Global Information Assurance Certification Paper

Craig Hawks

7/7/03

GSEC Practical Requirements (v.1.4b)

Case Study: Securing a Legacy eBusiness Application

1.0 ABSTRACT

I was assigned the task of reviewing an enhancement for a client's legacy eBusiness application. My task was to assure that the enhancement would not introduce any new security vulnerabilities in to the system. The enhancement was quite simple, so in an effort to add value I requested and received permission to perform an assessment the entire system. Performing that assessment provided an opportunity to test my SANS training. The following paper is a review of the assessment and subsequent remediation efforts of that system.

2.0 Before

For the sake of both simplicity and privacy I am going to use the acronym of LEA (Legacy eBusiness Application) to describe this application. LEA is an externally facing Internet eBusiness application that allows financial investment brokers access to consolidated financial investment portfolio information that has been collected from external sources. Being a legacy system, there was certainly newer and better technology available, but the user base was comfortable with LEA so the client was not inclined to upgrade.

LEA had been up and operational for approximately four years when my client contracted to add additional external users by way of three new broker companies. The necessary application changes to add the new brokers were minor and would take a very short time to assess and implement, so I suggested to the client that because LEA had been in production for such a long period of time, perhaps I should perform a detailed assessment of the entire application and environment rather than just assess the enhancement. My client was not opposed to this, but cautioned that my work needed to be completed in a very short period of time so that the scheduled introduction of the new vendor companies would occur as planned. I agreed to be quick and thus began my assessment.

2.1 Preliminary Assessment

Since time did not permit an all-inclusive assessment[1] of LEA, I felt I needed to hit the most important items as completely as time would allow. Authentication, authorization, confidentiality, and vulnerability would be my focus at the application level. Ports and patch level would be my focus at the Operating System/Server level. Although I would not actively look in other areas, if I stumbled across an anomaly I documented it for follow up later.

Before I could begin my assessment I needed to gain a technical understanding of LEA. To achieve this I met with the LEA project team and requested a system walk through. I also went on a search for documentation that would detail the system architecture, function, and interfaces. Although it took some time to locate all the diagrams and text, I did finally gather enough material to get started.

I concluded that the hardware architecture and infrastructure would be adequate for the additional users that this application was trying to support. The documentation, although thin, appeared on the surface to corroborate the project teams description of the overall system function as well as LEA's interactions with other interfacing adjacent systems. I would perform more detailed analysis in the days ahead.

Even though I was working in LEA's test environment I asked for permission in writing from client management before I began using the tools and techniques presented in the SANS Security Essentials course[2]. My first task was to footprint the LEA environment. It was during this exercise that I received my first surprise. According to the LEA system documentation, the LEA servers were supposed to be deployed in a separate segment of the corporate network with a firewall protecting the remainder of the corporation should a penetration against LEA be successful. However, my testing revealed that LEA was not on a separate network segment nor had the host-based firewall been installed! I feared that if the LEA test environment was configured incorrectly, then it was likely that LEA would be positioned and configured incorrectly in production as well. After talking to the client network configuration and web groups my concerns were confirmed. The firewall was not present in production and LEA was not on a separate network segment. This was a very high risk finding, so I quickly notified the LEA project team and my client management so that they could take the steps necessary to logically reposition LEA in the production environment.

Since my goal was to assess such things as vulnerabilities, patches and ports, I decided that the quickest way to accomplish this was to perform a penetration and verification test of the application and its environment. In order to accomplish this quickly I choose to use an automated tool as well as manual efforts of my own.

My choice was to use Nmap[3] to scan LEA and the LEA environment. And, I didn't want to strictly rely on automated tools, so I chose to perform some manual testing that might be best described as ethical hacking. My goal was to assess the application and environment and try to penetrate LEA and escalate my privileges and ultimately gain root access.

I was surprised at the quantity and type of vulnerabilities that I discovered. I found minimal authentication and that the password file was stored unencrypted in a public directory. I found there was little access control and that any

authenticated user could manipulate the URL string and traverse the entire application unencumbered. I found that there was little confidentiality. Sensitive user documents, with confidential customer information were stored unencrypted and unprotected in a common directory. The application formed the user document URL string, but this string could easily be manipulated so that a user could view or edit a document regardless of ownership. I found scripting errors in the application that revealed sensitive system information. I found many functions and fields that were susceptible to SQL injection and DOS attacks. The little transaction logging that did occur was stored in the clear in a public directory. The port scan found more open ports than I have space here to enumerate; few of which actually needed to be open. I also discovered that the LEA operating system, database server and web server had never been patched or upgraded. An accident just waiting to happen!

2.2 Vulnerabilities
Below are the most serious vulnerabilities that I discovered.

- **Authentication** – I found that LEA only employs basic access controls. User passwords are not encrypted thus leaving them open to view by any user. I found that the administration pages openly show the user passwords in plain text. Also, there is not a user lock out after repeated unsuccessful attempts nor is there a requirement for strong passwords. Consequently, LEA does not adequately protect the integrity of the data used in the application. The basic access controls in LEA do not assure the accountability of the individual logged on, therefore it cannot be ensured that only authorized users are viewing the files or performing authorized updates. Also, since LEA maintains both end users and administrative users with the same functionality, vulnerabilities transcend the user layer and migrate into the system level.

- **Unsecured Single Directory** - The unsecured directory in question was a directory where the consolidated portfolio information was collected and retained in client files. These client files were stored in one common directory. This is an insecure practice that can lead to unauthorized file access. (A better practice is to segregate the user files by client into separate directories.) My additional concern was that this single directory was not secured. This non-secured directory when coupled with the Parameter Manipulation vulnerability allowed direct assess to client file contents without enforcing any authentication or checking of whether the specific user is authorized to access the information contained in the file. Further, with the other vulnerabilities that I discovered in LEA, with very little effort any malicious Internet user could access all information in the client user files.

- **Parameter Manipulation** - This vulnerability allowed the user to manipulate the URL string and traverse the application at will. When

coupled with the Unsecured Single Directory vulnerability, the malicious user has an easy channel for violating the user file integrity. I also found many instances where a malicious user could manipulate the input fields and save the information to a file thus using the application to perform erroneous file updates that would appear normal and correct.

- **Input Validation and Control** - Input validation is useful for assuring that information entered via the application conforms properly in both type and structure. LEA performed no type of input checking; hence the user files often contained meaningless or erroneous data.

  Improper input control allows the malicious user an easy entrée for System compromise. Some of the more likely attacks are the Denial of Service[4] and the Buffer Overflow[5].

- **Configuration Exposure** – These errors disclose sensitive system information. My testing allowed me to find LEA's DNS information. When the database is unavailable and during other error conditions the header information is displayed in error messages as the source of the system error. Some simple parameter manipulation (see above) allowed additional content to be returned. The content referenced "include files" which in turn contained the database DNS. The DNS contained the username, password, and database name used to access the database. Using this DNS information allowed an additional path of compromise of the system.

- **Missing Logout Function** - LEA included a Login function but did not include a function to end the user's session. This is a standard requirement for authenticated sites to prevent the user from leaving the site and having their credentials remain valid. Without the Logout function, it is easier for a malicious user to impersonate a valid user and access user files.

- **Missing Change Password Function** - LEA did not include a function to enable the users to change the password assigned to them by the system administrator. Without allowing the users to change their passwords, accountability cannot be ensured because both the administrators and the users know the users' passwords and compromise of the system is easier because the passwords never change.

- **Unknown Functions** – I discovered a partially working Manager function and a partially working User Administration function on LEA. Although the names of these functions seem to suggest control access to the management site of the application, it was determined that attempting to use the functions had only part of the desired effect on LEA.

- **Test Page left on Server** - Several test pages were found in the application. These test pages appeared to attempt some database activity with an invalid data source. Test pages can disclose sensitive information and assist a malicious user with attacking the application. These pages should have been removed prior to roll out to production.

- **Email** – LEA had an obscure email function likely left over from a previous version. A malicious user could use this function to send emails with dubious content. The current user base had no need to send email from LEA, so I made a notation to recommend removal of this function.

- **Log File** – The log files for the application contained numerous pieces of information such as logon ID, logon time, transaction summary, etc. This type of information should be encrypted and located in a protected directory. This was not the case. The log file was unencrypted and stored in a publicly available directory.

2.2 Risk

Let's establish some definitions so that we can discuss risk and risk assessments[6].

- "A **risk** is the product of the level of threat with the level of vulnerability. It establishes the likelihood of successful attack." [7]
- "A **threat** is the potential for violation of security, which exists when there is a circumstance, capability, action or event that could breach security and cause harm."[8]
- "A **vulnerability** is a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy."[9]

It is important to honestly assess the threat level and vulnerability level. Although subjective in nature, a properly performed risk assessment will allow the client and team to focus the correct resources at the most critical issues.

The enumerated vulnerabilities are in the table below with the threat and vulnerability levels indicated.

Vulnerability	Risk Level	Threat Level	Vulnerability Level
Incorrect server location and no firewall	High	10	10
Unsecured user directory	High	10	10
Parameter Manipulation	High	8	8
Input Validation	High	8	7
Authentication	High	8	8
Configuration Exposure	Medium	6	7

Missing Logout Function	Medium	6	6
Unknown Functions	Medium	5	5
Test pages left on server	Medium	5	3
Missing Password Change Function	Medium	5	5
Email vulnerability	Low	3	3
Log File stored in the clear	Low	3	3

At point in my assessment I felt that it was time to re-engage the LEA project team. Although their knowledge of LEA was certainly better than mine and I felt that their perspective on the culture of the client and customer would an asset. I gave the team a short lesson in risk assessment and together we assessed and ranked the vulnerabilities and then collaboratively developed a remediation plan. It was important also to involve the development team in this process because they would be the individuals performing the remediation tasks.

2.3 Remediation Planning

In order to move forward I felt that we needed to develop some recommendations for remediating these vulnerabilities. Unfortunately because of the age of the application and nature of the vulnerabilities there was no single remediation plan that would be both quick and secure. Hence, I developed three possible remediation plans: a complete rewrite of the application, a partial rewrite of the application, or a wrapper type (internal IDS or application based firewall) solution.

The most costly and time consuming solution would be a complete rewrite of LEA using a more robust and secure design with the inclusion of secure coding practices since it appeared that the design and coding practices were at the root of most of these vulnerabilities. The development team after reviewing the code agreed with me. I felt that a complete rewrite would be the most certain in terms of remediating all vulnerabilities, so this became my plan “A”.

Plan “B” would be a less costly, but less secure solution; a partial rewrite of LEA choosing the highest risk items for rewrite. Although this is would not be a perfect solution, it made sense when weighing cost versus benefit verse risk.

Lastly, I had what I thought was an exotic, but potential useable idea. I thought that perhaps a “wrapper” type of solution; one of the Application Level IDS, Application Firewall, or Reverse Proxy type products might be useful and provide a more secure, less costly solution in the shortest time of all. However, the wrapper solution wasn’t something that I could be certain would remediate the vulnerabilities satisfactorily without actually doing some in depth testing. So this became my plan “C”.

Prepared with all of my findings and team recommendations I scheduled a meeting with client management. Although the client had been kept aware of my

findings as they occurred, they couldn't help but be overwhelmed by the quantity and severity of the vulnerabilities when presented collectively. I had come to the meeting prepared to discuss possible strategies for elimination or mitigation of these vulnerabilities and that helped to lessen some of their concerns.

My thought going into the meeting was that the client would lean toward plan "B"; the partial rewrite. They were cost conscious and just becoming Security aware so I wouldn't have been surprised if they had chosen what to their eyes would have been the least expensive solution. However, to my surprise the client wanted to explore the wrapper type solutions; plan "C"! They wanted to see how completely this solution remediated LEA and then perhaps leverage the solution onto other "problem" applications and servers. I readily agreed to this approach because I wanted to explore these products a little better. After some discussion the client agreed to set up a test environment with servers and software to mirror the LEA production environment. Once we narrowed the vendor field to a reasonable number we could then obtain demo code and see how effective the wrapper type tools would be for remediating LEA.

3.0 During

The client was quite generous in providing equipment and facilities for the test environment and the LEA development team was excited about the opportunity for new learning. I was especially interested to see how the wrapper solutions would perform, but I also was keenly aware that remediation of the vulnerabilities needed to be the primary focus and I couldn't get caught up in technology.

3.1 Solution Testing

A search of the Internet turned up several dozen vendors of wrapper type solutions in the form of proxy, reverse proxy, Intrusion Detection[10], Intrusion Prevention, etc, type products. It would have been impossible for every member of the team to review every product, so to evenly distribute the burden I divided the vendors into categories of: Application Level Intrusion Detection, Proxy/Reverse Proxy, and Intrusion Prevention; and assigned members of the LEA team to the various categories to perform reviews of these products. The team members knew they needed to review the product documentation looking for function and applicability to solving our list of vulnerabilities and they did an excellent job of narrowing the field. They compiled a list of products that looked good "on paper" and then together we narrowed the list to two products for a hands-on evaluation.

The first product was a Host Based Internal Prevention Solution (HBIPS)[11]. The second was a Web Application Firewall Solution (WAFS)[12]. Although my intention is not to disparage either of these products, my comments might be construed as such, so to avoid any legal problems I will only be referring to them by their functional acronyms.

The HBIPS was the first wrapper type of solution that we tested. This particular

HBIPS uses a combination of signatures and rules to perform its function. The user sets up the rules, the HBIPS learns the application signatures and use both together to perform the Intrusion Prevention function in a kind of defense in depth. (Detailing the function of these products beyond the scope of this paper, but I included some links at the end of this paper for anyone that desires additional information[13].)

WAFS was the second product that we reviewed. This product referred to a built in set of logic to determine the types of legitimate requests that LEA should process and the WAFS would filter out everything else. WAFS also a learn mode so that additional requests can be filtered out as well as provisions for relaxing the rule set when necessary so that requests WAFS has incorrectly chosen to filter out, can be allowed through to LEA.

To keep an appropriate level of consistency, the testing that we would perform would be the same as we performed in the earlier penetration testing, scanning and ethical hacking. In preparing to test, we were surprised to find that there was considerable set up and fine tuning with each of these products. We kept careful notes knowing that we would be saving much rework later on if we choose to roll out either product.

Each product performed as advertised and each product had its strong points and weak points. Our task now was to determine if either product remediated enough of our vulnerabilities to justify a purchase.

3.1.1 Solution Testing Results – Unsecured Single Directory

The unsecured single directory was a basic design flaw. We certainly didn't expect either product to create and maintain subdirectories, but we had hoped that perhaps one or both would be able to control access to the files. Unfortunately this was not the case. Due to the poor design of the system a properly authenticated user could still view user files that they did not have authorization to view. The only way to make these products control unauthorized access would be to set up a rule for each file which would take many hours of initial set up and then add an additional rule for every new file created on the system, and that would be completely unreasonable. However, we did take note that if we changed the design and segregated user files into separate directories, then a rule could be set up by user, which would assure that only the correct user would be viewing a file. The overhead of performing this function would be minimal.

3.1.2 Solution Testing Results - URL Parameter Manipulation

Controlling URL Parameter Manipulation is an area where both products functioned at their best. With minimal instruction we were able to learn how to compose the rule set for properly formed URL's and legitimate URL's. Thus it was fairly easy to configure the HBIPS. The WAFS learn mode was also straightforward so that after minimal set up it too was ready for testing.

We repeated the initial test of URL parameter manipulation and found that both products did an excellent job of blocking or disallowing this type of activity. We also made sure to test the positive; that valid web addresses were permitted and both products performed perfectly.

3.1.3 Solution Testing Results – Input Validation

Field level input validations were one of our medium risk areas. We needed to be sure that each field only allowed the valid range of inputs. Each product could be configured to partially prevent compromise of the system, however neither was able to completely remediate this vulnerability. Again, this was due to the design and programming of LEA. Had the initial programmers put in limit and format checking, then either of these products would have been an excellent choice to do macro level validation with the LEA performing the micro level checking.

3.1.4 Solution Testing Results – Authentication

Neither of these products was able to remediate the authentication issues. Strange as it seems both the HBIPS and the WAFS viewed authentication requests as legitimate valid operational requests and passed them into the application rather than aid in refining or excluding the user based on finely tuned rules.

In order to remediate this vulnerability we would have to build better authentication into the application, which would require a design change and coding.

3.1.5 Solution Testing Results - Configuration Exposure

HBIPS has the inherent ability to completely remediate this vulnerability. Within the bounds of our tests, any messages the LEA tried to display to the user were trapped by the HBIPS and a generic error message that would not reveal any configuration information was displayed to the user.

WAFS was able to remediate this vulnerability, but did require some fine-tuning to trap all of the application generated error messages.

3.1.6 Solution Testing Results - Missing Logout Function

Neither product was designed to provide logout functionality, so we weren't surprised by the fact that there was no way to configure either product to perform this function. This would be another function that we would have to design and code ourselves.

3.1.7 Final Analysis and Recommendations

I assembled all of the test results and created a report for client management. My analysis of the testing results showed that both the HBIPS and WAFS products remediated or mitigated most of the vulnerabilities and I felt that both products were a good value. Ultimately I recommended the HBIPS because it was slightly

easier to configure and its cost was slightly less than the cost of the WAFS.

I recommended that the development team redesign and recode the directory structure into multiple client directories. This would all the user files to be segregated by client. Once the files were segregated I recommended that the directories be encrypted and secured at the operating system and application level. Properly performed, these actions would completely remediate this vulnerability.

I recommended that the authentication scheme be redesigned and recoded with encrypted passwords, a change password function, and a logout function.

I recommend remediation of the Input Validation and Configuration exposure vulnerabilities through redesign and recoding. Both were judged to be high risk and I felt it necessary that they be remediated.

I wasn't certain if the client had relocated the servers to a separate, protected segment of the network and installed the host-based firewall as yet, so I included that recommendation in my report.

I did not recommend remediation of the Unknown Administrator functions. They were judged to be of medium risk and would have taken quite of bit of cost to remediate.

There were some medium and low risk items that I felt could be handled quickly so I included them in my report. There was an email vulnerability that could allow a malicious user to redirect a LEA generated email. Since generating an email from LEA was a seldom used function, I recommended that the email function be removed thus eliminating this vulnerability. I recommended that all of the test pages remaining on the production server be removed. I also recommended that the system log file and log file directory be secured.

I submitted my report to client management and followed up with a meeting to formally present my findings. I was pleased to find that my client elected to go forward with all of my suggestions. So, the LEA development team was instructed to work through the redesign and recoding while client management placed an order for the HBIPS product. And to avoid a repeat of the design and coding errors currently in LEA, client management arranged a "Secure Coding" training program for the development team.

So as not to inconvenience the three new broker companies, my client went forward with adding the broker companies to the production version of LEA while remediation continued in the development area.

3.2 Remediation

The redesign, recode and retest efforts took seven weeks to complete. During

that time the clients Web and Network group relocated the LEA production and test servers to a separate network segment and installed a host-based firewall. Concurrently the client obtained the HBIPS media and set about to load and configure it on the LEA server in the same manner as it had been configured during the Solution Testing phase of the project.

The remediated LEA code was rolled out to the test environment where we conducted a final set of tests to assure that the redesigned and recoded system, in concert with the HBIPS, remediated the vulnerabilities at the level that we anticipated. To my relief we were successful in that effort.

Some of my final work on this project consisted of performing port scans and insisting that unneeded ports be closed and remain closed. I tested and examined to assure that the Operating System, Database and Web Server were patched and upgraded to the appropriate level. I also counseled the client on the importance of keeping the servers and software patched and current. I also worked with the client to develop a regular schedule of scanning and testing to assure that ports remain closed. With this work complete, the remediated LEA was rolled out to production.

4.0 After

The client was very pleased by the final outcome of this remediation effort. The overall risk in LEA was greatly reduced. Prior to remediation LEA contained five high-risk items, five medium risk items and two low risk items. After remediation LEA only contained one medium risk item.

There were many additional wins from this project. The client became much more security aware as a result of working through the issues in LEA. The client made certain that all future development was performed using the learning gained from the Secure Coding practices course that the programmers took prior to remediating LEA. The client developed a practice of installing, testing and rolling out Operating System patches and upgrades on a regular basis. The client developed a system of periodic port and vulnerability scans. And the client is in the process of developing a corporate security practice.

---

[1] Slater, S, PhD, CISSP slater@nsconsult.com, A Practical Guide to Application Security Reviews,

[2] Cole, E., Fossen, J., Northcutt, S., & Pomeranz, H. SANS Security Essentials with CISSP CBK Version 2.1. USA: SANS Press, 2003

[3] Nmap Stealth Port Scanner;
http://www.insecure.org/nmap/

[4] Carnegie Mellon Software Engineering Institute,

http://www.cert.org/tech_tips/denial_of_service.html

[5] Carnegie Mellon Software Engineering Institute, http://www.cert.org/homeusers/buffer_overflow.html

[6] Boyle, K. kip_boyle@jeffersonwells.com, & Menninger, M. macr.menninger@amnu.net, Scaling Information Security Risk Assessments., Jefferson Wells International Seatle, Washington

[7] Cole, E., Fossen, J., Northcutt, S., & Pomeranz, H. SANS Security Essentials with CISSP CBK Version 2.1. USA: SANS Press, 2003. Appendices A-147, 2003

[8] Cole, E., Fossen, J., Northcutt, S., & Pomeranz, H. SANS Security Essentials with CISSP CBK Version 2.1. USA: SANS Press, 2003. Appendices A-147, 2003

[9] Cole, E., Fossen, J., Northcutt, S., & Pomeranz, H. SANS Security Essentials with CISSP CBK Version 2.1. USA: SANS Press, 2003. Appendices A-141, 2003

[10] Best Practices - Intrusion Detection Systems http://www.itsc.state.md.us/info/InternetSecurity/BestPractices/Intrusion.htm

[11] Innella, P.,& McMillanm, O., Tetrad Digital Integrity, LLC, An Introduction to Intrusion Detection Systems, http://www.securityfocus.com/infocus/1520

[12] Englund, M., Securing Systems with Host-Based Firewalls – Implemented With SunScreen™ Lite 3.1 Software SunIT Network Security Group Sun BluePrints™OnLine - September 2001, http://www.sun.com/solutions/blueprints/0901/sunscreenlite.pdf