Magenic Practices: Application Security

From: The Magenic Center for Excellence
Stuart Williams

Table of Contents

Executive Summary

Application security can affect your entire network. As recent events have shown, companies ignore application security at their peril. Prevention is the key and we will address current application security practices and how Magenic can provide you with the best security tools to help harden your system.

The application security tools utilized by Magenic include the SDL, STRIDE, DREAD, Defense-in-Depth (DiD), Magenic's approach toward security best practices, which include often overlooked security concerns. STRIDE is a security best practice that defines the types of threats faced by an application or service. DREAD assesses and numerically ranks the level of security threat. DiD is a security strategy that utilizes multiple layers of security to help protect the system.

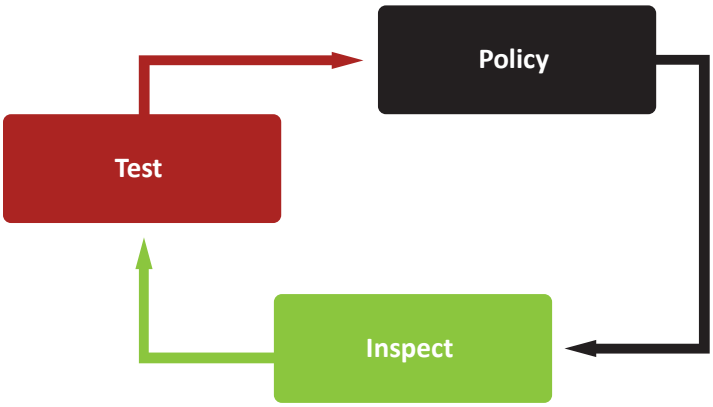
To offer the strongest protections, Magenic recommends that your organizational security goals should address:

- **Policy**
  - Define security goals and practices.
  - Establish governance mechanisms to support those policies.
- **Inspection**
  - Establish and conduct ongoing reviews to ensure compliance to policy, discover potentially new needs, and to handle the inevitable exceptions
  - Magenic has found that manual inspection by skilled persons often is the most

Magenic
Custom solutions that fit. Guaranteed.

effective and economical method of improving quality and security. This is most useful if done as early in the application's lifecycle as possible.

- Review of the architecture before the start of development, code and infrastructure review before production deployment.



- **Testing**
  - Automated security testing to continuously measure conformance to requirements, in order to uncover new vulnerabilities.
  - Testing can be done using security analysis tools, static code analysis, writing explicit security tests and other analytical tools to conform to your security needs.

Magenic specializes in the application development lifecycle including security. Our approach is a modified Total Quality Management (TQM) strategy that assumes that each activity continuously refines the next, evolving over time as needs and environments change. Magenic embraces TQM and considers security an essential facet of quality.

A Constant Challenge

The Security Development Lifecycle, while a powerful tool, can be daunting in its implementation. Microsoft defines SDL as a seven-tiered strategy:

- Security training.
- Assess the requirements necessary for your security development.
- Design the software to encapsulate security best practices.
- Implement software best practices.
- Verify the software is functionally complete.
- Release the software for public use.
- Formulate and implement an incident response plan.

SDL fulfillment can require a great deal of fortitude, time and expense to go forward alone.

Magenic's security staff can guide your team with practical advice and an actionable plan to help harden your applications.

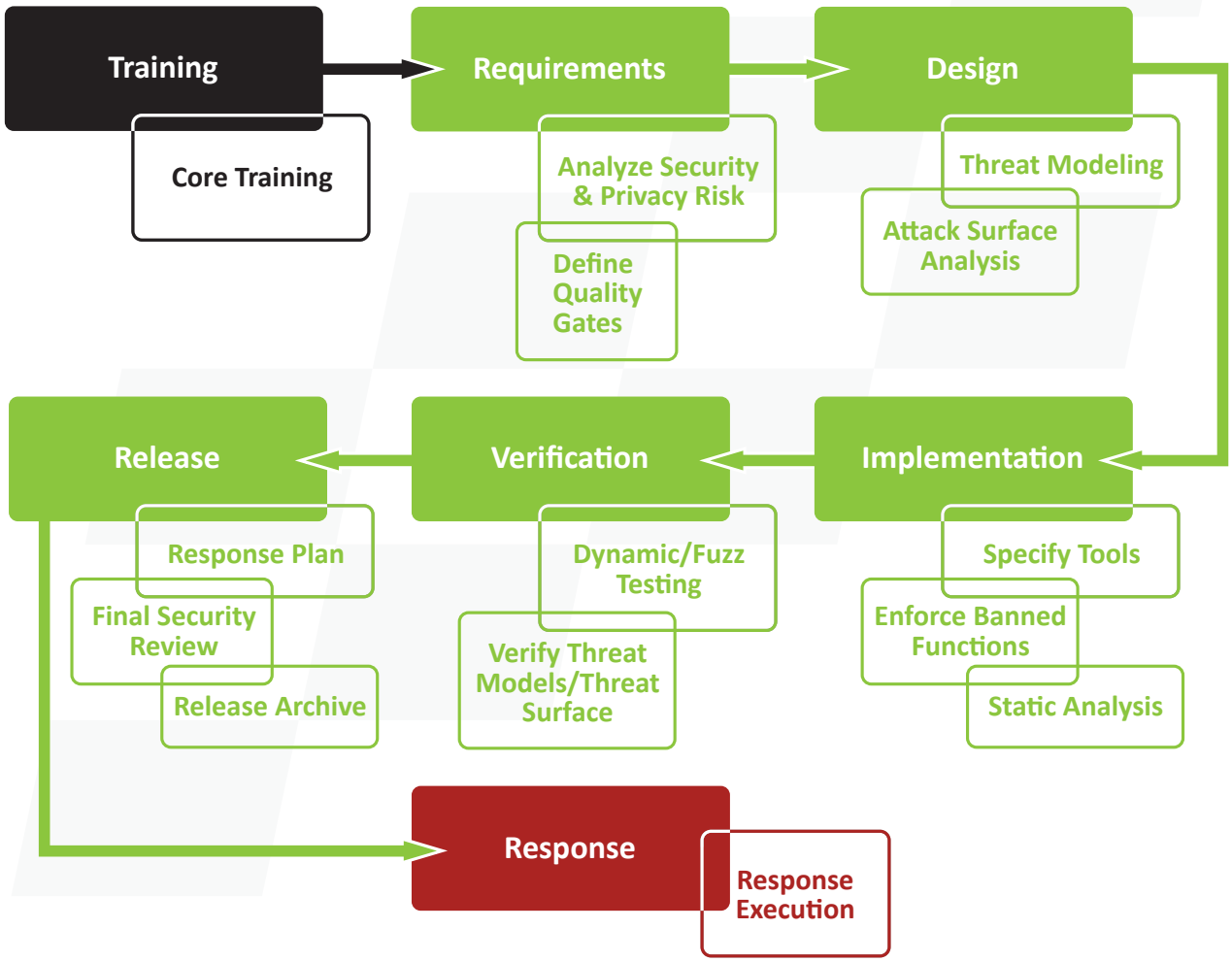
- Security Training
  - All technical team members must have training in software development security. Without training, application security cannot be easily understood or implemented.
  - Implement a security policy for the organization. Identify security practices, training requirements, certification procedures and partner organizations to assist in policy compliance.
- Requirements
  - Consider software security integration and key security objectives.
  - Analyze the trade-off between software security and end-user disruption.
  - Consider and plan for nonfunctional security requirements.
  - Require requirements to be testable.
  - Business team members must be briefed regarding security concerns. Template security questions and requirements must be part of every requirements package.
- Design
  - Threat modeling, via STRIDE and DREAD, identifies the requirements and structure for software design security best practices.
  - Ensure that threat modeling and attack surface analysis are built into every project plan.
- Implementation
  - Establish, follow and enforce software development security best practices.
  - Specify tools. Specific tools are listed in the *Solutions* section.
  - Static analysis FxCop/Code-Analysis and other Microsoft tools provide good analysis of code for potential vulnerabilities. Magenic recommends

Magenic
Custom solutions that fit. Guaranteed.

Magenic Practices: Application Security

implementation of code review by experienced security team members in addition to code analysis.
- Annual team member training on secure coding and other standard practices.
- Automated builds, static code analysis and security testing should be mandatory. All results should be available to relevant team members.

- Verification
  - Dynamic/Fuzz testing throws malformed and bad data at interfaces to see if that data cause problems.
  - STRIDE and DREAD should be updated at this point to ensure that new interfaces or new issues haven't emerged.
  - The release plan should include production environment hardening procedures and infrastructure security certification.
  - For credible verification, QA organizations should be trained in security testing. For each security related requirement, training should include:
    - Specific testing parameters.
    - Procedures to follow should the tests fail or prove to be inconclusive.
- Release
  - Upon public release of your software, response plans should be in place, understood by the entire team and updated upon review.
  - Each release should be archived for retrieval.
  - Ensure that the application, network and platform monitoring are configured correctly to detect the security issues.
  - Notifications and escalations should be well defined and tested.
- Response
  - Several times a year, a response simulation should occur, with all teams leveraged. This allows the team to effectively execute the plan and understand what role and responsibilities each team member plays.



Magenic Practices: Application Security

Solutions

To address your Security Development Lifecycle, Magenic utilizes several core and support tools:

- DiD
- STRIDE
- DREAD
- Application Security Guidance
- Assist IT host in server hardening & network security

Defense in Depth (DiD)

Defense in Depth (DiD) refers to the strategy of not relying on just one layer or chunk of code to protect against a specific threat. No matter what we do to harden applications or services they depend on the operating system and in turn the network. All of these depend on securing the physical locations the hardware resides in. Providing a series of rings that surround our data helps make it harder for attackers to get in. These rings include the following:

- Data Security

At the core of the DiD strategy is Data Security. Data security encompasses the following areas, each having a specific best practice:

- Azure & other cloud storage
- File system
  - Files and folders
  - Registry
  - Assembly security (signing and Authenticode)
- Database
  - Access controls
  - Use of stored procedures
  - Row and Column access filters
  - Object level security

Cryptography can be useful in thwarting certain types of security threats. If implemented correctly the performance hit can be minimal, which is a fair exchange for peace of mind. Since hardware is often the least expensive part of any system, Magenic recommends additional capacity to offset security needs.

- Application Security
  - Building an application that considers security from the outset saves your company time and money. This can best be accomplished by building applications and services securely. Building secure applications and services requires educating the entire team on security techniques and tools.
- Host Security

Securing the servers your applications are hosted on is critical. There is an increasing concern regarding the trend to virtualize applications without considering that the host and guest operating systems also need to be secured. Any security vulnerability in the host will often compromise the virtual machine. It has been widely and incorrectly assumed that if the host OS is secured, the guests will also be.

The single most effective thing that any organization can do to prevent server security vulnerabilities is to regularly accept security patches from the OS vendor. If the organization cannot test the effect of an OS patch on the applications and services within 1-2 days, they should apply them anyway and be prepared to roll them back on the rare occasion that a patch is problematic.

Not all patches should be blindly accepted, only security patches. Security conscious organizations must have a way to apply a series of patches and upgrades to a test environment and then regression test the applications looking for any issues. This ability is key to testing any release of software or configuration change.

Additionally, to assess host environments requires regular vulnerability scans against the machines (hosts, virtual machines, and physical servers) and addressing or patching any issues.

- Network Security

Network security mixes all of the common hardware and techniques for securing a network. An example would be a firewall creating outer, inner and DMZ zones where the servers live. Smart routing, sub-netting and virtual network technologies provide mechanisms to partition a larger infrastructure into chunks, each with differing levels of access and security.

Cryptography can help secure the communication across a network between application servers and clients (channel security). Cryptographic protocols can also be used to secure service payload (data in transit), and data storage (data at rest).

Magenic
Custom solutions that fit. Guaranteed.

Magenic Practices: Application Security

Since cryptographic controls can be exceedingly complicated, Magenic can assist in creating these for your company.

- Perimeter Security

Perimeter security is often the first line of defense, one that is provided by firewalls, reverse proxies, and intrusion detection systems. Since remote access technologies also fall into this category, they must be designed to allow authorized users into the network while making it difficult for others to access network resources.

Magenic recommends perimeter assessments, which are typically conducted via penetration testing (launching a variety of different types of mock attacks) and examining perimeter system configuration.

- Physical Security

The physical security of networks is often overlooked. If the server room is on fire or flooded, your applications are far from safe. Physical security consulting companies can assist in securing the infrastructure. Magenic has many partners we trust to assist in this area. Additionally, the American Society for Industrial Security (ASIS) can provide all sorts of resources that address this topic.

- Policies • Procedures • Awareness

The appearance of security where there is none, is worse than no security at all. It is important that security be an enterprise wide pre-occupation shared by technical and non-technical personnel alike.

Areas of consideration:

- A detailed policy for security with clearly defined roles and importantly a method to get exemptions.
- A cross-functional team to enforce, audit, and when needed change security policy.
- An organization responsible for monitoring the health and security of the infrastructure including security, according to a well-defined set of escalations and procedures for handling different types of threats as they occur.
- A plan to regularly evaluate and assess security.
  - Physical data center security walkthroughs.
  - Network security audit reviews and vulnerability scans.
  - Server, OS and application security scanning.
- IT and Business Education on security topics:
  - Developers: how to write secure code and use security assessment tools.
  - Quality Assurance: how to write and conduct application security tests.
  - IT: How to conduct assessments and monitor the infrastructure for security threats.
  - Project Managers: how to manage the SDL.
  - Business: how make ROI vs. risk.

Magenic
Custom solutions that fit. Guaranteed.

Magenic Practices: Application Security

Application Analysis Techniques

Magenic uses and recommends the following techniques for analyzing applications for security issues and determining the potential threat.

STRIDE

- **S**poofing Identity
- **T**ampering with Data
- **R**epudiation
- **I**nformation Disclosure
- **D**enial of Service
- **E**levation of Privilege

STRIDE defines the types of threats faced by an application or service. STRIDE is considered the best practice with the following advantages:

- Technical and business people can easily understand it.
- DREAD, an excellent assessment tool, is used to support the model.
- If properly executed, it adds significant visibility and value.

STRIDE can assist in reducing your organizations security vulnerability for many of the following threats:

Spoofing Identity

Spoofing is illegally capturing and then using another user's authentication information, such as username and password, to gain access to privileged information.

Tampering with Data

Data tampering involves the malicious modification of data such as:

- Unauthorized changes made to a database.
- Alteration of data as it flows between two computers over an open network.

Repudiation

Repudiation threats occur when users perform prohibited operations within a system that lacks the capacity to trace said operation.

Non-repudiation refers to the ability of a system to counter repudiation threats. Counter evidence for verification, such as a signature upon receipt, for example.

Information Disclosure

Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers.

Denial of Service (DoS)

Denials of service (DoS) attacks prevent service to valid users. This makes a web server temporarily unavailable or unusable by flooding it with bogus requests.

Elevation of Privilege

An unprivileged user gains privileged access and thereby has sufficient access to compromise or even destroy the entire system. Even minor privilege elevations are problematic as they can lead to information disclosure events.

Microsoft has tools to assist teams in creating a STRIDE plan, and for evaluating and acting on threats identified during the process.

Magenic
Custom solutions that fit. Guaranteed.

Magenic Practices: Application Security

DREAD

- DREAD numerically rates the danger of a particular vulnerability.
- The DREAD formula is a simple one. Add up the various threat levels and divide by 5, as outlined below.

Risk = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5

- DREAD is an easy-to-understand and practical threat classification system.
- The traditional calculation always produces a number between 0 and 10: the higher the number, the more serious the risk.

Quantifying the DREAD categories:

Damage Potential

If a threat occurs what is the extent of the damage?	0	None.
	5	Individual records compromised.
	10	Whole sets of data (tables) compromised.

Reproducibility

Can the threat exploit be reproduced?	0	Very hard or impossible, even for administrators of the application.
	5	One or two steps required, may need to be an authorized user.
	10	Just a web browser and the address bar is sufficient with no authentication.

Exploitability

What is needed to exploit this threat?	0	Advanced programming and networking knowledge, with custom or advanced attack tools.
	5	Malware exists on the Internet, or an exploit is easily performed, using available attack tools.
	10	Just a web browser .

Affected Users

How many users will be affected?	0	None.
	5	Some users, but not all.
	10	All users.

Discoverability

How easy is it to discover this threat?	0	Very hard to impossible; requires source code or administrative access.
	5	Can figure it out by guessing or by monitoring network traces.
	9	Details of faults like this are already in the public domain and can be easily discovered using a search engine.
	10	The information is visible in the web browser address bar or in a form.

Case Study

Magenic's security team exposes a Windows Communication Foundation (WCF) SOAP service endpoint to a hypothetical calculation and with a result or a fault. The consumer of the service would then send a request along with some security information—referred to as claims.

Threats

	Type	Threat Specifics	Mitigation
#25	**Spoofing**	Unauthorized service consumers could gain access to the service	▪ Force the claims to be changed regularly out of band. ▪ Monitor service requests and audit regularly to determine if any user has suspicious activity.
#26	**Repudiation**	Consumers can claim they never used the service to avoid paying	▪ Claims are unique to each user. ▪ Claims are changed frequently. ▪ An audit is kept. ▪ Only strong claims accepted.
#32	**Elevation of Privilege**	Requestors could see results that do not belong to them	▪ All input validated. ▪ Claims used to filter results. ▪ Trusted callers.
#33	**Tampering**	Unauthorized persons could tamper with the results from the service	▪ Use channel security (HTTP/S), private certificates and strong cryptography.
#34	**Information Disclosure**	The contents of the reply could be disclosed to unauthorized persons, and the credentials used could be stolen	▪ Use HTTP/S, private certificates and strong cryptography. ▪ Use claims to ensure only rows that belong to the claimant are returned. ▪ Claims are changed frequently.
#35	**Denial of Service**	The service could be bombarded with bogus requests preventing legitimate users from making requests	▪ Use network intrusion detection to remediate DoS attacks.

Below is a sample matrix defined by the above threats:

DREAD Matrix Sample

	Damage	Reproducibility	Exploitability	Affected Users	Discoverability	Score
#25	5	3	3	5	2	3.6
#26	7	5	1	1	1	3.0
#32	9	1	1	5	3	3.8
#33	1	1	1	1	1	1.0
#34	5	3	1	3	2	2.8
#35	3	2	1	8	5	3.8

Together the business and security teams derive the DREAD score, with specific reasoning behind their decisions, such as:

- (#32) Damage = 9 The business has identified that should another user's records be seen by unauthorized persons, the reputation of the company would suffer, resulting in a loss of customers and negative revenue impact.
- (#33) Tampering = 1 The team considers this threat to be minimal because strong encryption, private certificates and strong claims to secure the communication between the service and the consumers are already in use.

Using the results of STRIDE and DREAD

Based on the results of a security review, team members and key decision makers can decide what would be the best direction for potential vulnerabilities. Once decisions have been determined, the various team players could follow through on appropriate actions:

- **Architects**: Suggest architecture, design and implement changes to "harden" the application.
- **Infrastructure**: Can provide recommendations on ways the infrastructure (network, server, and configuration) may be improved.
- **Executives**: Drive the decision based upon what is best for the corporate long term goals.
- **Magenic**: Can assist in stitching together all of your SDL needs.

Analytical advantages and disadvantages

STRIDE/DREAD analysis can provide a great deal of value if properly executed. The execution should be done in conjunction with the Defense in Depth strategy and the other SDL best practices. Magenic can assist integrating the multitude of SDL tools available to secure your business security needs.

A cautionary note:

While some organizations may follow through with the STRIDE/DREAD assessments, they may fail to implement any changes toward fixing the identified security issues. Key decision makers must be made aware of the security assessments, risks and potential costs to the organization if remediation is not followed. Since STRIDE/DREAD is a high-level threat model, overlooking the assessments can potentially jeopardize business continuity and reputation.

Security Domains

Security domains represent the tools, vulnerabilities, and techniques for securing specific parts of the application and environment. While extensive, no white paper can completely represent all available security techniques. The Magenic team can assist in further identifying organizational vulnerabilities and the techniques necessary to secure your applications.

The foremost rule in security domain is the Principle of Least Privilege. Users and machine accounts should always have the least amount of access necessary to accomplish assigned tasks. "Denial of privilege is a kindness in disguise."[2]

Networks

It is beneficial for organizational application architects and implementers to have a cursory knowledge of network security operations. This allows for communication between all the involved teams. Our *references* section in the appendix lists network security manuals that would be of assistance. Additionally, working with an infrastructure partner is strongly suggested.

Servers

Securing an application means securing the servers it runs on. For other infrastructure security steps, such as patches, anti-virus, intrusion detection, etc., the following items should be considered:

- In forward facing (servers exposed to the internet) scenarios, the machines should be run in (be registered in) a separate domain from the enterprise domain. This creates a one-way trust from the outside in, for purposes of deployment and maintenance.
- Active Directory Security (ADS) should be the rule, not the exception. Mixing of username/password combinations between machines runs counter to SDL practices. The credentials of a user can be substituted for an impersonated set to connect the web server to the SQL server or to securely send messages deeper into the application stack. This is known as the "Trusted Subsystem" model, where impersonation may allow unauthorized

[2] Security Manual. Digital Electronic Corporation. In-house publication. Date unknown.

access.

- Carefully removing any service and disabling every port that is not needed, goes a long way toward securing a server. When in doubt, consult the hardware and OS vendors' documents and use their tools to secure the server.
- *Windows Server Security* section in the appendix provides further information.

Server Hardening Checklist

This table is a summary of Microsoft Patterns and Practices guidance for securing a production web server. Magenic provides real world technical advice under the Magenic Solutions column. Most of these items apply to all server types.

Step	Details	Magenic Recommendations
Services	■ Disable any services that are not required. ■ Disable the FTP and NNTP protocols. ■ Consider disabling Telnet, RCP, and other administrative IP services ■ Disable the SMTP protocol unless the server handles mail	■ E-mail should be handled on machines dedicated to e-mail, to isolate them from the rest of the applications. ■ Leverage the existing enterprise e-mail infrastructure.
Protocols	■ Configure the TCP/IP stack to mitigate the threat from network denial of service attacks. ■ Disable the NetBIOS or SMB protocols, but not both. ■ For more Information: Plan security hardening for extranet environments	■ Follow MS recommendations only if intrusion detection and DoS protection are not available in the infrastructure. ■ Tinkering with the TCP/IP stack can have unintended consequences. ■ Thoroughly test to make sure the application still runs correctly and that performance problems have not been introduced.
Accounts	■ Follow the principle of least privilege for all service and user accounts. ■ Delete or disable any unused accounts. ■ Disable the Guest account. ■ Rename the Administrator account. ■ Disable the IUSR_MACHINE account. ■ Use a strong password policy. ■ Restrict remote logons to admins.Disable null sessions to prevent anonymous logons unless required.	■ Regularly audit accounts in the production environment and their privileges. ■ Over time, accounts tend to "acquire" additional rights.

Magenic
Custom solutions that fit. Guaranteed.

Magenic Practices: Application Security

Step	Details	Magenic Recommendations
Files and Directories	■ Ensure the IIS anonymous account does not have write access to Web content directories or command-line tools. ■ Remove all SDKs, resource kits, and debugging tools.	■ While this may be good advice, it can be problematic: If write access is removed under file based logging conditions, most logging engines are unable to write log entries. Careful testing and logging configuration may be needed. Removing file based logging unfortunately also removes the "logging of last resort" and any "tracing" logging that can help diagnose issues. If file based logging is used then the file locations should be isolated. A quota should also be set to keep the disk consumption to a manageable level. Most logging libraries (like Enterprise Library) and system logs have mechanism to "roll over" log files at intervals to allow for old log files to be purged. The same can be said for SDKs etc. When removed, the ability to diagnose issues is likewise downgraded. So it's a trade-off that must be carefully considered.
Shares	■ Remove all unnecessary shared folders. Restrict access to any required shares.	■ Log files should be "swept" (copied) to a central place for retention & analysis. ■ The fewer shares the better. ■ Regularly audit both access to shares, permissions and accounts.
Ports	■ Limit Internet-facing open ports (to 80, 443) ■ Use IPSec to encrypt or restrict intranet communication ■ For More Information: Deploying IPSec	■ Again, less is more. ■ In cases where web services are exposed, other ports besides 80 and 443 will need to be left open. ■ IPSec can be tricky to configure, but is useful in situations where there is a concern about traffic between servers being compromised. Notice that IPSec is not appropriate in all scenarios. An example would be, the client to the web server or from the web server to SQL server SSL is preferable. ■ Use Kerberos instead of NTLM where Windows authentication is required for external users. Kerberos uses one port (88 UDP + TCP) vs. NTLM requiring a range of open ports.
Registry	■ Restrict remote administration of the registry. ■ Secure the Security Account Manager (SAM) database to protect user credentials.	■ Registry security can sometimes interfere with application functionality and logging. ■ Use caution whenever working with the registry.

Magenic Practices: Application Security

Step	Details	Magenic Recommendations
Auditing and Logging	■ Log and monitor all failed logon attempts. ■ Log all failed file system actions. ■ Store the IIS log files separately from your Web site, and restrict access to the files. ■ Archive log files on a different server or backup media at least every 24 hours. ■ Audit access to the meta-base file.	■ Managing log files on a separate physical drive is also good practice, specifically for performance reasons.
Sites and Virtual Directories	■ Move your Web site to a non-system drive / partition. ■ Remove or secure Remote Data Services (RDS). ■ Restrict Web permissions in the IIS metabase. ■ Remove FrontPage Server Extensions.	■ For performance reasons alone, webs should not be on the system drive.
Script Mappings	■ Map any unused file extensions to the 404.dll.	
ISAPI Filters	■ Remove any unused ISAPI filters.	
IIS Metabase	■ Restrict access to the IIS metabase; only the Administrators group and the Local System account should have the Full Control permission level. ■ Restrict the banner information that IIS returns in HTTP response headers.	
Server Certificates	■ Ensure that your digital certificates for SSL are valid and up-to-date.	■ Many organizations do not obtain their root certificate from a reliable source or the certificates are allowed to expire. ■ Enterprises who choose to be their own certificate authority should ensure they are well versed in managing certificates and regularly audit certificate use, expiration and renewal. ■ Expired or untrusted root certificates are also a usability issue.

Magenic
Custom solutions that fit. Guaranteed.

Step	Details	Magenic Recommendations
Config Files	■ Map any unused .NET Framework file extensions to System.Web. HttpForbiddenHandler. ■ Disable remoting on Internet-facing Web servers. ■ Ensure that tracing is disabled. ■ Ensure that debug compiles are disabled. ■ Ensure that ASP.NET errors are not returned to clients.	■ Disabling debug builds (in configuration) can make figuring out what went wrong problematic. ■ Carefully consider the trade-off. Most of the exploits involving the debugger require that the attacker "own" the server first.

Microsoft Products

SQL Server

Key SQL server security issues that Magenic recommends utilizing:

- Microsoft suggests disabling the standard ports 1433 (SQL) and 1334 (SQL Resolution) to make it harder for malicious attackers to find the SQL server. While this may prevent some security problems, it is best to remember a key security axiom: **OBSCURITY IS NOT SECURITY**.
- In all cases, SQL ports should not be forward facing.
- Do not allow SQL Security, only use integrated (AD) security
- Turn off all of the protocols not being used to connect to the SQL server
- Turn off all unneeded SQL services
- Consider isolating SQL Reporting services to their own servers
- Consider using SSL to secure connections to SQL server
- Consider that symmetric cryptography performs better than asymmetric methods and may be acceptable in all but some exceptional cases.
- Disable anonymous access to SQL services. To accomplish this, set the key to:
  RestrictAnonymous = 1
  This key is in the Windows registry located at: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA
- Only put SQL files on NTFS or other secure file systems (e.g. not FAT*).
- Restrict or avoid access to tables, prefer views and stored procedures for all application data access in most cases.
- Disable public, sa, etc. accounts and groups.
- Assign appropriate access rights and group membership. In general, the Rule of Least Privilege should be followed.
- Consider the trade-off in enciphering individual fields (either from within or at the application level) vs. securing an entire table or database.
- Avoid 'SETUSER' in favor of 'EXECUTE AS.' 'SET USER' determines the context for multiple commands until it is turned off. This can lead to inadvertent elevation of privilege or information disclosure.

Application Security Guidance

Magenic's guidelines toward application security:

- In situations where a password is necessary, length beats strength. System implementers should be encouraged to use pass phrases instead of traditional passwords (do not disallow spaces through policy).
- This offers several advantages:
  - Pass phrases are more difficult to crack.
  - Users are more likely to remember pass phrases. This decreases the likelihood of written password reminders, thus increasing your network security at no additional cost.
  - Pass phrases also support multi-lingual situations. Use of the strongest one-way hashing at the earliest tier possible will help prevent disclosure.
  - Combining the password with some other root such as the username or e-mail address prior to hashing

Magenic
Custom solutions that fit. Guaranteed.

prevents the same passwords from having the same hash when stored.

- Do not "roll your own cryptography". The Windows platform provides strong cryptography nicely hidden by .NET and made significantly easier to use via the Patterns and Practices Enterprise Library Blocks listed under *Application Security* in the Appendix. Writing cryptographic algorithms is best left to advanced mathematicians and cryptographers.

  Do not use "secret" words embedded in code, such as hard coded "salts" for cryptographic uses or "home-made hash strings" for scrambling algorithms.
- Consider using .NET secure string class to protect text in memory.
- Secure your configuration once deployed. The .NET system configuration library transparently handles the cryptography if the correct method is used to secure the configuration files.
- Consider wire security:
  - IPSec between servers
  - SSL between client and web server, web server and SQL server
- For web services, additional enciphering of the payloads is desirable and may be required for compliance to enterprise or government guidelines.

Authentication

Authentication is essentially the need for computer systems to verify that the user is authorized to utilize the program or system. Passwords would be one example of authentication. The direct links are listed under the Authentication heading.

Web Applications and Services

Further guidelines:

- Error messages, stack traces, etc. to the end-user, can be avoided using the following techniques:

  Leverage global.asax.cs to make a last resort error handler that deflects users to a "sorry page".

  Disable showing errors to users in IIS.

  Configure error handling (specifically 5xx HTTP server errors) in the web config. Custom errors should always be enabled for web applications and services.

  For WCF services, always turn off `<IncludeExceptionDetailInFaults = false>` either in code or configuration.

  For more information see Web Security Resources in the Appendix.
- Turn off tracing in production: `<trace enabled="false" localOnly="true">`
- Disable debug compilations `<compilation debug="false">`
- Disable client side cookie access `<httpCookies httpOnlyCookies="true">`
- Carefully pick the correct authentication model.
- Always use stored procedures for data access to avoid SQL injection attacks.
- Validate user inputs server side even if you did client side validation. A malicious entity can bypass your page entirely and try and send directly to your code behind. No data from the user should be trusted even if the transport is HTTP/S.
- Avoid cookie-less session design. Using cookie-less sessions opens a security hole because the session token is transmitted in the URL allowing for a request to be high-jacked and the user session "spoofed".
- Generally impersonation should be avoided for performance and stability considerations unless the following criteria are met:

  Integrated (AD) security is used.

  Fine-grained permissions are required throughout all of tiers for auditing purposes and or access control secured by Access Control Lists (ACL).
- Implement anti-cross side scripting protection (For more information see the next section—Web Security Resources or the Appendix under *Web Security Resources*).

Web Security Resources

- OWASP Top 10 Guidance
- Alik Levin's Web Security cheat sheet
- Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication (Vintage but sound information)
- Improving Web Services Security: Scenarios and Implementation Guidance for WCF

Windows Applications

Securing Windows applications can be problematic, as the developer or administrator has little control over applications running on a client machine. This exposes the business logic and running state to localized operation, which in turn makes it more vulnerable to malicious use.

As web applications increase in use, deployment and security issues will decrease. Until this is the norm, developers and administrators must consider the impact the deployment of a Windows application will have on the user's desktop from both a security and stability perspective. To fully discuss Windows application security, it is beneficial to observe various Windows application scenarios that have been secured.

Windows Application Scenarios

Windows applications utilizing best practices:

- The application is stand-alone and rarely needs external resources that are not secured by the local user's credentials. Applications of this type generally need access to the local file system to be useful.
- The application supports disconnected users, which only connect periodically to the network. These types of applications allow caching of states, synching changes and resolving concurrency conflicts when the client is connected. Typically the applications are built to support team members in the field. This application type is also popular on non-PC form factors such as tablets or smartphones.
- The application provides "presence" information, such as chat or instant messaging.
- The application requires a very rich user interface, especially if it is associated with local state and large data sets, such as cached tab content or large table editing.
- The application requires the local presence of lots of data. The presence of local caching alleviates demand on the wire. Teams should carefully consider if this scenario is acceptable. Users rarely want to page through lots of data.

Windows Application Security Concerns

Magenic recommends the following best practices when building Windows applications.

- Consider signing all assemblies. .NET has two different kinds of assembly signing, either or both can be used for different purposes:
  - Strong name signing helps prevent assembly tampering and is required to put an assembly into the Global Assembly Cache (GAC). Strong naming is the foundation of code access security. This allows developers or operators to specify the permissions and functions that the code is allowed to execute.
  - Authenticode signing assures users that the assembly is from the organization it claims to be.
- If the configuration contains sensitive information such as connection strings, the configuration sections should be enciphered.
- All .NET applications should have the static code analysis security rules run and addressed.
- Always proceed based on the assumption that once code is on a user's machine, it is vulnerable and there is little control over how it will be used.

Windows Service

As with other application types, SDL best practices should apply to Windows services. Service security follows what identity the service runs and what rights are assigned to that identity.

For more information, see The Services and Service Accounts Security Planning Guide in the appendix.

Security Tools

Magenic uses and recommends these security tools:

- General Windows Security
- SDL Tool
- Web Security Tool
- FxCop/Code Analysis
- Anti-Cross Site Scripting (XSS)
- XSS testing
- Microsoft Security Assessment Tool 4.0
- Microsoft Baseline Security Analyzer
- Microsoft Application Compatibility Toolkit (ACT)

Magenic
Custom solutions that fit. Guaranteed.

Summary

From the earliest days of computing, security has been a concern. As governments and enterprises put more and more of their business onto computing platforms, the risk has increased. For applications, the best practice is the Security Development Lifecycle . Magenic has been helping customers plan, write and deploy more secure applications for over sixteen years.

Utilizing Defense in Depth, our security team can recommend a layered strategy to manage risk. Magenic has thought of and utilized a multitude of ways to protect data security, file systems, databases, application security, host security, network security, perimeter security, and physical security.

By applying STRIDE and DREAD Magenic can analyze your applications for security issues to determine potential threats. These threats are ones that can slow or even stop your business from functioning, if they aren't handled in a proactive manner.

Magenic can support your network security and server hardening by working with your infrastructure teams to help deploy and secure applications. Additionally, we collaborate with your infrastructure security partners to help harden your networks.

Application security is a minefield of small details. With the vast experience of Magenic's security team, we know where to look for the proverbial needle in the haystack. Such needles may include something as simple as password protection or as complicated as cryptographic algorithms. Either way, such minutia searches often need Magenic's security team to assist your security team.

When you are addressing your businesses Security Development Lifecycle, Magenic's security team is there to help. We have custom solutions to your challenges; solutions that cause you to question how you ever did without Magenic.

Appendix

References

Bragg, R, Rhodes-Ousley, M, & Strassberg, K. (2004). *Network Security: The Complete Reference.* McGraw-Hill Osborne Media.

Cole, E, L., R, & W., J. (2009). *Network Security Bible*. John Wiley & Sons Inc.

Windows Security Server

Microsoft. (2010). *Microsoft Security Compliance Manager.* Retrieved from: http://www.microsoft.com/downloads/en/details.aspx?FamilyID=5534bee1-3cad-4bf0-b92b-a8e545573a3e&displaylang=en

Microsoft. (2010). The Threats and Countermeasures Guide. Retrieved from: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24696

Microsoft. (2009). Plan Security Hardening for Extranet Environments (Windows Sharepoint Services). Retrieved from: http://technet.microsoft.com/en-us/library/cc287966(office.12).aspx

Microsoft. (2003). Deploying IPSec. Retrieved from: http://technet.microsoft.com/en-us/library/cc737024(WS.10).aspx

Application Security

Microsoft. (2008). *Enterprise Library 4.1*. Retrieved from: http://www.microsoft.com/downloads/en/details.aspx?FamilyId=1643758B-2986-47F7-B529-3E41584B6CE5&displaylang=en

Authentication

Donaldson, M. (2008). *Some Application Security Terminology.* Retrieved from: http://geekswithblogs.net/MainaD/archive/2008/04/03/120994.aspx

Microsoft. (2011). *Authentication (Windows).* Retrieved from: http://msdn.microsoft.com/en-us/library/aa374735%28v=vs.85%29.aspx

Magenic
Custom solutions that fit. Guaranteed.

Web Security Resources

Microsoft. (2011). *Servicebehaviorattribute. includeexceptiondetailinfaults property*. Retrieved from: http://msdn.microsoft.com/en-us/library/system.servicemodel.servicebehaviorattribute.includeexceptiondetailinfaults.aspx

Open Web Application Security Project. (2010). *OWASP Top Ten Project.* Retrieved from: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Levin, A. (2009). *ASP.NET Security Architecture Cheat Sheet for Very Busy Architects.* Retrieved from: http://blogs.msdn.com/b/alikl/archive/2009/03/19/asp-net-security-architecture-cheat-sheet-for-very-busy-architects.aspx

Microsoft. (2006). B*uilding Secure ASP.NET Applications: Authentication, Authorization and Secure Communication.* Retrieved from: http://msdn.microsoft.com/en-us/library/aa302415.aspx

Microsoft. (2008). *Patterns & Practices: Improving Web Services Security Guide*. Retrieved from: http://wcfsecurityguide.codeplex.com/

Microsoft. (2005). *The Services and Service Accounts Security Planning Guide*. Retrieved from: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=5543

Magenic Security Toolbox

Casaba Security. (2011). *Watcher: Web Security Testing Tool and Passive Vulnerability Scanner.* Retrieved from: http://websecuritytool.codeplex.com/

Microsoft. (2011). S*ecurity Tools*. Retrieved from: http://technet.microsoft.com/en-us/security/cc297183.aspx

Microsoft. (2011). *SDL Threat Modeling Tool.* Retrieved from: http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx

Microsoft. (2011). *Security Warnings .NET Framework 2.0.* Retrieved from: http://msdn.microsoft.com/en-us/library/ms182296(VS.80).aspx

Microsoft. (2010). *Microsoft Anti-cross Site Scripting Library v4.0.* Retrieved from: http://www.microsoft.com/download/en/details.aspx?id=5242

Microsoft. (2005). *Testing Your Web Applications for Cross-site Scripting Vulnerabilities*. Retrieved from: http://technet.microsoft.com/en-us/library/cc512662.aspx

Microsoft. (2009). *Microsoft Security Assessment Tool 4.0.* Retrieved from: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=12273

Microsoft. (2010). *Microsoft Baseline Security Analyzer 2.2.* Retrieved from: http://technet.microsoft.com/en-us/security/cc184923

Microsoft. (2010). *Microsoft Application Compatibility Toolkit 5.6*. Retrieved from: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=7352

Microsoft. (2011). *SDL Threat Modeling Tool 3.1.* Retrieved from: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=2955

Microsoft. (2011). *Microsoft Security Development Lifecycle.* Retrieved from: http://www.microsoft.com/security/sdl/default.aspx