Chapter 4 – Attack Surface Reduction

Tom Olzak, MBA, CISSP
February 2012
InfoSec Institute

In previous chapters, we examined risk assessments and architecture design from 50,000 feet. However, as practitioners, we need a better understanding of the details included in risk reduction and architecture considerations. What steps must we take to ensure an attacker has few opportunities to reach her goals?

The foundation of acceptable risk is a minimized, monitored, and managed attack surface (AS). The process of achieving this state is attack surface reduction (ASR). ASR closes all but required doors leading to system assets and constrains others with access rights, monitoring, and response.

Most of the following is based on the work of Howard, Pincus, and Wing (2002). I modified and added to their relative attack surface quotient (RASQ) analysis to make it less academic and more practical for daily application.

ASR Objectives

We spend considerable time and resources identifying and eliminating vulnerabilities. Further, we require our vendors to produce bug-free code—and look at how well that is going… There will always be vulnerabilities.

Vendors, customers, and the security professionals must come to understand that any software solution, appliance-based or not, ships with known, unknown, or future vulnerabilities (Manadhata & Wing, 2010). Our job is to eliminate or minimize pathways leading to them and mitigate the damage caused if exploited. According to Manadhata and Wing, "A smaller attack surface makes the exploitation of vulnerabilities more difficult, and hence mitigates risk" (p. 1). We achieve this by minimizing exposed system targets, controlling system and network segment access across the network, and enforcing least privilege for all security subjects.

Figure 1 is a conceptual model of an aggregate attack surface model. Aggregate because, although it is the system attack surface with which we are most concerned, various pre-system access controls reduce both the opportunities to reach a system and the number of system elements an attacker can actually see or use. The amount of time and effort in ASR activities is system- and data-classification dependent.

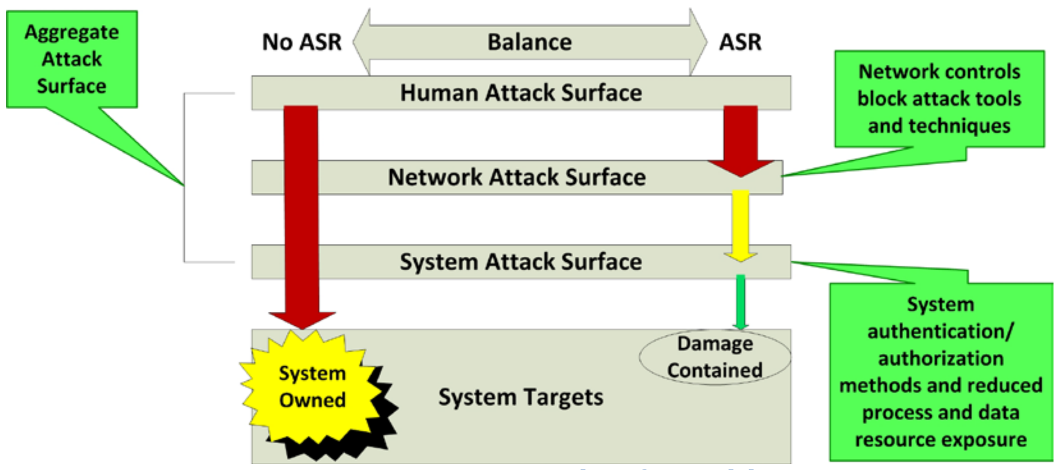

Figure 1: Aggregate Attack Surface Model

Definitions

Some of the terms and concepts in this chapter are new to many readers. If you are already familiar with RASQ, skip this section. If not, it is important to familiarize yourself with them before attacking the following ASR processes.

- **Attack**. Unlike many other approaches to attack modeling, Howard, Pincus, and Wing (2002) see an attack as a sequence of actions beginning and ending with a specific system state. See Figure 2. It resembles a root cause analysis assessment, and that is exactly what it is. Post- or pre-attack analysis using this approach can help identify gaps in the attack surface or supporting controls.

  Note that a specific set of conditions must be present in order to successfully perform an action to arrive at the next set of conditions: system state. The best way to minimize the attack surface is to deny an attacker either the initial system state or any one of the requisite intermediate states.
- **Vulnerability.** Usually, we consider vulnerabilities as simple weaknesses in a system. However, this is not detailed enough. For ASR purposes, a vulnerability is something in the ACTUAL behavior of the system that deviates from the INTENDED behavior (Howard, Pincus, & Wing, 2002). For example, an eval function in an application executes as expected, but the attacker provides its input. The developers and designers intended neither the attacker's input nor the resulting system state.
- **Attack surface.** The AS is the aggregate of all vulnerabilities and controls across all systems and networks. It is the collection of targets exposed to an attacker.

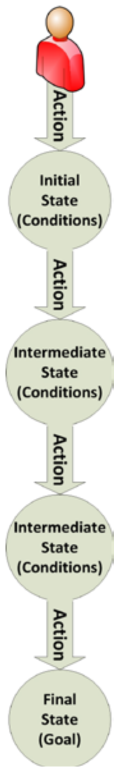
Action
Initial State (Conditions)
Action
Intermediate State (Conditions)
Action
Intermediate State (Conditions)
Action
Final State (Goal)

Figure 2: Attack

System ASR

To maintain consistency with common ASR terminology, we define systems in this chapter as any endpoint device. All ASR activities have one objective: prevention of access to *systems* for the purpose of using them in unintended ways or for untended purposes. Subsets of systems include servers, desktops, laptops, and other network-connected endpoint devices. To simplify system ASR, we break down an AS into three dimensions (Howard, Pincus, & Wing, 2002):

1. Processes and data resources and the actions we can perform on them
2. Channels and protocols
3. Sets of access rights

Processes and Data Resources

Processes are actions we can execute on a system. Remember, an attack is a sequence of actions performed within required sets of conditions. (See Figure 2.) A process can be a target or an enabler.

- **Target**. A target is a something the attacker must control to reach his goal. A target is either a process or a data resource,
- **Enablers**. An attacker uses an enabler to create conditions necessary to reach, modify, destroy, or manipulate a target.

In some types of attacks, a process might be a target; in others, it might be used as an enabler. Examples of processes include

- Browsers
- Mailers
- Database servers

A data resource contains information important to reaching a target or might actually be the target. Examples of data resources include

- Files
- Directories
- Registries
- Access rights (ACLs, account files, etc.)
- Shares
- Virtual directories
- Databases
- Configuration files
- Key stores
- Static and dynamic Web pages
- Audit logs
- Volatile and non-volatile memory
- Directory services

Executables

Executables are a subset of data resources. They fall into two categories: eval sources and carriers.

Eval sources

Eval sources contain eval functions. Many programming languages include them. They accept input from external sources and cause it to execute as if it was part of the primary application. Examples of primary applications include

- Browsers
- Mailers
- Applications
- Services
- Web handlers
- Add-on DLLs
- ActiveX controls
- ISAPI filters
- Device drivers
- Helper applications (e.g., scripts)

Unless you have conflicting evidence, the best approach is to assume any executable contains an eval function.

Carriers
Attackers often embed executables in carriers, including

- Viruses
- Worms
- Trojan horses
- Email messages

Channels and Protocols
Attackers use channels and protocols to reach and manipulate processes and data resources. They fall into two categories: message passing and shared memory.

Message Passing
Message passing components establish sessions between devices and successfully exchange packets. The endpoints of message passing channels are processes. Common channels and protocols include

- FTP
- TCP
- UDP
- HTTP
- Streaming
- RPC connections
- Named pipes

Shared Memory
System components allowing storage and retrieval of information are shared memory, including

- Files
- Directories
- Registries
- Volatile memory
- Removable media

An attacker can also use a channel as a data resource. For example, scanning systems via various channels might reveal open doors through attacks may be launched.

Set of Access Rights
The final system AS dimension is access rights. Access rights identify subjects, the objects they can access, and what they can do after access is granted. Figure 4

places these concepts within an ASR context. Subject A is able to access and manipulate the initial object. However, access rights restrict the subject itself. For example, if the object is a process, it can only reach one of the other objects once-removed from Subjects A and B. Access rights might apply to processes or constraints on channels or processes. They not only prevent initial access; they also set boundaries on unauthorized use of compromised targets and help prevent the conditions necessary for the next phase in an attack.

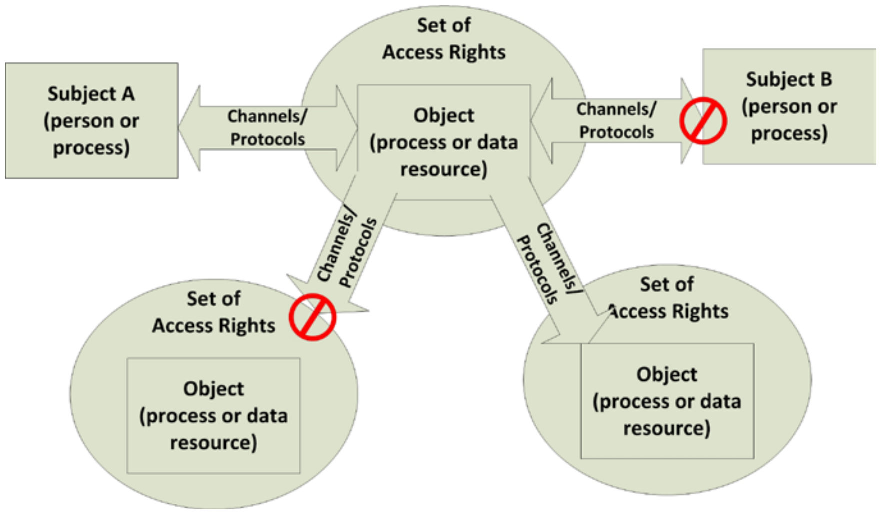
Set of Access Rights
Subject A (person or process)
Channels/ Protocols
Object (process or data resource)
Channels/ Protocols
Subject B (person or process)
Channels/ Protocols
Channels/ Protocols
Set of Access Rights
Object (process or data resource)
Set of Access Rights
Object (process or data resource)

Figure 3: Set of Access Rights

A simple formula provides additional insight into the relationship between the three dimensions. A system attack surface is the total of all *exposed* processes/data resources and *allowed* channels/protocols as *constrained* by the existing set of access rights.

$$AS = \frac{\text{Processes/Data Resources} + \text{Channels/Protocols}}{\text{Set of Access Rights}}$$

Reducing the System Attack Surface

Eliminating a system's attack surface is simple: expose no processes or data resources and shut down all channels and protocols. However, this would not please management... we need a better plan.

System ASR begins with the operating system and all low-level components: including firmware. In Chapter 8 we investigate the universal extensible firmware interface (UEFI) and how it helps ensure a trusted system foundation on start up. It also allows operating system module and driver verification. This is a valuable feature, but it might still leave many doors open. Remember, there are always unknown and future vulnerabilities, even in the most trusted systems.

Patch

As always, the first step in system ASR is applying all available security patches. Yes, there will always be vulnerabilities, but applying available patches constitutes due diligence. Why would we ignore attack opportunities when they are known and easily eliminated?

Shut Down Unneeded Channels and Protocols

Channels and protocols are open doors inviting anything malicious to come in and make itself at home. This is part of hardening a device and includes limiting exposed

- Ports
- Protocols
- Services
- Application interfaces
- Links to file systems
- UI elements
- Inter-process communication points
- Public methods applications can call during execution

Most servers have a specific purpose. In a Microsoft environment, installing the appropriate server role is a good start. Using server roles automatically limits exposure to the minimum required. For example, if you want a server to simply print, you use Server Manager to install the operating system with the appropriate feature set. For more information on Windows Server 2008 R2 server roles, see the TechNet article at http://technet.microsoft.com/en-us/library/dd283014(WS.10).aspx.

Follow this analysis of the server's baseline security with the Microsoft Security Baseline Analyzer (MBSA). This confirms that only the channels and protocols you need are exposed to processes outside the server. Once you are satisfied, either create an image of the server or a baseline security template. Using an image or template helps enable configuration of future same-role servers with the same level of trust.

Secure System Trust Boundaries

Trust boundaries are an extension of the AS. They strengthen it while helping keep threat actions from reaching vulnerabilities. System trust boundaries exist both internally and externally. A trust boundary (TB) separates two processes or two systems. Before sensitive data can pass or critical actions take place through the boundary, a trust relationship is necessary. Establishing a trust relationship requires verifying identity and acting in accordance with the appropriate set of access rights.

Internal Trust Boundaries

An internal trust boundary might appear as in the model in Figure 4. Each system process runs within the security context of the user who started it or is constrained by the set of access rights given specifically to the process. Services and other processes should allow use of a dedicated account ID and password. The account is assigned privileges according to actions performed by the service and resources accessed.

Using only identity-verified and access-constrained processes is the cornerstone of internal trust. For example, a foundation of system trust results from using only UEFI verified drivers, operating system modules, and application components. Internal process trust management helps ensure processes only react to requests from other trusted processes

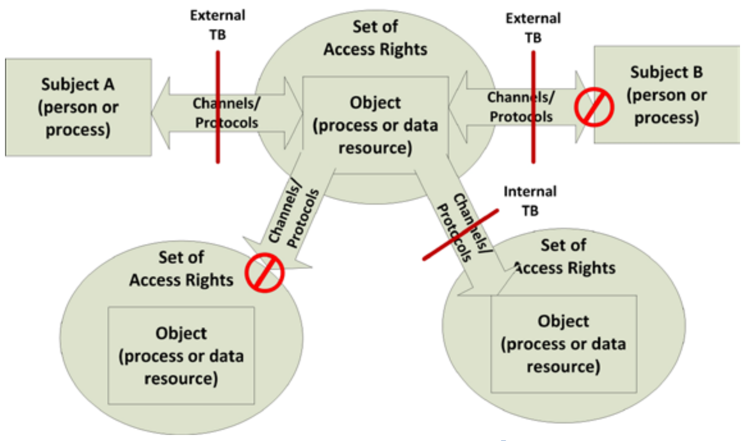

Figure 4: System Trust Boundaries

Much of the work required to create process trust relies on secure coding. Microsoft MSDN (2011) provides guidance about how to establish a process security context:

> You can use a token to specify the current security context for a process using the *CreateProcessWithTokenW* function. You can specify a security descriptor for a process when you call *CreateProcess, CreateProcessAsUser,* or *CreateProcessWithLogonW* function (para. 3).

Regardless of how processes are designed, one principle should underlie all development and implementation: least privilege. No process should have more rights and permissions than are absolutely necessary to perform intended tasks. This significantly reduces the attack surface and helps contain damage when an attacker exploits a vulnerable process.

External Trust Boundaries

External trust boundaries exist between systems, network segments, organizations, and across user interfaces (UIs). Requests for services or data from outside a system are a required component of the action/condition chain required for malicious control of a target. Ensuring strong trust boundary controls exist is a good way to block access to system vulnerabilities.

System to system

The first consideration in system-to-system trust analysis is whether external systems have resolved their own trust issues. See Figure 5. In Figure 5A, each server has a reduced AS. However, Server A's attack surface is smaller and closely managed. If Server A connects to Server B, Server B extends Server A's attack surface. In this case, the extended AS potentially elevates System A risk to an unacceptable level.

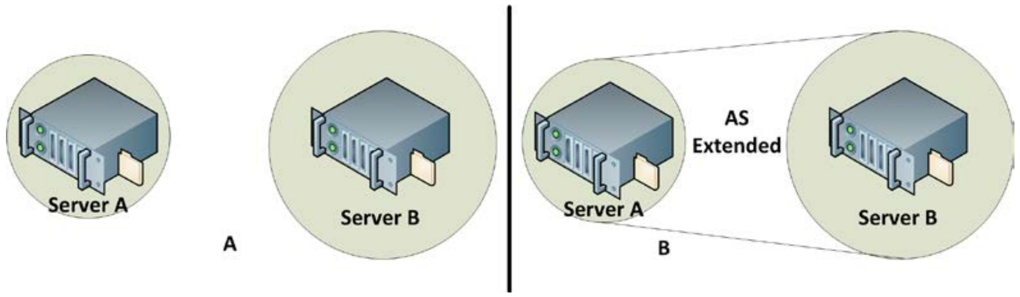

Figure 5: Extending the Attack Surface

Anytime two systems connect, whether end-user or enterprise, they share a common attack surface. In our example, the degree to which risk is increased in Server A largely depends upon the set of access rights Server A grants processes in Server B. This requires consistent and business-balanced trust management across all network-connected devices.

In addition, Server A and Server B should require identity verification (e.g., certificates) before establishing common sessions. The same is true of Web services or other processes that might request use of a process or data resource from another system. Again, least privilege enforced with authentication and access rights is necessary.

Finally, anti-malware, host-based firewalls, and other local security controls help establish the trustworthiness of each system.

Network segments

A best practice approach to creating trust boundaries between sets of systems includes network segmentation. Because Chapter 5 addresses this concept in detail, we only take a high-level look here, addressing how it helps create attack surface protection.

In a flat network model, no segments exist to support blocking unwanted packets from reaching connected servers. This does not mean that all systems can establish sessions with each other. What it does mean, however, is that any device reaching the data center network can potentially locate, scan, and attempt to manipulate processes and access data resources on any connected server it finds.

Figure 6 depicts simple data center network segmentation. A layer 3 switch creates two VLANs and controls traffic to them using VLAN access control lists (VACLs). In addition, IPS devices monitor for anomalous behavior in case an attacker "gets lucky." In effect, creating a network segment creates an intermediate AS between a system AS and potential malicious intent.

A VACL is useful for preventing anything unintended from reaching servers. For example, if a VLAN contains only Microsoft SQL Server instances, system ASR might dictate that we only allow packets using TCP Port 1433 to enter. We might further restrict port 1433 access to one or more explicitly approved application servers.

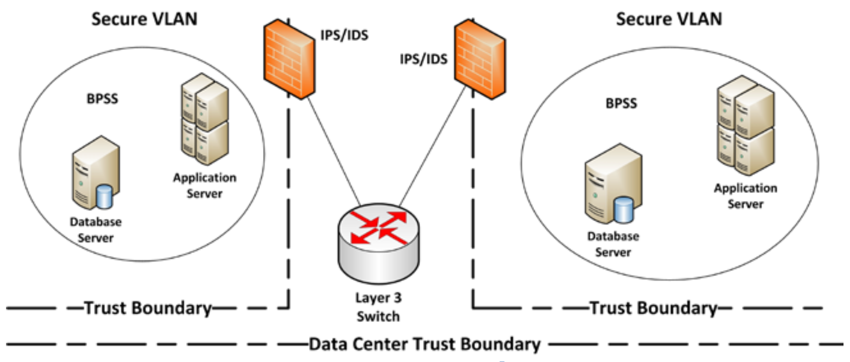

Figure 6: Data Center Network Segmentation

Providing one or more intermediate attack surfaces, including the enterprise network edge and a data center trust boundary, assists with denying malicious access to explicitly open processes and data resources. Further it reduces the probability that processes inadvertently exposed during change management activities, or because of missed steps in system builds, become part of an attack.

Trust begins in an organization and extends to external business partners, customers, mobile employees, and cloud services. Again, trust begins at the system/component level. Figure 7 depicts the process of adding reduced AS devices to a business process system, resulting in an aggregate AS and level of trust.

Aggressive ASR, process and application integrity, and a combination of prevention and detection controls create each system's trust level. IT places systems in network segments with their own trust levels. Consequently, a system's attack surface becomes a function of the network AS and its own.

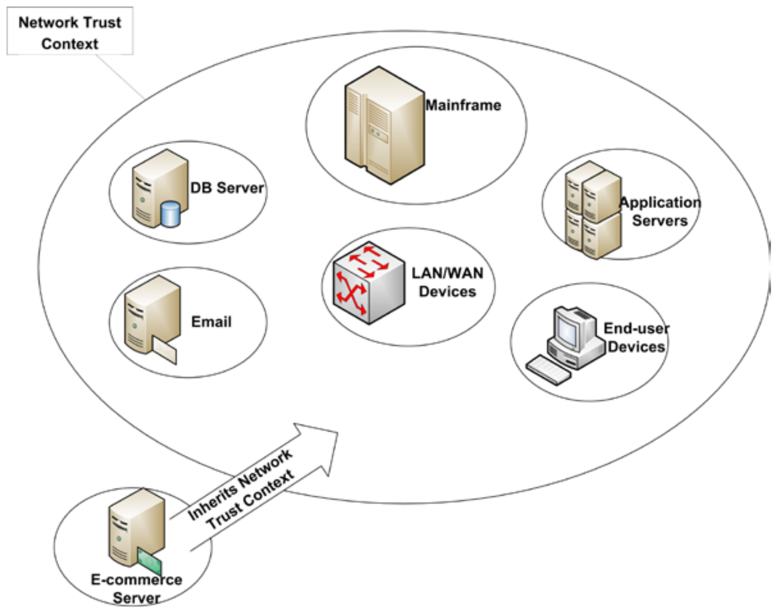

Figure 7: System and Network Trust Contexts

Organization to organization

Today's businesses often must communicate electronically with one or more external entities. Connections between different organizations' enterprise networks are common. When an organization establishes such connections, its attack surface is extended to that of the external entity, with a trust boundary between them. Two actions help retain an acceptable AS: assurance of the external entity's expected levels of trust and strong trust boundary management.

Assurance of trust requires strong agreements and proof of compliance. For example, an agreement might require the external organization to take specific steps to reduce its attack surface and limit potential attacks across the mutual connection. Based on a former U. S. president's approach to arms control, *trust but verify*, it is our responsibility to ensure compliance with the agreement. Proof might be provided by our own audits, SSAE 16 certification, etc.

Trust boundary management requires the same due diligence applied to internal trust boundaries. A firewall blocking all but expected traffic is a good start. Running all incoming traffic through an IPS is a supporting control.

The Human Attack Surface

Regardless of how many technical controls we implement, no matter how few processes and data resources we expose, authorized users can allow attacks with a click of the mouse or an unguarded telephone conversation. Enforcing least privilege, separation of duties, and need-to-know helps limit damage, but it cannot prevent it altogether.

Human ASR requires security awareness training. Awareness training helps focus the attention of employees on behavior necessary to maintain security policy compliance, thereby assuring data confidentiality, integrity, and availability (Olzak, 2006). An organization developing an awareness program should consider including

- The definition of information security and why it matters
- Where to find the organization's security policies, especially acceptable use, and how to translate management's expectations to practical, day-to-day behavior
- Procedures to follow to help protect sensitive information assets
- Regulations that apply to the business and what they mean to their roles
- The potential effect of a major security incident on the business

Three security awareness audiences exist within an organization: business users, business managers, and IT staff. Business users require basic awareness instruction. Managers require basic instruction and additional information about how to enforce policy. Finally, technical standards and guidelines combine with basic instruction to train IT personnel to create systems with attack surfaces, trust, and risk at levels consistent with management's expectations.

Perform a Risk Assessment

Once you think your attack surface is as small as you can get it, and appropriate controls stand guard at all critical trust boundaries, you should test to ensure you have not missed anything. Testing involves the risk assessment process included in Chapter 2. Including processes, data resources, channels, and protocols addressed earlier, the following is a list of common attack vectors relevant to Microsoft environments (Wing, Howard, & Pincus, 2003, slide 5).

- Open sockets
- Open RPC endpoints
- Open named pipes (should never exist…)
- Services
- Services running by default
- Services running as system
- Active Web handlers
- Active ISAPI Web pages
- Executable vdirs
- Enabled accounts
- Enabled accounts in admin group
- Null sessions to pipes and shares
- Guest account enabled
- Weak ACLs in file system
- Weak ACLs on shares

- Weak ACLs in registry
- VBScript enabled
- Jscript enabled
- ActiveX enabled

Follow your risk assessment with aggressive penetration testing. Schedule regular risk assessments and penetration tests to ensure continued compliance with expectations.

Summary

I summarize this chapter with a list of guiding principles for ASR. They help apply its sometimes-complex aspects. The list is based on the work of Michael Howard (2012).

1. Reduce the amount of running code. Use the 80/20 rule; if 80 percent of the users accessing the system do not need a service or process, do not let it run. If you are the developer, make it the default setting: if the security practitioner, turn it off.
2. Restrict access to network endpoints used by your application to the local network segment or an explicit IP address range. Conversely, consider allowing access to system entry points only for subjects in trusted network segments.
3. Limit access to network endpoints using authentication. Simply validating a subject reduces your system's attack surface.
4. Reduce the privilege under which processes execute. This includes both code written in-house and by third-parties.
5. As you review data flow diagrams and attack trees, look for anonymous threat paths: paths for which authentication or authorization are not necessary. Consider controlling them with authentication and assignment of access rights.
6. Apply the 80/20 rule to all protocols. (See Principle #1.)
7. Define your minimal attack surface early in system or application design and measure it periodically to ensure compliance.
8. If you have a large attack surface, you will spend more time managing vulnerabilities and trying to ensure all code and system configurations are perfect. Again, this is an impossible task.

References

Howard, M. (2012). *Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users*. Retrieved February 11, 2012, from MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/cc163882.aspx

Howard, M., Pincus, J., & Wing, J. (2002, February 11). *Measuring Relative Attack Surfaces.* Retrieved February 8, 2012, from Carnegie Mellon School of Computer Science: http://www.cs.cmu.edu/~wing/publications/Howard-Wing03.pdf

Manadhata, P. K., & Wing, J. M. (2010). An Attack Surface Metric. *IEEE Transactions on Software Engineering* .

Microsoft MSDN. (2011, September 7). *Process Security and Access Rights*. Retrieved February 10, 2012, from Microsoft: http://msdn.microsoft.com/en-us/library/windows/desktop/ms684880(v=vs.85).aspx

Olzak, T. (2006, April). *Strengthen Security with an Effective Security Awareness Program.* Retrieved February 11, 2012, from InfosecWriters.com: http://www.infosecwriters.com/text_resources/pdf/Awareness_Program_TOzak.pdf

Wing, J., Howard, M., & Pincus, J. (2003, December). *Measuring Relative Attack Surfaces (PowerPoint Presentation).* Retrieved February 11, 2012, from Academia Sinica, Institute of Information Science: http://iis.sinica.edu.tw/wadis03/slides/Wing.ppt