] points out that while these concepts of multilevel security remain important ideas, they are rarely implemented in real systems.

A more recent security-centric ring-based architecture is described in [2] as shown in Figure 1.

Figure 1 - Seven Ring Gemguard Architecture

Their software architecture consists of seven security rings, with guard software in each ring. They state that...
this approach provides the following: **high assurance security** through hardware and kernel enforced protection against malicious software; **multilevel security** by enforcing organizational access controls that cannot bypassed; **cryptographic communication security** using IPSec based authentication, confidentiality, and integrity; and **integrated information systems security** through complementary protection at transport and network layers, and support for multi-vendor products with increased security and assurance while reducing cost and risk.

An editorial paper by Schell [3] shows that recent computer prototyping efforts have shown properties of ring-based software architectures that include memory segmentation and at least three hardware states for use in implementing protection rings. These rings are for the security kernel, the operating system, and the applications. The security kernel, located in the most protected ring, enforces mandatory access control policies. Schell [3] argues that the value of applying a ring-based software architecture for security has been shown through research, but industry has failed to widely deploy systems following such an architecture.

In an application-specific implementation of a ring-based architecture, Nguyen and Levin [4] describe improvements made to the OpenSSH authentication service to restrict the execution OpenSSH processes by applying a ring-based program execution policy. The authors state that mandatory access control and ring protection can make application-level security mechanisms more secure. In their implementation, all users including the root user are subject to mandatory access control and ring enforcements. Four rings correspond to four execution domains: unprivileged application, privileged application, admin, and operating system as shown in Figure 2. Programs assigned to a less privileged ring will be unable to execute or access objects allocated in a more privileged ring.

The above descriptions support advantages of a security-centric software architecture. However, these architectures all apply to specific areas of the software and/or hardware rather than encompassing the entire system with a focus on security. Such an architecture would be very useful to engineers who look to design security-centric software systems from the top down.

![Figure 2 - Rings Applied to Execution Domains](image)

### 3. DESIGN OF A RING-BASED SOFTWARE ARCHITECTURE

Shaw and Clements [5] define an architectural style as a set of design rules that identify the kinds of components and connectors that may be used to compose a system or subsystem together with local or global constraints. Real systems typically hybridize and amalgamate the pure architectural styles. This allows the architect to choose the useful aspects of several different styles in order to accomplish the task at hand.

A technique that utilizes parts of various architectural styles is the layered approach, in which each layer contains one or more software objects or services (known as entities). Bachmann et al [6] classifies the ring-based architecture as the most common variation of the layered architecture for software systems. The innermost ring corresponds to the lowest layer, while the outermost ring corresponds to the highest layer. Geometric adjacency or an arrow between rings denotes the "allowed-to-use" relation. And, unlike a layered architecture, a ring-based architecture can show geometric adjacency of the entities [6]. Just as east meets west on a globe, so can the entities on each side of a ring be joined together.

Another difference between ring-based and layered architectures is that a ring-based architecture logically implies that in order to penetrate an inner layer, the adjacent outer layer must be penetrated first. This is an important detail that is not necessarily present in a layered architecture. Finally, any entity in an inner ring is accessible by any of the entities in its closest outer ring. Consequently, adjacency of entities within a specific ring (or among adjacent rings) has no implicit meaning; entities in a ring can actually be viewed as members of a set. From this point of view, set membership indicates a
specific level of security access and protection. Interaction between the entities in a specific layer (or ring) moves the architecture away from a layered approach and more towards a distributed environment concept [6].

The architecture approach we propose not only allows a ring to act as a location for object or service entities, but it also permits a ring to function specifically as an interface between its adjacent rings. Such an interface can offer confidentiality (privacy), authentication (who created or sent the data), integrity (data has not been altered), non-repudiation (the order is final), access control (prevent misuse of resources), and availability (permanence or non-erasure of data).

This interface concept is analogous to the concept of gates. Fernandez [7] states that some computer software architectures define a set of protection rings that correspond to domains of execution with hierarchical levels of trust. Rings are a generalization of a mode of operation that defines domains of execution used in Multics and the Intel processors. A crossing of rings is done through gates that check the rights of a process. A process calling an entity in a higher ring must go through a gate. These gates act as protected entry points.

A specific ring interface needs to handle both control and data issues that occur between architecture components. Control issues in an architectural style describe how control passes among connectors [5]. In terms of a ring-based architecture’s topology, the kind of geometric shape the control flow can take will need to be addressed, whether it is linear, acyclic, hierarchical, etc.

Data issues describe how data moves around a system. They include topology, continuity, mode, and binding time. Control and data interaction issues include shape and directionality [5].

The design of a ring-based software architecture, with each ring denoting a set of entities or interfaces, can follow one of the patterns for operating system access control proposed in [8]. The paper describes five patterns that can be used to control access to process resources: the file authorization pattern, the virtual address space access control pattern, the execution domain pattern, the reference monitor pattern, and the controlled execution environment pattern.

Now that we have summarized traditional design approaches for a ring-based software architecture, we will next apply Warden’s five-ring model to improve on this architecture.

4. A COMPUTER SECURITY ADAPTATION USING WARDEN’S CONCENTRIC RINGS

It is meaningful to think of a software system as an enemy system to be exploited. This is essentially how computer crackers view software systems. In this sense, engineers should look to design the architecture for a computer software system using the perspective of an enemy who wants to exploit the system.

Warden’s Five-Ring Model

Warden [9] outlines an approach for analyzing the concept of an enemy system using the five-ring model shown in Figure 3. The five rings represent five main components of a system: central leadership, organic essentials, infrastructure, population, and a fighting mechanism. This gives the simplest possible view of the overall system that makes the model more accessible. Once a generic model is described, more and more detail can be added by expanding particular areas [9].

Warden describes the five rings by decomposing the human body into the five-ring model. At the center of the model is the brain along with its preceptors such as the eyes and ears. The preceptors allow the brain to gather and distribute information. The brain provides the leadership to the entire system making it the most essential component. While the body can survive without the brain, it will cease to be what Warden calls a “strategic entity”. A strategic entity is simply an entity that is capable of making decisions.

The second ring of the model is organic essentials that consist of essential inputs and the facilities to transform the inputs into energy that can be directly used by the system. The organic essential inputs for a human body refer to direct sources of energy such as food and oxygen.

![Figure 3 - Warden’s Five-Ring Model](image-url)
The vital organs such as the heart and lungs take the inputs and convert them into a form usable by the system. These vital organs comprise the rest of the second ring.

The third ring is the infrastructure. This would include the bones, blood vessels, and muscles. The further we get from the center the less vital the rings are. For example, a human being can survive as an entity even if he or she loses a limb or two. In comparison, losing a vital organ such as the heart makes it much more difficult to survive.

The fourth ring is the population of the system. In Warden’s example of the human body, the population ring could consist of cells. Once again this ring is less vital than the previous ring. The human body can lose countless numbers of cells and still survive.

The fifth and final ring is the fighting mechanism. The human body uses white blood cells as a means to fight against foreign pathogens present within the system. While a system can survive for a while without a fighting mechanism, it most likely will not survive very long.

Warden’s model is generic. Figure 4 shows how the five-ring model can be used to model other systems such as a political state, a drug cartel, and an electrical power grid.

<table>
<thead>
<tr>
<th>Body</th>
<th>State</th>
<th>Drug Cartel</th>
<th>Electric Grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leadership</td>
<td>Brain - eyes - nerves</td>
<td>Govt. - comm. - security</td>
<td>Leader - comm. - security</td>
</tr>
<tr>
<td>Organic Essentials</td>
<td>Food and oxygen</td>
<td>Energy Money</td>
<td>Coca source plus conversion</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Vessels, bones, muscles</td>
<td>Roads, airfields, factories</td>
<td>Roads, airways, sea lanes</td>
</tr>
<tr>
<td>Population</td>
<td>Cells</td>
<td>People</td>
<td>Growers</td>
</tr>
<tr>
<td>Fighting Mechanism</td>
<td>Leukocyte</td>
<td>MPs firemen</td>
<td>Street Soldiers</td>
</tr>
</tbody>
</table>

Figure 4 – Five Ring Model Applied to Other Domains [9]

Adapting Warden’s Model to Computer Security

Warden has shown how his model can be used to describe a variety of systems. We now apply the generic nature of that model to a computer system. By doing so, this application process uncovers new security vulnerabilities of a software system, thereby shedding light on the major objectives of a security-centric software architecture.

Warden [9] points out that strategic thinking requires the use of deductive reasoning. From an architecture perspective, this means a software engineer should build a well-organized system using a top down approach. The engineer needs to focus on the totality of the system security rather than trying to design ad hoc security features for various system components. By considering the overall system first, an engineer can construct a system such that an attack, if waged, would be in vain and a waste of the enemy’s time and resources. In order to create such a software system, one must start with a security architecture point of view that denies any kind of strategic paralysis (the physical inability to oppose an attack from the enemy).

In applying Warden's ideas, each software system in its first (i.e., center) ring has some kind of central control that gives leadership, direction and meaning to the system. The function of every subsystem depends on this central control, which sets the objectives of the system. This is the primary target of the enemy and should be the primary element to be defended by the system. The center ring’s leadership consists of the program executable code and any sensors (input/output devices) used by that code to perceive the status of the environment. The executable code gives meaning and purpose to a computer. Like the brain in the human system, without the executable code, the computer system is just an energy sink that serves no purpose. While we can’t really apply Warden’s definition of a strategic entity to executable code without getting into a philosophical debate, we can state that executable code provides leadership and purpose for the rest of the system. Without executable code, the CPU, RAM, and other hardware are meaningless.

The second ring of a software system contains the organic essentials. These would consist of electrical power and electromagnetic signals as well as the components used to convert electrical inputs into something that the system can use. The facilities to convert the inputs could include power cables as well as the logical hardware components that turn the signals into 1’s and 0’s.

The third ring is the infrastructure. This is the transmission part of the software system, which includes
the main memory, the system bus, and any communication cables. It also includes the major production components of the system that transform input data into a finished or partially finished product and pass it onto another component. There needs to be redundancy in the third ring of a computer system to allow survival in the event of damage to infrastructure.

The fourth ring contains the population. This ring could represent several things in a computer system. If communication via a computer network is the focus then the population could consist of packets or frames of information being sent across a network. Clearly, a number of packets and frames can be lost without destroying the entire system. We know for a fact that packets and frames are lost from time to time in modern computer networks, even when malicious behavior is not involved. In the case of a single computer, the population could be bits or bytes of data either in RAM or on the hard drive.

The fifth and outermost ring consists of components whose sole job is to attack and nullify the effect of any intruder attempting to enter or damage the system. This would involve a set of protection components that are called upon by a component from an inner ring to deal with an intrusion. The system should be able to generate as many of these protection components as necessary.

It is important to note that a component in any of these five rings is also a system with its own internal set of subsystems that can be categorized into these five levels. The system can become more and more detailed as needed in order to decompose a ring into additional rings or categories.

5. PROTECTING CENTERS OF GRAVITY IN A SOFTWARE SYSTEM

Warden refers to centers of gravity as the components instrumental to a system’s function and survival. In his model, the five rings describe these centers of gravity. Consequently, the whole software system should be viewed as a possible target when securing it, not just the security components in the outer ring. Without the functioning inner rings, the outer ring is a useless appendage. Regardless of this fact, engineers need to ensure that the security components in each ring cannot be easily defeated, thus allowing easy access to the next inner ring.

A system may have several crucial components that, if compromised, would lead to the failure of the first (i.e., leadership) ring. Once these crucial components have been identified they can be given high priority with respect to security. The leadership ring must receive the highest level of protection of the whole software system. If it fails, then the whole system fails. No method should exist that would allow changes to the program executable code without approval of the leadership ring. In addition, the leadership ring should control the disabling of any system sensors. With the innermost ring protected, each of the remaining rings must be protected in turn to avoid the threat of strategic paralysis.

One way of protecting the software components in the second (i.e., organic essentials) ring is through redundancy. This involves back-up battery power or alternate power sources. It also involves redundant components for sending and receiving electromagnetic or optical signals, or the ability to send and receive signals in different parts of the electromagnetic spectrum.

Protection must also occur through the sensing of excessive battery drain or signal jamming. If these signals are detected, the system should take action in order to attack the source of the battery drain, switch frequencies, or shut down the system for a period of time until the battery drain or frequency jamming ceases.

Redundancy should also be used to protect the third (i.e., infrastructure) ring. This can be achieved by the use of a second system bus or additional communication cabling. In addition, backup components are needed for each of the major production/transformation components of the software system. The software system should also provide support for transportation of data and information control between components. One way to achieve this is with a slow, but dependable method such as message passing on a system bus. Another way is to use a faster but less dependable method such as shared memory or magnetic/optical data storage and retrieval. Moreover, the system should have the ability to detect a denial of service attack and delete low priority or data-jamming traffic in order to thwart such a threat.

The fourth (i.e., population) ring is less vulnerable to attack because of the large quantity of data containers that the system can produce. The major threat is exhaustion of memory due to dynamic allocation. Another threat is corruption or destruction of the contents of the data when in transit on a system bus or communication cable. It is well understood that various error-detection mechanisms and sliding-window protocols used in typical computer networks should be applied as a security measure for this threat [10].
The fifth ring that constitutes the attacking force of the software system is not as vital if each of the inner rings has been equipped with security protection measures as detailed above. Nevertheless, the software engineering design goal of cohesion requires that the attacking role be centralized in one set of components rather than mixed in with the functionality of the components in the various other rings. There should be no doubt left in the minds of any attacker that the system has the means and the will to fight back. This should include automatic shutdown of a system until a certain signal is received or a key is entered to restore normal operation.

The system can also support special distress signals. These signals would not only alert other systems in the local network about a security threat, but would also act as a beacon to home in on the enemy in possession of the compromised system. A more drastic measure could involve anti-tampering devices built into the system that cause erasure or self-destruction of certain key components if such parts are opened or removed from the unit.

When designing security for a system, the detection and handling of threats should always assume a parallel attack [9]. Such an attack means that more than one component in a ring or among different rings will be attacked simultaneously. This requires that the attack and defense mechanisms of the software system not be geared around a single thread (i.e., serial) approach. Consequently, each defense mechanism in each ring should have the capability to react independent of the threat responses in other rings.

Centers of gravity in a software architecture are thus the components of a system that provide important functionality and survivability. The importance of such components makes them prime targets for attack. If a center of gravity is compromised, it could mean the demise of the entire system. The five-ring model gives a designer a way to identify and secure the centers of gravity when formulating a security-centric ring-based software architecture.

6. RELATED WORK

We can see examples from other research of how computer systems have vital areas of control that must be given higher priority with respect to security. Kopp [11] points out that the center ring of a software system contains command and control processing facilities and the storage of related data. Damage to this area will result in substantial reduction in the computer’s ability to handle and process information. Without the ability to process information the system ceases to survive.

Devanbu and Stubblebine [12] state that from a security perspective, the most serious problem in a software system is one of mismatch between the security framework of the legacy system and the security framework of the target standard protocol. As an example, they point out the differences in the security policies and enforcement mechanisms between Unix and CORBA. Unix authentication is based on user-password authorization. CORBA uses Kerberos-based authentication. The Unix file system uses the access control based on user, group and world. CORBA uses access control based on owner credentials and service controls.

Schaumont and Verbauwhede [13] take a different approach than rings to integrate security into a software-based communication system. Using the 3G mobile telephone as their area of application, they view the security concerns of the system in the form of a security pyramid. From top to bottom, the layers are circuit, microarchitecture, architecture, algorithm, and protocol. The architecture level handles the mapping of the algorithms and security protocols into the system. The algorithms and protocols are Java applications using Java cryptographic extensions.

Ioannidis, Bellovin, and Smith [14] propose the use of fine-grained access controls at the level of individual data objects. All data objects arriving from remote sources are tagged with a non-removable identifier. This identifier dictates its permissions and privileges rather than the file owner’s user ID as in a standard Unix system.

7. CONCLUSION

The importance of computer system security demands better security-centric software architectures. Warden’s five-ring model provides a way to portray a computer system as an enemy system. This modeling technique identifies the centers of gravity that need the most security protection. In addition, it proclaims the need for layered defenses against computer security threats through the application of a security-centric ring-based software architecture.
8. REFERENCES


