CASE STUDY : ETHICAL HACKING

Braden Simpson
braden@uvic.ca
V00685500

March 3, 2013

1 INTRODUCTION

This paper introduces a selection of vulnerabilities in software systems as well as organizations that can be leveraged by hackers to produce unwanted results. Consequences of hacker activity can range between anything from information leak to denial of service to data corruption, and more. The vulnerabilities discussed in this paper are the following :

- Social Engineering
- Improper Permissions for Configuration Files
- Phishing
- Application Vulnerabilities
- Hosting Multiple Services

2 SOCIAL ENGINEERING

Social Engineering is the leading cause of 'hacks' in software and other industries. It is very prevalent and effective because it capitalizes upon the most error prone processes, human process.
Social Engineering is when a person manipulates another human being, through any medium (chat, in-person, email, etc.) to obtain access, information, or perform actions that are not what the company or organization intended. It is essentially convincing people to do things that are bad. Some examples of social engineering attack techniques from PC World[1] are:

The Familiarity Exploit When the attacker deceives people into thinking that he/she should be present by being familiar with the system or organization.

Hostile Situations When the attacker appears to be hostile, or angered, they tend to face a lot less opposition. This can be beneficial when trying to convince someone.

Gathering Information The main factor of successful social engineering is knowing the whole organization. The more information known will help in convincing the people working there that it's a legitimate request. One of the most common ways is for the attacker to get a job at the organization they plan to attack.

One of the most common social engineering attacks is on customer service departments, this is because they are the most likely people to break. This is because their job by definition is to make the end users experience as good as possible, and the attacker can use this to their advantage.
Baiting is another form of social engineering, which takes place when an attacker will give someone at an organization an item that appears to be safe, and it is in fact malicious. An example would be a person at a conference giving out malicious USB sticks as a promotion.

2.1 Solutions

There is no surefire way to stop social engineering, as it attacks the people which can only be secured to a point. The best way to avoid it is to offer training sessions for all the employees at the organization for what social engineering attacks look like, and how to differentiate them from real requests from users.
Another way to avoid social engineering attacks is to write strict policies for interactions with people, outsiders and employees included. Some policies like 'Never give out your password, even to IT,' are a good example of ways to stop unknowing employees giving out their password to an attacker impersonating an IT department.
Investigating and suspecting all people, and being scutinous of the authorization information given by people is another method of improving security against social engineering.

3 Improper Permissions for Configuration Files

Improper permissions on configuration files is when a file containing potentially sensitive information may be stored in a place that could be accessible to the internet, including search engines and crawlers. Although this risk is usually easily mitigated, it is a serious problem and has plagued the software industry since the dawn of powerful productivity tools and frameworks. One very common example of this is the problem outlined in the presentation slides, is unprotected .RDP files.[2] The .RDP (Remote Desktop Protocol) files are configuration files for Microsoft's Remote Desktop clients. Once these are obtained it could be possible for anyone with the

file to potentially connect to servers. The example shown is using a search engine like Google to specify those filetypes and scan for available RDP files.
The example noted, among other exploits of this are problems with improper permissions on sensitive files such as .RDP configuration files. This example can extend to many other applications such as:

- Webserver configuration files
- DNS files
- Important internal memos
- Other service configurations

3.1 SOLUTIONS

One solution to this are to construct policies on where to put sensitive information (make sure it's not on your webserver in a public directory). Another could be to eliminate the need for those sensitive configuration files if possible (store less RDP files and manually enter some of the data). Another possible solution could be for the search engines such as Google to not index those sensitive files, although this option seems less likely as it is not Google that is being hacked.

4 PHISHING

Phishing is the act of deceiving a pereson into thinking that the system they are attempting to use is a legitimate system, when really it is another system entirely, which is going to do unwanted actions with the information the user enters.
This is a very common attack and is very successful against novice users. The most common is impersonating a website that requires a login, by stealing the front end code for the website (for example facebook.com). The attacker would then somehow distribute the link to their malicious version of the website and the user would mistakenly enter their credentials. This exact attack happened to UVic, with their login page a few years back, and it was shown in the presentation that the attacker setup a *uvic.ru* website to try and convince users it was legitimate.

4.1 SOLUTIONS

The best solution to the phishing problem is for the users to be well educated on what a phishing attempt looks like, and how to detect if a website is in fact legitimate. Users must be trained to diligently check the URL, since the attacker may completely alter every other aspect of the site.
In addition, it is beneficial for a corporation to purchase all similar domains for their website (for example facebook.com would purchase facebook.co as well) to prevent attackers from using a similar domain name as an attack website.

5 Application Vulnerabilities

Application vulnerabilities include a broad range of attacks, all focused on the application running which may present an vulnerability which can be exploited in a number of ways, including:

SQL/Code Injection This attack is done by exploiting an aspect of an application that takes user input and then processes it somewhere else in the application. Since that is such a common task for applications, and databases are so prevalent, this type of attack is very common. The attack is very simple to perform because it simply consists of running many inputs against the system in the hopes that the system has a security flaw.

Cross-site Scripting(XSS) This attack is similar to the previous, except it usually consists of one user entering malicious code that will then be run by another user's computer unintentionally. This is very commonplace for websites that take user input and display them to other users. An example could be a attacker posting a message on a forum, and at the end of the message, the attacker puts malicious javascript that steals the users cookies and emails them to the attacker. Then whenever a normal user visits that post, they would have their cookies stolen and sent without their knowledge.

Cross-site Request Forgery(CSRF) This is similar to XSS, except it is usually more targeted. It is when a an attacker creates a link that performs a trusted action, and disguises that action as something harmless. For example a Mexican attacker had embedded an action inside an email picture to rewrite a DNS entry for the bank to point to their malicious website rather than the banks.[3]

In addition to the vulnerabilities listed, attackers can use tools such as metasploit to automatically test for attack vectors on an address and attack them with an array of attacks inlcluding injection and known exploits in common programs, such as java, flash, etc.

5.1 Solutions

Countermeasures to application vulnerabilities are always very difficult to perfect. The majority of the countermeasures arise from people learning from successful attacks, this is because most are the result of bugs in the frameworks written.
In order to protect a system from application vulnerabilities, the system should have a schedule for keeping all frameworks, libraries, and programs up to date. All the applicaton's developers must follow the best practices and standards for developing in those frameworks. The sytem's must have a team of penetration-testers attempting to break into the system on a routine basis, in order to find the exploits before the outsiders do.

6 Hosting Multiple Services

This is a common issue. It occurs when many services, such as Database management systems, Secure Shell, File Transfer Protocol, and many others, are running on the same server. The more services that are running on a machine, the more vulnerable it becomes; this is due to the ports that must be open for the outside world to talk to the machine on. An open port means that there is another attack vector for an outsider, and it means that the security for the whole system could be in jeopardy. The example given in the presentation is that there are a very large amount of services all running on the same box, smtp, dns, http, pop3, ldap, and others. This was found out by a simple port scan on the IP in question.

6.1 Solutions

The problem arises mainly when a company or organization wants to economize by making one server do everything, instead there should be multiple which do different tasks. Another solution is to ensure that nothing is running on an open port that doesn't absolutely need to, for instance most networks won't have more than port 80 (http) running. As well, the UVic network disallows opening ports up, simply to prevent this vulnerability.

References

[1] J. Heary, "Top 5 Social Engineering Techniques, PC World," Victoria, BC, Canada, Mar. 2012. [Online]. Available: http://www.pcworld.com/article/182180/top_5_social_engineering_exploit_techniques.html

[2] E. van Wiltenburg, "Presentation Slides, UVic IT," Victoria, BC, Canada, Mar. 2012. [Online]. Available: http://www.ece.uvic.ca/ henrylee/2013/07/07EthicalHacking.pdf

[3] W. A. Sec, "Web Hacking Incident Database, Web Application Security Consortium," Victoria, BC, Canada, Mar. 2012. [Online]. Available: http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database