

Realtime
publishers

"Leading the Conversation"

The Essentials Series

Messaging and Web Security Volume III

by Dan Sullivan

Article 7: OpenID: Pros and Cons	1
Implementation and Organizational Issues with SSO	1
Advantages of OpenID	2
Disadvantages of OpenID.....	3
Summary	3
Article 8: Overview of XACML.....	4
XACML High-Level Architecture.....	5
Policy Specifications and Decision Rendering.....	6
Summary	6
Article 9: Ajax Security Overview: Problems and Solutions	7
Ajax Security Weaknesses.....	7
Tools and Technique for Securing Ajax	8
Smash—Secure Mashup.....	9
IceFaces—Secure Ajax Frameworks.....	9
Ajax Vulnerability Scanning.....	10
Summary	10
Article 10: Where to Spend Your Security Budget Part 2: Evaluating Security Options.....	11
Limits of Formal Risk Analysis.....	12
An Additional Method for Allocating Resources	13
Summary.....	15
Article 11: Techniques for Detecting Stealth Malware	16
Hiding Malware	16
Methods for Detecting the Presence of Stealth Malware.....	17
Potential Limits of These Techniques.....	18
References.....	18
Article 12: Localized Malware	19
Motivations for Localizing Malware	19
Avoiding Detection.....	19
Improving Click-Through Rates.....	20
Staying Under the Radar.....	20
Examples of Localized Malware	20
Controlling the Impact of Localized Malware.....	21

Copyright Statement

© 2008 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimepublishers.com and the Realtimepublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

Article 7: OpenID: Pros and Cons

OpenID is a single sign-on (SSO) system for Web applications. The standard is not controlled by any one vendor, and services are free to end users. Web developers are relieved of building authentication systems from scratch and can take advantage of standards adopted by the likes of Google, IBM, Microsoft, AOL, and VeriSign. If that were all that were to be said about OpenID, the standard would likely be a raging success; but there is more to the story. Security vulnerabilities, phishing scams, and consolidated authentication information are legitimate concerns.

This article will examine three aspects of OpenID:

- Implementation and organizational issues related to SSO
- The advantages of the OpenID standard
- The disadvantages and potential drawbacks of the standard

Before delving into the advantages and disadvantages of OpenID, it is worthwhile to outline some of the implementation and organizational issues related to SSO.

Implementation and Organizational Issues with SSO

The ability to log in once and use multiple systems is known as SSO. The idea is increasingly popular for Web applications but history demonstrates the difficulty in achieving that objective.

The desire for a Web-based SSO grows with the increasing use of Web applications, especially those personalized for individual users. Systems such as social networking sites, customized newsfeeds, forums, service providers, retailers, and banks and other financial service organizations all require users to authenticate to access their accounts or maintain personalized information. This proliferation tends to lead users to employ a single or small number of username and password combinations for all sites. The need to manage authentication credentials for multiple sites drives the poor practice of reusing passwords across sites. This is not a new problem.

SSO has been a long standing goal in enterprise applications, but interoperability issues have hindered successful SSO. For example, a typical enterprise may have an intranet portal with one authentication mechanism, an enterprise resource planning (ERP) system with another, and a set of databases with yet another method for tracking user identities. Add to this list custom-developed legacy applications, each with their own slightly different authentication mechanisms, and one can appreciate the scope of the problem. One of the largest stumbling blocks to implementation in these environments is the variety of standards involved. Web-based SSO, however, needs only an HTTP-dominant environment in order to function.

In the Web environment, everyone essentially agrees to use the same protocol. Unlike an enterprise, though, there is no central controlling body that can dictate all applications use a single standard. What could have been a tradeoff between relatively easy-to-address technical issues with challenging inter-organization/political issues were resolved by the OpenID framers. Today, we have an authentication framework that is widely adopted and is at least the first step to a reliable and efficient method for Web SSO.

Advantages of OpenID

The OpenID framework provides advantages for both Web application users and Web application developers:

- Users of OpenID will reduce the number of login credentials they need to create and manage. They need to maintain only a single account with an OpenID identity provider, such as VeriSign, Yahoo, and Google. Lists of OpenID providers are available at http://en.wikipedia.org/wiki/List_of_OpenID_providers and at <http://openid.net/get/>.
- An OpenID can be used at a growing number of portals, blogs, and other Web sites.
- The wide support for OpenID is likely to lead to the development of tools, such as VeriSign's OpenID Seatbelt for Firefox (see Figure 1), to make the framework easier to use.

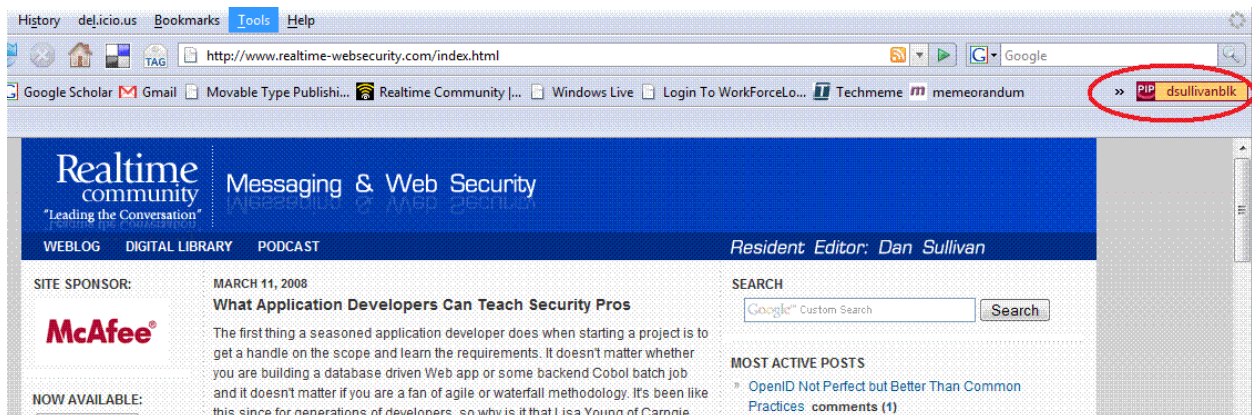


Figure 1: Browser add-ons, such as VeriSign's Seatbelt for Firefox, eases the login process by detecting when the user accesses an OpenID-enabled site, and if not logged in, prompts the user for authentication and then returns the user to the site.

There are also advantages for Web application developers. For starters, anyone with an OpenID can easily use an OpenID-enabled site. Another advantage is that services provided by other OpenID-enabled sites can be integrated into a Web site without requiring a separate login. As with any technology, though, there are potential drawbacks with OpenID.

Disadvantages of OpenID

The potential disadvantages of OpenID fall into a few broad categories. First, OpenID providers consolidate identity information on a large number of individuals. This could make them an appealing target for attackers involved with identity theft and online fraud. This alone may be enough to dissuade some potential users from adopting OpenID.

A second disadvantage is the potential for phishing scams based on OpenID logins. If users are not careful (and even in some cases, if they are) they could be taken from a compromised site to a fake OpenID login page controlled by a phisher. If that were to happen, the user could be tricked into logging into the fake site thus providing the phisher with a user ID and password. The attacker could then pass those credentials on to the real OpenID provider which then performs the actual authentication.

The third potential problem with OpenID is that ID providers will collect personal information, such as names and email addresses, and possibly other information. Unless privacy policies are carefully monitored, users' information may be shared in unacceptable ways.

Summary

OpenID is one of the more successful attempts to provide a SSO service for Web applications. Although a number of major Web-oriented vendors have adopted the standard, there are still security concerns that will likely slow its adoption.

Article 8: Overview of XACML

Developers and application managers have long developed or relied on application-specific authorization mechanisms. For example, a user of an enterprise resource planning (ERP) system might have roles and privileges defined within the ERP. That same user may have additional authorizations for a relational database that is used by the ERP or for a file system on a server used by a related application. Software designers have become adept at developing distributed systems; unfortunately, security often has to be implemented and managed using multiple silos of security management. The advent of the eXtensible Access Control Markup Language (XACML) is changing that.

XACML is a standard developed by the Organization for the Advancement of Structured Information Standards (OASIS) to meet a demanding set of objectives to provide

- A framework for organizing rules and privileges around logical decision points
- A mechanism for supporting multiple subjects who have multiple roles
- A way to base access control decisions on attributes of both a user and a device
- A method to share policies in a distributed environment
- A way to separate policy definition from application implementation

The OASIS committee accomplished this by defining XACML in terms of:

- A high-level architecture
- Policy specifications as well as specifications for requests and decisions
- A process model for interpreting policies and rendering access control decisions

These constituent elements implement authorization decisions using the concepts of subjects, resources, and actions. Subjects perform actions on resources; for example, a database administrator may try to create a file on a database server. Rules governing whether or not such an action is allowed are defined in the XACML languages, which are evaluated using the process model implemented in servers that instantiate the XACML architecture. Together these elements provide the means to overcome the limitations that have long beset the development and management of distributed systems.

XACML High-Level Architecture

In addition to the subjects, resources, and actions just described, the XACML architecture includes four key components:

- PEP – policy enforcement point
- PDP – policy decision point
- PAP – policy access point
- PIP – policy information point

Processing of an authorization request begins with an application making a request to a policy enforcement point (PEP). The PEP creates a formal XACML data structure embodying the request, and sends it to the policy decision point (PDP). The PDP accepts the request and—based on the subject, resource, and action specified—retrieves relevant policies from the policy access point (PAP). The PAP is responsible for storing and retrieving security policies defined in XACML. The PDP may need additional information about a subject or resource before it can make a decision. For example, the PDP may need additional attributes about a subject's position in the organization. This additional information is collected from a policy information point (PIP). The relationship between these four components is shown in Figure 1.

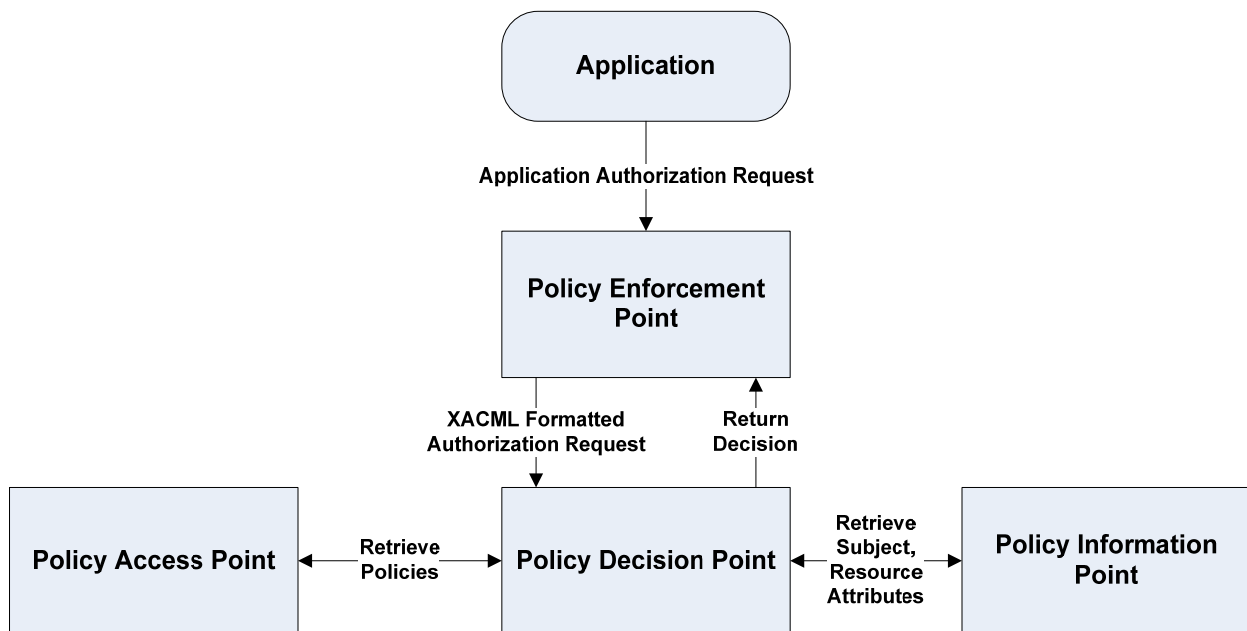


Figure 1: The XACML high-level architecture and decision processing flow.

Information flows from the PEP to the other XACML components and uses XML structures to share information.

Policy Specifications and Decision Rendering

Policies describe what actions subjects are allowed to perform on resources. It is not difficult to imagine how complex policies can become, especially when multiple policies may be applicable at once. A policy is made up of a target, a set of rules, rules that combine algorithms, and obligations.

A target is the subject of a policy, such as a class of servers (a resource), a group of employees (a subject), or an action (for example, changing a database record). A rule is a Boolean expression, such as “subject is a member of the manager group” or “time of day is between 8:00AM and 6:00PM local time,” and the rule result, known as an effect, typically “Permit” or “Deny.”

Rules also have targets that can be subjects, resources, and actions. Rules can be as broad or narrow as requirements dictate. Unlike proprietary authorization solutions, the set of subjects, resources, actions, their attributes, and the granularity of rules are not pre-set. With the flexibility of defining a broad range of rules comes the possibility of conflicts between rule effects.

Rules that combining algorithms are used to resolve such cases and can include:

- Deny Override, in which case a deny decision is returned even if another rule resolves to permit
- Permit Override, in which case a permit decision is returned even if another rule resolves to deny
- First Applicable, in which case the effect of the first rule that applies is returned

Other rule algorithms are possible as well. Obligations are additional actions carried out by the PEP in addition to the authorization decision.

Summary

Many of today’s most complex applications are distributed systems. Without a common authorization framework, systems designers and administrators are faced with coordinating silos of proprietary access control and authorization mechanisms. XACML provides a standards-based, flexible, and fine-grained mechanism for defining and enforcing access controls across distributed systems.

Article 9: Ajax Security Overview: Problems and Solutions

Ajax has become a de facto standard for Web user interface development. Developers using a wide array of tools, from Java Server Pages (JSP) and Java Server Faces (JSF) to Ruby on Rails and PHP, are producing applications with more desktop-application-like features because of Ajax. This is a boon for usability but at the cost of increased security risks. The fundamental problem is not that Ajax introduces new security risks (Ajax is a combination of HTML, XML, and JavaScript) but that it allows us to increase application exposure to existing risks. Such does not have to be the case. If one class of tools can help us introduce vulnerabilities into our applications, another set can help us find them, and yet another can help us keep them out.

This article will briefly describe some of the more common vulnerabilities and risks associated with applications using Ajax. We'll then explore methods for reducing those vulnerabilities and mitigating the associated risks.

Ajax Security Weaknesses

Security professionals have pointed out a number of problems with commonly used Ajax programming techniques, but if there is one that stands out above the rest, it is that we place too much trust in the client. A fundamental premise of Web application development should be to not trust the client.

Developers have essentially no control over what data is sent back from the client or what happens to their code once it is transmitted to a client. For example, a piece of JavaScript sent to the client to perform some operation could be changed by an attacker to alter data sent to the server.

Another problem is that the browser security model does not work well when content is pulled from different sources. A mashup, for example, retrieves data from multiple sites. Often the content pulled from one source is used to dictate what is pulled from another site. Developers can manage this either by using a Web application proxy that combines the results from multiple sites or by directly scripting a page to retrieve data from individual sources. This kind of functionality leaves the application vulnerable to cross-site scripting attacks.

Some of the other well-known security issues with Ajax include:

- Cross site request forgeries (CSRF)—These attacks exploit vulnerabilities in Web sites that trust a user or client. If a user has an open session at one Web site and visits another site, an attacker can use the second site to issue a command to the first site that appears to originate with the trusted user.
- Denial of Service (DoS) attack—JavaScript executes in the client browser, so an attacker could send a command designed to consume resources on the client.
- SQL injection—Unsanitized input to a database query can expose data to an attacker. For example, a poorly crafted WHERE clause designed to retrieve one customer record (for example, “CUSTOMER_ID = cust_id_var”) could retrieve all customer records if an attacker injects an additional clause, such as “ OR 1 = 1”, which would evaluate to true for a customer.
- Client-side injection—These attacks are used to access client-side data or change client-side code. They can take the form of document object model (DOM) injection attacks that manipulate data structures or JSON and XML attacks that change XML structures.

These attacks are enabled by implicitly assuming the client is a trusted platform. It is not. Therefore, commonly used programming models and assumptions have to change when using Ajax.

Tools and Technique for Securing Ajax

There are probably as many ways to improve Ajax security as there are frameworks for developing with Ajax. Rather than try to inventory all the tools available, this article will look at representative examples of tools and techniques, and in particular:

- Smash
- IceFAces
- Vulnerability Scanning

Each of these offers different advantages and should be seen as complementary, not competing, approaches.

Smash—Secure Mashup

Smash is a model for introducing trust domains and communication channels for components. Smash was developed by IBM and released for public use through the OpenAjax forum. Smash uses three abstract elements: components, channels, and an event hub. Different components represent different information sources. Components encapsulate data and code from a single trust domain and use ports to pass data in and out of the component. Channels are communication paths between components. Security policies dictate which components can communicate over channels. An event hub provides a publish and subscribe service that implements channels. Components need to trust only the event hub to enforce its policies on communication; it does not have to trust individual components.

 Smash has been implemented in major browsers using iFrames. More details about Smash are available at <http://websphere.sys-con.com/read/518647.htm>.

IceFaces—Secure Ajax Frameworks

As noted earlier, a fundamental issue in Ajax security is that we tend to fall into a pattern of trusting the client. IceFaces, a component set and framework for Ajax development, makes use of server-side security to avoid the common pitfalls.

The implementation details of IceFaces are beyond the scope of this article, but the basic idea is to leverage the security of Java Server Faces (JSF) and the J2EE security model. IceFaces uses a lightweight bridge component on the client. The bridge is implemented in JavaScript but is limited to submitting data and displaying presentation data incrementally to eliminate vulnerabilities from malicious changes on the client. Changes to the interface are represented in a server-side DOM object and sent, incrementally, to the client via the bridge. This provides the means to keep display and more complex logic on the server side, where it is presumably more secure, and send incremental updates to the client. Data from the client can be verified and sanitized on the server as well.

Smash introduces client-side security mechanisms and IceFaces uses a server-centric approach. Both can still benefit from vulnerability scanning.

Ajax Vulnerability Scanning

The vulnerability scanning market is so large and dynamic it cannot be adequately represented by picking a single product as a typical example. A number of vendors provide vulnerability scanning as a service and others sell scanning products; of course, a number of open source projects also provide vulnerability scanning tools.

Ajax vulnerability scanning should be included with Web application testing in general; this broader topic includes testing for

- Business logic
- Authentication and authorization
- Session management
- Data validation and injection attacks
- Web services testing

Ajax-specific testing should include careful analysis of endpoints, which can be exploited by attackers.

Summary

Ajax is a powerful set of techniques for improving the quality of Web interfaces. There is much concern about the security implications of Ajax (and justifiably so) but there are also ways to deal with these issues. Smash uses a client-side model while IceFaces uses a server-centric approach to mitigating risks. These are just examples of different development approaches that can be used. Of course, vulnerability scanning is called for regardless of other methods employed.

Article 10: Where to Spend Your Security Budget Part 2: Evaluating Security Options

We all want to maximize the benefits of our security resources, but it is not obvious how to do so. There is a wide array of risks that businesses must face; the applications we run and the processes that constitute our operations all harbor vulnerabilities. Security vendors provide an array of products that address many of the security threats in the constantly evolving environment of information security. No single set of product or process recommendations will work for every organization, and even if by some twist of fate they did, the recommendations would have to change soon thereafter—security threats are continuously changing and adapting to the countermeasures we deploy. What we need is a method for choosing the optimal set of security products and processes given the requirements of a particular business at a specific point in time. That is the purpose of this article.

This is the second part of a two-part article on allocating a security budget. The first article (also available as part of the Messaging and Web Security Essential Series, volume 3, in the [Realtime Digital Library](#)) examined three steps to identifying security priorities:

- Gathering risk assessment data
- Identifying and classifying information assets
- Determining the business impact of security threats

That article outlined methods to categorize threats based on the severity, criticality, and likelihood of the threat. Starting where the first article left off, we now turn our attention to examining how different types of countermeasures can be used to mitigate risks to our highest-priority assets.

Limits of Formal Risk Analysis

With so many options, how can IT professionals determine the best combination of resources for their businesses? In an ideal world, we would have enough data about particular kinds of threats and their frequency to calculate how much we should spend on each threat; we would also be able to buy a product or implement a procedure tailored to those threats. These are the assumptions behind a typical textbook risk analysis process. For example, these processes would typically include:

- Determining a precise value for each information asset in an organization
- Identifying all ways this asset could be damaged, lost, or stolen
- Calculating the cost of a single loss, known as the Single Loss Expectancy (SLE) in risk analysis parlance
- Estimating the number of times a year you will experience each threat, known as the annual rate of occurrence (ARO)
- Combining the results of these calculations to calculate the annualized loss expectancy (ALE)

These calculations have well-formed mathematical formulas, but the formalism surrounding these methods can hide their fundamental limitation—the results are only as good as the starting data. Sometimes we do not have the data we need to begin. Consider:

- What is the likelihood of a data loss due to hacking? If a company has never had a known event of that type, is it because there has never been such an event or is the company unaware of past data leaks?
- As one commonly hears in the investment business, past performance is no indication of future performance. Can one assess the chances of a Storm worm-infection based on the success of containing less-sophisticated malware?
- In isolation, a piece of confidential information may be of little to moderate value to a competitor, but combined with other pieces of information, it could allow a competitor to assess a strategy, understand a forthcoming proposal, or gain other insight into a business' operation. How should that initial piece of information be valued in these calculations?

Without well-founded data to begin the process, we must be careful to avoid being lulled into a false sense of confidence by the precision of our calculations. If our risk analysis indicates we could rationally spend as much as \$274,893 to prevent data loss due to hacking, we had better understand the assumptions behind the data input into those calculations.

Another point to consider is that countermeasures do not always map one-to-one with a list of high-priority risks. Consider, for example, content-filtering technologies. Network content filters are useful for scanning content that enters a corporate network to block malware and spam. The same technologies can be applied to content leaving the corporate network to prevent confidential or proprietary information from leaking out.

Formal risk analysis techniques are appropriate to allocating security budgets but they should be used in conjunction with other methods. In addition to quantifying risks, the exercise of making these calculations helps to highlight the difficulties of precisely understanding the scope of security threats. As computer scientist Jan L. A. van de Snepscheut quipped, “In theory there is no difference between theory and practice. But in practice there is.” In theory, risk analysis should be all we need, but in practice, more is required.

An Additional Method for Allocating Resources

Following the steps outlined in Part 1 of this article series, one can define a prioritized list of risks. We will now add to that a list of countermeasure technologies that can mitigate those risks. We will combine these risks and countermeasures into a table (Figure 1 provides an example) to better understand the different ways countermeasures can be combined to address risks.

The purpose of filling out this matrix is to identify a combination of countermeasures that best meets the risk management goals of an organization. Unlike the basic formal risk analysis, this method makes it easy to account for the overlapping uses of a countermeasure. For example, Web content filtering applies to both data loss and malware infection; employee training and monitoring are applicable to all three risk categories.

Risks		Countermeasures					Monitoring
		Encryption	Web Content Filtering	Email Filtering	Client-based anti-malware	Improved employee screening and training	
Data Loss	Stolen Laptop	S				S	
	Transmission to business partners	S					
	Storage at Business Partners						
	Information sent to unmanaged device					S	M
	Employee theft	M				M	M
Malware Infection	Malware/spyware	W	M	M	S		
	via email		S	S	S		M
	via instant messaging		S		S		M
	via compromised Web site		S				M
	via unmanaged mobile device						W
Stolen Computing and Network Resources	Spam/phishing botnets		S		S	M	S
	Compromised servers						S
	Unauthorized P2P networking		S		W	M	S

Figure 1: A risk x countermeasure matrix rating the relative effectiveness of each countermeasure for mitigating each risk.

If a countermeasure is useful in mitigating the risk, it is assigned a rating of strong (S), moderate (M), or weak (W) according to how well it reduces the risk. These ratings are a function of particular requirements and may vary from one business to another. In the hypothetical scenario represented in Figure 1, for example, a company is considering folder-based encryption. This countermeasure offers some protection against file stealing malware but only for some files in encrypted folders, so encryption is ranked as a weak countermeasure for data loss due to malware.

The matrix can be used in a number ways. First, the ranks can be assigned weights (for example, S=3, M=2, W=1), which are summed to calculate a relative value of a countermeasure. In Figure 1, encryption would be valued at 9 while Web content filtering would be valued at 17 (see Figure 2). This measure can give a global ranking to a countermeasure based on security requirements. Combining this global ranking with the cost of each solution can help highlight the relative value of the technology to the business. Also, the matrix can be used to identify subsets of countermeasures that meet certain criteria, such as all risk categories must have at least one strong countermeasure. This can help identify combinations of countermeasures that meet overall requirements.

		Countermeasures					
		Encryption	Web Content Filtering	Email Filtering	Client-based anti-malware	Improved employee screening and training	Monitoring
Data Loss	Stolen Laptop	3				3	
	Transmission to business partners	3					
	Storage at Business Partners						
	Information sent to unmanaged device					3	2
	Employee theft	2				2	2
	Malware/spyware	1	2	2	3		
Malware Infection	via email		3	3	3		2
	via instant messaging		3		3		2
	via compromised Web site		3				2
	via unmanaged mobile device						1
Stolen Computing and Network Resources	Spam/phishing botnets		3		3	2	3
	Compromised servers						3
	Unauthorized P2P networking		3		1	2	3
		9	17	5	13	12	20

Figure 2: A risk x countermeasure matrix with numeric weights.

The matrix does not produce the “correct answer” to the question of which technologies to purchase, but it does organize key information so that combinations of countermeasures can be compared. What it lacks in the precision of formal risk analysis, it makes up for by capturing general information about the technical and procedural options available to practitioners.

Summary

Evaluating security tools, products, and processes is difficult because there are so many dimensions along which one can analyze the problem. The first part of this article series outlined a series of steps to gather risk data, classify information assets, and assign priorities to risks. This article highlights the limitations of formal risk management calculations—that is, the difficulty in accurately estimating input data. A supplementary method can be used to better understand how combinations of countermeasures may be used to meet the security needs of an organization.

Article 11: Techniques for Detecting Stealth Malware

Malware developers go to great lengths to keep us from finding their malicious code. Malware can execute without showing up in process lists. Registries do not indicate the presence of the unwanted software. Directory functions do not list unexplained disk space in use. The fundamental problem stems from trying to use a compromised system to detect the presence of a compromised system. Fortunately, advances in virtualization have led to techniques for observing the behavior of operating systems (OSs) and local hardware that can aid in the detection of stealth malware.

In this article we will look at

- Techniques for hiding malware from detection
- Methods for detecting the presence of malware in spite of stealth techniques
- Potential limitations of these detection techniques

Hiding Malware

To hide its presence, stealth malware uses a number of techniques that manipulate OS data and functions. These techniques, generally called *hooking*, come in a variety of forms. There is, for example, Import Address Table (IAT) hooking. An IAT contains lists of pointers that are used to reference functions needed by an executable but provided by a dynamic link library (DLL). By tampering with these pointers, attackers can change the code that is called by a program; rather than run the expected DLL, the compromised device runs attacker-injected code.

Another way to tamper with OS code is to change system services code. For example, a program might call a system service to create a process, which, in turn, calls another lower-level function. If the pointer to that function is changed, an alternative function can be run to create a process with different characteristics.

In some cases, rootkits and stealth malware may manipulate kernel objects, which under normal circumstances are inaccessible to user-mode programs. Device drivers can operate in kernel mode and thus have access to kernel mode data structures, such as those containing process and networking information. Malware that has access to elevated privileges, such as in an account running with administrator privileges, can operate in kernel mode and inject malicious code, which in turn tampers with kernel objects.

By manipulating the functions that OS and other applications call and by changing data in kernel mode objects, malware can manipulate some of the lowest levels of the OS and mask the presence of unwanted programs. Crucial to evading this type of attack is to avoid compromised OS code and data.

Methods for Detecting the Presence of Stealth Malware

Beyond basic OS functions, several techniques have been developed for detecting stealth malware. One approach, known as a cross-view diff-based approach, detects hidden objects by comparing the results of two scans. One scan is made using higher-level OS functions that are typically compromised by rootkits and other stealth malware. A second scan is made with low-level functions that are below the point of tampering. The results of the two scans are compared and differences in files, registry entries, and processes can indicate a stealth malware infection.

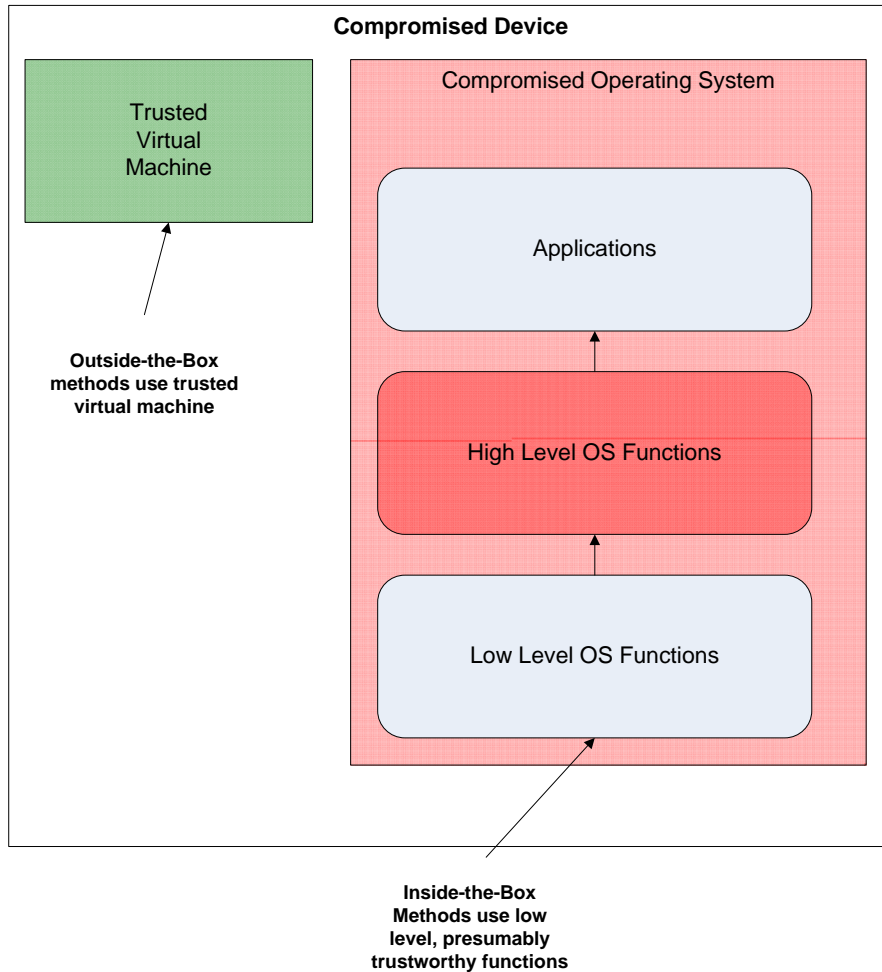


Figure 1: Stealth malware detection can occur either inside the compromised OS or outside in a trusted virtual machine.

An advantage of this approach is that one can use the possibly infected device to detect the malware. This technique is known as an inside-the-box method. A disadvantage of inside-the-box detection methods is that they depend upon possibly compromised code; this shortcoming is avoided with an outside-the-box approach.

An outside-the-box approach can take advantage of virtual machines to isolate a trusted version of the OS running under a hypervisor to monitor activity on another virtual machine running on the same physical device. Virtual machines are logically separated, so the trusted virtual machine can detect on another virtual machine only very low-level operations, such as disk operations, instead of higher-level OS operations, such as file manipulations. There have been advances in this area to overcome the gap between low-level detection and higher-level interpretation. For example, work at George Mason University and Perdue University [Jiang, Wang and Xu, 2007] demonstrates techniques for reconstructing a higher-level view of data obtained by virtual machine monitoring.

Potential Limits of These Techniques

Both inside-the-box and outside-the-box techniques are promising; however, given the history of malware detection, it is reasonable to assume that malware developers will actively seek ways to avoid these detection methods. Inside-the-box methods that may be circumvented in lower-level functions are corrupted by new stealth techniques. Outside-the-box techniques depend upon inferring information from low-level data; perhaps malware developers will find a way to sufficiently change the behavior of their code to mask their existence. The next innovative countermeasures developed by malware developers remain to be seen; the only thing we can be sure of is that someone, somewhere is going to try.

References

Xuxian Jiang, Xinyuan Wang, Dongyan Wu “Stealth Malware Detection Through VMM-based ‘Out-of-the-Box’ Semantic View Reconstruction” (2007, available at <http://ise.gmu.edu/~xwangc/Publications/CCS07-VMwatcher.pdf>).

Article 12: Localized Malware

Malware continues to evolve and not just in terms of technical sophistication. Recent research on malware trends has found significant effort on the part of malware developers and cybercriminals to adapt their malicious applications according to regional or cultural conditions. This article will examine several issues related to this trend:

- Motivations for localizing malware
- Examples of localized malware
- Suggestions for controlling the impact of localized malware

Although localized malware, by definition, is highly varied, we have sufficient examples to make reasoned conjectures about further enhancements and specializations that emerge as part of this trend.

Motivations for Localizing Malware

Malware developers and cybercriminals are not posting their motivations online, but the single most important driving factor behind localized malware is money. Anti-malware researchers have found examples of localized attacks on online banking customers in one country and attacks on online gamers in another. Financial gain is the common theme between both instances. There are always exceptions; for example, there are instances where local fame is the apparent motivation, but for the most part, these specialized attacks are financially motivated.

These overarching motivations have existed for some time, certainly long before the emergence of localized malware. This leads to the question, why bother to go to all the effort to localize malicious programs? There are a number of possible reasons.

Avoiding Detection

The first objective of a malware or phishing attack is to reach the target. Content filters using statistical pattern recognition are highly accurate when it comes to detecting spam, especially when the text within the spam message follows commonly used patterns. A few words about pharmaceuticals or wire transfers to exiled Nigerian officials are enough to trigger even the most basic email filter.

Spammers have tried to trip up the statistical filters by adding nonsense text, but even those techniques have common attributes that can be detected. For example, scrambled text is placed at the end of an email message so as not to disrupt the flow of the message; this pattern can be exploited to identify unwanted messages. Localizing spam, phishing lures, and Web sites can help improve the chance of avoiding detection by avoiding patterns found in mass distribution spam.

Improving Click-Through Rates

Once a piece of malware has made it past network and local client antivirus, anti-spam, and other content filtering, there may still be a need for some kind of user interaction:

- Clicking a link in a message to browse to a compromised Web site where a drive-by download will infect the client device with malware
- Opening a file and in the process launch a Trojan that installs a keylogger
- Clicking a link to a phishing site posing as a legitimate business site to get users' login credentials

By now, many users are wary of clicking on messages that appear remotely suspicious. One of the ways spammers, phishers, and malware pushers reduce suspicion is to specialize the content of a message. This can be done by using details about the target (for example, consider the recent round of phishing scams that targeted business executives with fake court documents about their businesses). Localized malware exploits the same basic principles to make an attack appear so targeted to a particular person or region that it does not generate the same level of suspicion as a more generalized attack.

Staying Under the Radar

Another advantage of localizing malware and spam is that they are less likely to reach levels that all but ensure a widespread attack will be detected. Once again, this trick has been used before. Spear phishing, in which a narrow, targeted audience is sent phishing lures with precisely tailored messages, is an example of this kind of “stay under the radar” strategy.

Clearly, there are distinct advantages to localizing malware even if it means substantially reducing the size of the potential target base.

Examples of Localized Malware

In a recent study on localized malware published in McAfee Avert Lab's [Sage Report](#), a number of examples are described. These examples give a sense of the breadth of localization in the malware cyber-economy:

- European Union (EU) citizens, in spite of their 23 languages, are targets of attacks in local languages
- Transnational sports events, such as World Cup soccer, can be exploited with appeals to local fans
- In places where online gaming is increasing popular, such as China, malware developers are looking to steal virtual goods that can then be sold for real cash
- Brazilians have been the unfortunate target of online banking scams that use elaborate social engineering to lure victims in a country where many banking customers routinely use online services

Similar types of specialized attacks occur during holidays or around well-publicized events, such as Hurricane Katrina in the United States in 2005.

Controlling the Impact of Localized Malware

Localized malware is growing precisely because it is a way to avoid both technical and human measures to control it. In the long run, technical solutions such as localized content filtering and localized anti-spam measures will reach the market and provide more defense against these emerging threats. In the meantime, user education is the best means to blunt the effectiveness of localized malware. If users are aware that new, specialized scams are used to push malware and steal personally identifying information (PII), they will be more likely to hesitate before clicking that link or opening that attachment.