

Role Comparison Methodology

Measuring Vendors on the Customer-Centric Security of Software

Richard Ford, Ph.D.

Herbert H. Thompson, Ph.D.

Fabien Casteran, M.Sc.

February, 2005



1990 W. New Haven Ave., Melbourne, FL 32904

Tel: (321) 308-0557 Fax (321) 308-0552

info@securityinnovation.com www.securityinnovation.com

Executive Summary

When considering which platform to deploy across an enterprise or to serve a particular role, IT decision makers have always looked at a small set of criteria, such as purchase price, compatibility with existing applications/technologies, maintainability and deployment cost.

Historically, security has been conspicuously absent from that list. In many cases though, the cost to enterprises of poor security acquisition and deployment decisions has eclipsed other traditionally evaluated costs. With this in mind, the industry urgently needs objective measures of platform security that are meaningful in a deployed context. Any meaningful security measure *must* be based on a holistic view of a system and must also consider what role that system will serve.

We set out to define a methodology that would identify and analyze objective security metrics to enable meaningful comparisons of software vendors and the products they deliver to customers. With that in mind, we wanted to keep a tight focus and produce a work that could serve not only as a basis to help customers today, but also as a critical building block for future research by ourselves or others in this area.

When we presented some high-level, preliminary results at the 2005 RSA Conference, we received many requests for us to publish the methodology itself so it could be examined, scrutinized and thoughtfully commented on; this paper is a direct result of those requests. We think that any research can benefit from such of scrutiny and that input from academics, industry leaders, analysts, and users will help improve future research and studies.

Scope of Analysis

Surprisingly, with a few notable exceptions – in particular, the Forrester report by Laura Koetzle¹ – there has been very little published research on objective metrics that help customers evaluate software vendors on security. With that in mind, we wanted to produce a work can not only help customers today, but also serve as a scientific foundation for future research by ourselves or others in this area.

To get a full view of Security Risk, one has to get a view of two factors:

- Vulnerability of software, systems or networks (whichever is appropriate), and
- Threats against those vulnerabilities

Of the two factors, our own experience leads us to believe that the latter is more difficult to quantify and predict in an objective manner. This is an exciting and open field and we strongly encourage others to consider this as an area for thoughtful research. However,

¹ “Is Linux More Secure than Windows” by Laura Koetzle, Forrester Research, covers some of the issues outlined in this paper well.

given that there are research opportunities in both areas, we have chosen to try and make progress in studying and measuring the vulnerability factors first; this is a critical precursor to other threat-based metrics.

Our Research Goals

For purposes of our ongoing research, we want to look at customer-focused comparative measures of security for platforms that the vendors have the ability to affect and improve. With that focus in mind, we look only at the underlying vulnerability of software and not delve into the threat environment. The idea behind this approach is two fold; first, our aim is to examine those aspects that are under the control of the vendor or developer. As such, attack rates and other external factors are not included. Second, this metric directly relates to the level of “pain” experienced by customers attempting to maintain a secure system, and can be used to help steer vendors via positive customer pressure. We believe these issues are central to creating a study that is practical and useful to customers.

To understand the utility of this approach, take the example of a corporation that has determined that it has confidential and valuable information that it must protect from potential attackers. The corporation may handle credit card information as an online retailer or bank, for example, or be the target of political activism, such as the several Republican web sites which were attacked during the recent presidential election². Since the assumption by this type of corporation is that there are motivated attackers that could target their systems, this essentially equalizes the threat factors, regardless of selected software platform, elevating the vulnerability of the software and the behavior of the vendors as the primary factors in the risk equation.

Future Areas of Research

Defining a methodology for measuring the threat environment will also be an excellent area for future research that would build and expand on the foundation of this work and provide a richer set of criteria for customers.

We can easily imagine a taxonomy that articulates the different kind of threats and would explore metrics to define a different measurable “threat factor” for each threat. This model could then be combined with software vulnerability metrics to provide a richer set of decision criteria than the simpler base model we investigate here.

²*Wired*, “Hackers Take Aim at GOP,” <http://www.wired.com/news/politics/0,1283,64602,00.html>

Introduction

Security is a serious concern for those deploying modern computing systems – so much so that the relative security of different solutions can be a major factor in choosing platforms and applications. Additionally, given product deployment lifecycles, many companies are making choices that will affect their operations for as much as the next ten to fifteen years.

In developing our methodology, we examined existing research and found very little work in the area of repeatable, objective methodologies for security measurement. The one work we did find was the 2004 Forrester study around days-of-risk³. While it was a solid foundation of work, we identified some open issues:

- Number of Products question. Forrester measured aspects of all vulnerabilities from vendors and not specific to products. In theory, if one vendor had 600 products that it was maintaining and another had 2, the deck would be significantly stacked against the company with more products.
- “Apples to apples” question. A Windows Server OS and a Linux Server OS are vastly different in terms of components that may be installed during a given deployment. What is the impact on the final results if one looks just at the components necessary to support a given role, such as the role of Web Server or File & Print Server?

Additionally, there is the Severity question, one that the Forrester research did factor for the vulnerabilities they studied. Customers treat vulnerabilities differently based on their perceived severity. Do vendors allocate *their* resources (time to patch, etc.) based on severity? How do vendors compare on high, medium and low severity issues in terms of responsiveness and customer exposure in the context of specific versions used in specific roles?

Given these questions, our methodology focuses on software solutions as they would likely be used by real customers to fulfill specific roles. By concentrating on *roles*, it is possible to create server comparisons that are meaningful, and firmly centered on customer requirements. We believe that this provides for meaningful comparisons of equivalent functionality.

Platforms Compared

In our analysis we wanted to choose two platforms to compare what would be most meaningful to customers. Given the high level of interest in Windows versus Linux, we chose the most recent version of Windows server software, Microsoft Windows Server 2003 and Red Hat Enterprise Linux 3⁴. We note that there are many different distributions of Linux that we could have selected, but recent analyst reports indicate that business customers are largely selecting to deploy either Red Hat or SuSE in production environments. Thus, we selected Red Hat for these tests as it is the current leading

³ “Is Linux More Secure than Windows” by Laura Koetzle, Forrester Research

⁴ Although we look at the ES version of Red Hat Enterprise Linux 3, the packages installed in the AS version are very similar and thus we expect the results with that platform to be comparable.

distribution⁵. Furthermore, due to its strong position in the enterprise solutions market, Red Hat is arguably the best representative of the open source distributors and the preferred candidate for our analysis.

The methodology is generic, though, and could be applied to other products and vendors.

Requirements

Comparing the security of an operating system based on an analysis that considers all security issues reported on all applications that run on this operating system is neither realistic nor helpful to decision makers. In reality, very few users have all of the vendor applications installed or running on their systems. Thus, the focus of our methodology is to compare the security of systems configured in particular server roles, as this is much more representative of “real world” scenarios. As such, the systems we will consider *only* function under a specific role.

We believe there are two large classes of deployments. One group is those who deploy a server largely using the default vendor settings and selections as prompted by the installation procedure. The second group is those who deploy a server utilizing modularity to uninstall any unnecessary components from the server role. From a security perspective, if it is reasonable and possible to uninstall a component so that the code of a non-role component is not present this action will reduce the attack surface of that system. One of the security strengths frequently cited with respect to Linux is that its modularity allows a true “minimal build” of a server role, thereby reducing its effective attack surface and making it more secure. In our studies, we will apply this methodology to examine both the default and a minimal configuration to better understand the security implications for the two types of deployments.

Microsoft, on the other hand, has pursued a strategy that they have called “integrated innovation” that makes it difficult to easily uninstall many components. For example, on Microsoft Windows Server 2003, we will consider components like Internet Explorer to be present on the workloads since they cannot be easily uninstalled from that role. In our research, we will treat the Microsoft default and minimal deployments as being essentially the same and assume that a customer would need to patch *any* issue that is present in Microsoft’s server software. Thus in the “minimal” configuration where we would not count browser (Mozilla) issues for Linux server roles for example, we will count Internet Explorer issues on Windows.

Additionally, we will consider all of the services and applications that must be running on each system to satisfy its role, and in doing so, will provide a real-world security comparison that is meaningful for the role being studied. When performing actual studies, we will deploy the software and physically validate configurations.

The focus of this analysis is to capture the user concerns when choosing a server in a particular role. User needs may vary greatly and therefore system requirements can also

⁵ Source: IDC report “Worldwide Linux Operating Environments 2004-2008 Forecast and Analysis: Enterprise Products Pave the Way to the Future”, December 2004.

vary greatly. Thus, our focus is to find system requirements that cover the majority of user needs and concerns in the most likely scenarios.

To assess security we will consider both quantitative and qualitative metrics. Our analysis will then validate the two solutions on both fronts.

Assumptions & Rules

As indicated previously, we analyze two basic cases for each server, one consisting largely of the default configuration as installed by the vendor's installation software and one that is a more minimum installation of the server role, again using tools and guidance provided by the vendor.

Linux Default Configuration For this case we will use the "default" configuration of a machine wherever possible. When features had to be enabled to provide for required functionality, documentation and/or default processes will be followed wherever possible. Our choice to consider the default case is based upon our own observations of empirical evidence that default configurations are frequently encountered in the "real world".

Linux Minimal Install We also consider the case of the minimal install where the server is specifically tailored to serving the role by disabling default but unnecessary components and services during the install process. The minimal install scenario provides for a reduced attack surface for the server and makes the assumption that the server is deployed by an administrator that prioritizes security above other criteria.

Microsoft Default and Minimum Configuration As Microsoft's design does not make it easy to remove some additional components, the default and minimal configurations will both be considered to include all components that ship with Windows Server 2003.

Analysis NOTE: Although additional steps could be carried out to “harden” servers from attack, we will not consider them in our methodology. We will compare default and minimum installations of Red Hat with a Windows Server assuming all components installed. In preliminary reviews, we received questions about this approach and the level of hardening we did or did not do.

As security practitioners, we know either system is highly “securable” by an expert with the right skill set. For example, we could lock down Internet Explorer or assume that a company had a policy of “no browsing” from servers. Similarly, many packages could be recompiled or removed from Linux in order to reduce the attack surface. However, at the end of the day, each vendor follows its own philosophy and makes decisions that impact the vulnerability and ease of security management of systems for any customers that are less savvy than top security practitioners. Experience in the security space has shown time and time again that the default configuration is often the configuration used in the real world – for better or for worse.

One of the key criticisms levied against previous security comparisons is that they do not give fair credit to the Linux ability to create and deploy a minimum set of components – a security advantage it has over Windows. In comparing the fully-loaded Windows configuration against smaller Linux configurations, we plan on leveraging the modularity of Linux to create a minimal configuration for a given role.

Beyond installation, the following is a list of user concerns and needs that we assume for this analysis:

- The user requires the features, trust, support and professional maintenance provided by a trusted software distributor. For example, in the case of an open-source solution like Red Hat Enterprise Linux, we assume that, in the interest of manageability and its associated cost, users will only install versions of the OS components blessed and released by the OS vendor, so that their support contract remains valid.
- The web server is expected to serve pages with dynamic content in a fast, reliable and secure way.
- The different components of the web server should be compatible and easily integrated.
- The web servers must have comparable scalability and performance.
- The database management system must be able to execute simple queries in a fast and reliable way. We assume that the backend database is relatively simple; that is, that the stored data does not need extensive or complex computations for the web site purposes.

When considering the relative security of different solutions, it is important not just to consider *what* is installed, but also *how* it is installed. Thus, the default security features – essentially, the *context* in which a role is deployed – is important when considering the long term security viability of a solution.

Contextual information that is important with respect to security includes:

- Open ports in the default/configured deployment
- Users (and their privileges)
- Other applications that may modify behavior with respect to vulnerabilities
- Technology that mitigates the vulnerability of a system (e.g. buffer overrun protection)
- Environmental considerations (such as “a server satisfying this role is usually firewalled”)
- General attack surface considerations such as privilege level of exposed services

Additional assumptions on Quantitative Data

Most of the quantitative data available is related to vulnerabilities, patches and patch quality. This information can be obtained from public bug reporting lists and from the vendors issuing the patches. While these results represent only the vulnerability dimension of security risk, they do provide insight into the aspects of security quality that under the control of the vendor - code security quality and security response. These metrics however must be considered in combination with several other important qualitative factors when choosing a platform based upon cost of security maintenance and likelihood of security breach.

In order to make an unbiased comparison between two platforms the set of assumptions used in gathering the data is crucial. This information is also essential to making the experiment reproducible. The following is the set of assumptions that was used to gather the vulnerability information and patch information.

- i. It is assumed that Red Hat customers only install patches released by Red Hat and are taking other similar steps to ensure they comply with their maintenance contract. Similarly, Windows customers only utilize fixes released by Microsoft.
- ii. All the fixes released by Red Hat and Microsoft pertaining to the two operating systems will be recorded along with the applications to which the patches pertain. For each fix, the vulnerabilities addressed will be entered using the Mitre vulnerability identifier⁶ (e.g. CVE-2004-0079).
- iii. When an application is installed by default, all updated versions will be considered to be default applications in as well.

⁶ The Common Vulnerabilities and Exposures (CVE) database is a widely-accepted standard for identifying specific vulnerabilities. CVE is available online at <http://www.cve.mitre.org>. Also see section “Mitre CVE List” below.

- iv. For purposes of the time period we study, we assume systems are fully patched up to the date of the study. For example, in looking at 2004, we assume all patches from 2003 had already been deployed on both systems.
- v. The ‘first public’ date for a vulnerability is the date at which the vulnerability was first released on a public list or web site (Bugtraq, Red Hat, Microsoft, Full-disclosure, k-otik) devoted to security, or a publicly accessible list of bugs or problems posted to the home site of package or its mailing list. We do not consider discussions on the Linux ‘vendor-sec’ alias as public.
- vi. Dates for patches are based on the release date for the distribution of interest.
- vii. Release dates for a vulnerability patch or fix are specific to a distribution/architecture. If a fix for a component (e.g. libpng) is released on 01/01/1970 for a certain Linux distribution (e.g. Gentoo Linux) and a fix for the same issue is released for Red Hat on 01/10/1970, the release date for the fix on Red Hat will be 01/10/1970. This is not applicable for the Windows platform.
- viii. For past issues, the release date for a patch is the first published vendor report that includes the patch for the applicable platform for which the patch fully fixed the vulnerability. If the patch had to be re-issued to address some portion of the security issue, the latter date is used.
- ix. Documentation packages will not be entered. Applicable only for the Red Hat platform.
- x. 80x86 is the only architecture considered for the Red Hat platform. Not applicable for the Windows platform.
- xi. For Windows applications, the patch ID is the bulletin number associated with a specific vulnerability.
- xii. It is assumed that patches are installed in the order of release.
- xiii. Functional bugs are only entered as vulnerabilities in the event that patches introduce them. In this case, the vulnerability will not be associated with an application. An additional “functional” flag has been added to the vulnerability table to make querying easier.

In order to compile the data in a single, centralized location we created a database that was populated using different sources and consolidated using the Common Vulnerabilities and Exposures identification numbers. By creating such a database, we have the ability to easily query exploits, vulnerabilities and patches based upon a number of different criteria, such as vulnerability implication, role or days of risk.

Mitre CVE List

In our analysis we frequently refer to the CVE or CAN identifier of a vulnerability. CVE stands for Common Vulnerabilities and Exposures and is a taxonomy that attempts to standardize the naming of all publicly known vulnerabilities and exposures. Initially, vulnerabilities are assigned a candidate number (CAN). These candidates are then examined by CVE’s Editorial Board – made up of industry experts – where a decision is made for their inclusion in CVE. CVE is maintained by the Mitre Corporation, a not-for-

profit organization that performs independent research and analysis for the U.S. Government. In our analysis, we refer to a vulnerability as distinct if it has its own CVE or CAN identifier. It is possible that vulnerabilities are announced that have not yet been assigned a candidate number. While this is rare, these vulnerabilities would also be considered in our analysis.

NIST ICAT Severity Ratings

One of the most hotly debated topics in security is the severity of impact a particular vulnerability can have. The National Institute of Standards has introduced the ICAT Metabase which contains information about known vulnerabilities and uses CVE identifiers to catalog its entries. One interesting aspect of ICAT is the severity rating it assigns to vulnerabilities. ICAT ratings are a generally accepted and objective way for system administrators and IT professionals to gauge the impact of a vulnerability on a typical system. Although ICAT severity ratings do not offer contextual guidance for how severe a certain vulnerability is in a certain context, they do provide an objective way to classify vulnerabilities. In this capacity, we use ICAT severity ratings in our analysis to group vulnerabilities into classes. ICAT uses three broad severity classes for vulnerabilities: High, Medium and Low as defined below⁷:

A vulnerability is “high severity” if:

it allows a remote attacker to violate the security protection of a system (i.e. gain some sort of user or root account),

it allows a local attack that gains complete control of a system,

it is important enough to have an associated CERT/CC advisory.

A vulnerability is “medium severity” if:

it does not meet the definition of either “high” or “low” severity.

A vulnerability is “low severity” if:

the vulnerability does not typically yield valuable information or control over a system but instead gives the attacker knowledge that may help the attacker find and exploit other vulnerabilities.

we feel that the vulnerability is inconsequential for most organizations.

Unrated vulnerabilities – While ICAT contains severity ratings for the majority of the vulnerabilities in CVE, there are several vulnerabilities that remain unrated. Ratings are continuously updated on the site and this study’s rating information is current as of January 27th, 2005. When evaluating the statistics presented later in this report referring to severity we encourage you to treat vulnerabilities with a rating of “Not Known” with extreme caution as they have the potential to be high severity.

⁷ Information is taken from the official ICAT documentation found at http://icat.nist.gov/icat_documentation.htm

Analysis NOTE: One interesting question we received when getting our preliminary results reviewed was “why didn’t you use the CERT severity metric?” The answer is that the CERT severity metric incorporates multiple factors in ways that are subjective and that can change frequently. From CERT’s web site:

The metric value is a number between 0 and 180 that assigns an approximate severity to the vulnerability. This number considers several factors, including

- *Is information about the vulnerability widely available or known?*
- *Is the vulnerability being exploited in the incidents reported to US-CERT?*
- *Is the Internet Infrastructure at risk because of this vulnerability?*
- *How many systems on the Internet are at risk from this vulnerability?*
- *What is the impact of exploiting the vulnerability?*
- *How easy is it to exploit the vulnerability?*
- *What are the preconditions required to exploit the vulnerability?*

Because the questions are answered with approximate values that may differ significantly from one site to another, users should not rely too heavily on the metric for prioritizing vulnerabilities. However, it may be useful for separating the very serious vulnerabilities from the large number of less severe vulnerabilities described in the database. Typically, vulnerabilities with a metric greater than 40 have been candidates for a CERT advisory, and we will continue to use this metric for US-CERT Technical Alerts. The questions are not all weighted equally, and the resulting score is not linear (a vulnerability with a metric of 40 is not twice as severe as one with a metric of 20).

CERT severity, based upon this definition, would vary greatly over time. The rating also takes into account how many systems might be deployed – this would mean vulnerabilities in software being used by only a few people would always have a lower severity rating by definition. Also, some of the factors are very subjective. For example, what does it mean for a vulnerability to be “easy” to exploit? Does it become “easy” after a sample exploit is published? Are the severity ratings on the web site currently accurate or were they only accurate on the day CERT published them? We can’t easily tell.

So, for our purposes, ICAT, which provides a rating for each CVE Name seems to be a more objective and transparent rating.

A word of caution on ratings – When examining vulnerabilities, there is a tendency to ignore or devalue those rated as “Low”. We have seen several attacks however that have exploited several “Low” rated vulnerabilities to gain complete remote control over a machine. The synergistic combination of vulnerabilities that are rated low and medium in

isolation can lead to an exposure of the highest severity. Another issue is the contextual severity of a vulnerability. Rating systems like ICAT assign severity labels to vulnerabilities based on their potential impact to a system or application in isolation. These ratings offer minimal insight into the impact of a vulnerability on a deployed solution. For example, protections like host-based firewalls may mean that a particular system is not vulnerable to certain exposures that are rated as “high”. While contextual ratings can be useful for patch deployment prioritization in an organization, it is important to remember that configurations are constantly in flux and contextual protection is only temporary. Vulnerabilities must be fixed at their root cause to truly limit exposure in the long run.

Analysis of a Server Role

Quantitative Metrics Description & Introduction

Historically, platform security comparisons have been made on quantitative and readily available data. Such comparisons have frequently made a simple count of “security bulletins” or “security advisories” issued by vendors.

While advisory counts are popular, there is little evidence to support their usefulness in making strategic security-aware deployment decisions, since vendors control how many vulnerabilities might be addressed by a single security advisory. These simple bulletin counts may not represent the underlying security quality of the products very well, unless extra care is made to assure vendor behavior is similar. For example, SuSE Enterprise Linux and Red Hat Enterprise Linux have a high degree of correlation between components. However, though both fix a similar set of core (e.g. Linux kernel, X- Windows, hardware drivers, rsync) component vulnerabilities, the number of SuSE security advisories is significantly lower. If one were to carry out a simple analysis of advisory count, it would paint a better – but ultimately misleading – picture of the relative security of SuSE with respect to Red Hat. In reality, the number of vulnerabilities fixed by each is of the same order of magnitude, but Red Hat is more granular (and one might argue transparent) in its release of security advisories.

Another problem with such raw bulletin/advisory counts is that they do not take into account the context in which the platform is used. A system in a web server role, for instance, is likely to have a very different configuration and attack surface than a system deployed in a file server role. Provided that these two systems were running the same OS, however, they would likely look identical from a raw vulnerability or advisory count.

Such approaches also fail to take additional contextual information into account such as what environments a particular server filling a given role is likely to operate in (NATted, Firewalled, etc.) In short, these base metrics provide little information to someone that must make informed acquisition and deployment decisions.

Instead, we take a role-based approach to measuring security based on likely deployed configurations. For quantitative data, this approach means only considering those vulnerabilities and patches that apply to the deployed role. For instance, we shall not consider a vulnerability reported in a component that is not installed in our functioning web server solution.

Time Period

The methodology could be applied for any fixed time period for comparisons. For our studies, we will use vulnerabilities disclosed earlier than 2004 if and only if the solution vendor (Microsoft or Red Hat) have released a fix for these issues in 2004. Similarly, we will not consider vulnerabilities *announced* in 2004, but fixed in 2005. One could select different time periods in the future and repeat the studies using the new time periods and begin to study trends as well.

For each of the configurations we would like to study, we identified a set of quantitative metrics that we could extract and calculate from the data:

- Total # of vulnerabilities in the configuration (count)
- Total # of High Severity vulnerabilities in the configuration (count, by severity)
- Total # of Unknown Severity vulnerabilities in the configuration (count, by severity)
- Total # of Medium Severity vulnerabilities in the configuration (count, by severity)
- Total # of Low Severity vulnerabilities in the configuration (count, by severity)
- Average Days of Risk for all vulnerabilities in the configuration (average)
- Average Days of Risk for High Severity vulnerabilities in the configuration (average, by severity)
- Average Days of Risk for Unknown Severity vulnerabilities in the configuration (average, by severity)
- Average Days of Risk for Medium Severity vulnerabilities in the configuration (average, by severity)
- Average Days of Risk for Low Severity vulnerabilities in the configuration (average, by severity)
- Cumulative Days of Risk for all vulnerabilities in the configuration (total)

Days of Risk Discussion

Cumulative & Average

In our metrics, we refer to cumulative and average days of risk. Each are important to consider because each offer some insight into the user-centric security of the solution and also the security service of the vendor. Cumulative days of risk speak to the overall exposure of a solution as well as the security service of the vendor in one statistic.

Average days of risk are a valuable measure of vendor security service as they speak to the average time between when a vulnerability is disclosed to public and when a vendor fix is available, representing the time period that customers are exposed to higher levels of risk, but do not have a patch.

Analysis NOTE: One of the key things we learned during reviews of our preliminary results is that there are strong opinions on the value, or lack of value, for days-of-risk metrics. For us, we see a pretty clear distinction in customer risk for before and after an event becomes widely known to the public. Thus, days of risk is an interesting measure to see how the combination of a vendor's disclosure policies, response process, and patch test and release process combine to shrink (or not shrink) the time period when customers are exposed without a patch alternative from the vendor.

One point that was raised was that this type of metric automatically skews in favor of a closed source solution. Another point questions Microsoft advocacy of "responsible disclosure". In fact, we acknowledge that a vendor with a longer quality and testing process might benefit more from responsible disclosure, but since responsible disclosure leads to less risk for customers, it seems to emphasize and not detract from the importance of the metric.

Actually, the data is pretty clear that both Microsoft and Linux community follow responsible disclosure to some degree. For example, for the Samba issue fixed by Red Hat in RHSA-2004:670-10, one can follow the references to see that the vendors kept the issue from being made public on the date when the vendors were first notified. Here is a partial bugzilla entry:

Jerry at Samba reported to vendor-sec on 20041209 a remote root flaw in Samba, affecting all versions. Requires authenticated user. Issue was discovered by iDEFENSE.

This issue is currently embargoed until 20041216:1200UTC

One might argue that the issue was 'public' when first reported to vendor-sec, due to the number of people on the list. However, we used 12/16/2004 as the first public date the issue was widely known. Red Hat had 15 issues fixed with zero days and most of those benefited from responsible disclosure privately to the Linux vendors – actions we applaud.

Further, it seems clear that customer risk could be reduced even further by more security response coordination by Linux vendors. Consider for example RHSA-2004:413-07, which patched a kernel issue and made the issue public on 8/3/2004. Suse users didn't have a patch until 8/9/2004 and Gentoo Linux users waited until 8/25/2004, according to the references in the CVE list.

Qualitative Metrics Description & Introduction

Beyond patches and vulnerabilities, there are “softer” qualities of security that are difficult to quantify but undeniably impact deployed security. Qualities like security lifecycle support, bulletin descriptiveness, default security features and the like all have a direct impact on deployed role security. In our analysis, we will study how vendor policies and practices in these areas may impact customer security. Here are the areas which we will be studying:

- Default security features in a role
- Port protection capabilities
- Lifecycle support policies
- Security advisory descriptiveness
- Patch impact disclosure
- Patch rollback capabilities
- Patch deployment Technology
- Patch release policies for timing and bundling

Step by Step Gathering of the Data

As we’ve stated, we wanted a methodology that others could repeat and get the same results. We’ve built our own database, but to facilitate that goal, here are the steps that can be followed to build your own set of data for analysis.

- A. Build out a spreadsheet of vulnerabilities for Windows Server 2003..
 - a. Sequentially examine each Security Bulletin released by Microsoft during the time period studied, not relying on the Security Bulletin search list provided. Microsoft Security Bulletins and the vulnerabilities addressed by them originate at: <http://www.microsoft.com/technet/security/current.aspx>.
 - b. For each Bulletin, read through and identify if Windows Server 2003 is affected. Typically, Microsoft includes a table in the Executive Summary of the bulletin with a row for each vulnerability listed by CVE Name and showing the Microsoft severity for each platform affected.
 - c. For each CVE Name, fill out the following columns. Note that a Bulletin addressing multiple vulnerabilities results in multiple rows:
 - i. CVE Name (e.g. CAN-2004-1028)
 - ii. MSFT Security Bulletin identifier (e.g. MS04-007)
 - iii. Date of Security Bulletin/Fix
 - d. Since we assume all components are present on WS2003, there is no need to group components into role groups as we do for Red Hat.

- e. Since we assume all components are present on WS2003, we do not do a validation step to see if the component is physically installed. We assume it is installed in all cases.
- B. Build out a spreadsheet of vulnerabilities for Red Hat Enterprise Linux 3 Enterprise Server (RHEL3ES).
- a. Sequentially examined each security advisory for RHEL3ES released by Red Hat during the time period studied. RHEL3ES security advisories and the vulnerabilities addressed by them originate at:
<https://rhn.redhat.com/errata/rhel3es-errata-security.html> .
 - b. For each security advisory, read through and identify and confirm that RHEL3ES is affected. The advisory identifier, affected component and associated CVE Names are typically all listed in the header of the security advisory.
 - c. For each CVE Name, fill out the following columns. Note that an advisory addressing multiple vulnerabilities results in multiple rows:
 - i. CVE Name (e.g. CAN-2004-1028)
 - ii. Security Advisory identifier (e.g. RHSA-2005:010)
 - iii. Date of Security Advisory/Fix
 - iv. Each package patched
- C. Gather Information to Calculate Days of Risk
- a. For each CVE Name on either server spreadsheet, look up the references listed at <http://cve.mitre.org>. For example, CAN-2004-0021 details are listed at <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0021>.
 - b. Follow each reference, examining the date of publication of the referenced web page that made the issue public. Enter the oldest date into the spreadsheet as “Date first public” and the URL into the spreadsheet as “First public reference”.
 - c. Since the CVE list does not guarantee to capture the first public reference, though often it does, cross check the references with any references listed in the security advisory or Security Bulletin. If an earlier public reference is found, use the earlier reference and date.
 - d. Additionally, search the Internet and common security newsgroups for public discussion of the security vulnerability and use that date and reference if found.
 - e. Add a column to the spreadsheets called “Days of Risk” and subtract the “Date first public” from the “Date of Bulletin/fix” to calculate the days of risk for that vulnerability.
- D. For both the WS2003 and RHEL3ES spreadsheet, add the ICAT severity listing.
- a. Download the latest ICAT Metabase from <http://icat.nist.gov/icat.cfm>.

- b. For each CVE Name in the server spreadsheets, enter a column value of HIGH, MEDIUM, LOW or UNRATED
- E. For the RHEL3ES spreadsheet, create a spreadsheet for each server role.
- a. Install and build a RHEL3AS system in the configuration being studied (e.g. database server role). Use the 'rpm -qa' command to check for each package that is patched with any security advisory during the time period.

With the above five steps completed, you should have spreadsheets capturing the list of vulnerabilities for each platform and server role for a given time period, along with the severity rating, date first public, first public reference and the days-of-risk calculation. From this, you can calculate counts, totals and averages as desired.

Conclusions

In our research, we are studying both quantitative and qualitative data that affects the vulnerability, and thus operational security risk of different server roles. In order to produce a meaningful comparison of platforms, systems will be examined in their default configurations and then looked at in minimal server role configurations. When the default configuration does not provide for a functional web server, systems will be configured according to manufacturer's directions.

When considering quantitative data, we will examine the number and type of vulnerabilities that have been reported for each platform. We filter these based upon the features and packages that would typically be found on a web server. For each vulnerability, we determined the total time that elapsed between wide public disclosure of the vulnerability and the availability of a patch that closed the vulnerability.

On balance, as security practitioners, we know that both the Red Hat and Microsoft solutions can be used to provide a secure solution, when deployed and administered with the right skills and under the right policy. However, customers and users of software need metrics that can help them make good security decisions for their networks and want ways to differentiate what sort of baseline security software development vendors can provide them with.

We believe that the methodology we have outlined meets our requirements of being repeatable and objective and establishes a baseline element of security research on which more advanced research can build to provide a full picture of software security risk.