Random Ramblings on System Security

Mark Sitkowski
Design Simulation Systems Ltd
http://www.designsim.com.au

My original intent, was to write a kind of smart-ass open letter entitled something like "Dear JP Morgan, Target, Neiman Marcus, Michael's, Home Depot etc" and other victims of recent system breaches and data theft, pointing out the various mistakes they made in putting together their IT systems.
After some consideration, I decided that it would be less patronising to merely present best-practice in system security, as a series of points, with a short discourse on the relative merits of each point.
I promise not to insult your intelligence by telling you to close all ports except 80 on your firewall, or by telling you how to set up the permissions on the operating system.
Instead, we'll list all the components of a bulletproof IT system, and you can choose to omit various parts, if the extra risk is justifiable.

A Bulletproof IT System

Let's assume that we want to design a hypothetical IT installation, which can be considered to be either a retail operation, or a financial institution, equipped with either POS or ATM terminals, operated by bank cards or store cards.
The system will possibly consist of a web server, with your landing page and login page, a database server, a business server, which contains transaction records and account details, and an authentication server, which confirms the identity of each system user.

The Hardware

- Don't run your web server on an Intel processor. Use Sun, Hewlett-Packard, or IBM proprietary hardware.

I did say "Bulletproof", so this recommendation has to be included, but, before you say 'As if!", ask yourself this question:
"In what language is malware written?"
If you answered "x86/x64 assembler" then you win a prize.
Malware is written to run on an Intel CPU, since these are universally deployed in machines throughout the world, and it is cost-effective for the hacker to concentrate his efforts on malware which will attack the largest target market (no pun intended).
So, even if the bad guys manage to plant their rubbish on your machine, it won't run.
Of course, this is, at best, security by obscurity since, when enough companies have made the switch, hackers will learn how to write malware for other CPU's.

However, given the current market forces, this is unlikely to happen in the next decade, which may justify the cost of removing this large element of risk.
If I ran a bank, I'd do it.

The Operating System

- Use Unix

If you can't use Unix, stop reading now, and go and find something else to do.
Unix was designed to be used by hundreds of university students and software developers, with the express intent that they should be incapable of interfering with each other, or breaking each other's toys. The concept of permissions saw to that.
Unix doesn't suffer from viruses, since the worst a virus can do, is trash the environment of the person who introduced it. It can't replicate, because it doesn't have permission to do so, and it can't infect the boot sector, or the memory, for the same reason.
PC's were designed to be Personal Computers, and neither DOS nor Windows was designed to be used by more than one person. This is why they had no network capability, until MS ported the BSD socket libraries to Windows (remember 'winsock.h'?).
Some people don't like Linux, because it's a bit too home-grown, and they'd like the warm fuzzy feeling of being able to contact a support person, if something falls apart.
If you really must use an Intel CPU, these versions of Unix (in order of preference/support will do it:

- Sun Solaris. Easy to install, excellent support.
- Xinuos, derived from SCO OpenServer (reportedly, used by McDonald's, Pizza Hut, NASDAQ et al)
- Apple OS X. Well, it is Unix...
- CentOS (okay, so it's Linux, but it's very robust).

The Network

- Each server on its own subnet

It's a fundamental law of system design, that the only thing accessible from the web, should be the web server. The other components, like the database server and business server should be totally invisible. In other words, it should be impossible to type a URL into a browser, and access any of the other system components.
We do this, by running everything except the web server on a separate subnet.
It works like this:
The web server (probably apache) is configured to access pages from a VPN containing the business server. The business server is configured to only accept connections from the CGI of the web server. As far a the user's browser is concerned, its address bar shows the pages as if they were resident on the web server. If the user types the apparent address in the address bar, apache will respond with '404 – page not found'.
Otherwise, you'd be able to directly access bank balance sheets, by typing a URL.

The apache Server

- Remove GET permission from the cgi-bin directory
- Remove POST permission from the htdocs directory, and the icons directory
- Remove any files from these directories which shouldn't be there, such as backup copies of web pages.

Backdoors are planted with POST queries to the htdocs and icons directories. Utilities such as 'wget' are used to suck all the files from any accessible directory, and recreate a copy of your website, which is then used to spread malware, and recruit members for a botnet. Additionally, all the web pages retrieved from your htdocs directory get examined by the hacker, (especially the javascript and hidden inputs), for clues as to how to hack your system through your CGI.

The Application Software

- Don't use PHP

Around 90% of all hack attempts exploit known vulnerabilities in PHP and applications written around it, such as Joomla and WordPress.
Attack queries usually attempt to overwrite index.php, wp-login.php and a whole bunch of images in WordPress and Joomla.

- Only use compiled code for CGI executables

Anything which runs in an interpreter, such as shell scripts, perl scripts, ruby or PHP scripts, can have code understood by the interpreter added to it. This is how SQL injection and cross-site scripting is done.
In this context, if your website uses forms, make absolutely certain that, whatever collects the form data is not an interpreted script.

- Only do trivial form integrity checks in javascript.

Javascript can be read directly in a user's browser, and rewritten by a hacker. If you do something dumb, like user authentication, in javascript, you'll find yourself with a few extra unexpected users logged in to your system.

BYOD – The Enemy Within

- Don't BYOD.
  Turn off DHCP or, at least, limit it to a few known, trusted MAC addresses.
  Resist the attempts by HR to make you feel guilty, and point out that other people's money is at risk, and that the company's hardware is good enough for the users.
  Sure, MAC addresses can be faked, but the risk of that happening is less than the risk associated with throwing free IP addresses at anyone with a laptop.
  If you permit uncontrolled BYOD, you deserve to get hacked. Even if everyone is security-conscious and trustworthy, they will be going home with your data on their devices. Ask the FBI about a certain operative, who left his laptop on a park bench...
  Also, nobody stays in one job forever. When they leave the company, perhaps to join your competitor, they will be taking your data on their mobile phones, slabs and laptops. Of course, you can always wipe the hard drive.
  Yeah, right.

POS Terminals, ATM's and other entry points

- See "Authentication"

Authentication

- How secure is username/password?
  My granny can write you a man-in-the-middle script, which will collect these things by the hundred. She can also write another script, which will use them to login to your system. Especially, if you let her use her own laptop on your WiFi.
  So how do you stop this?
  Simple. You make sure that any information collected by granny is useless. Don't just provide a dumb text box, into which the user types the password, because the row of asterisks fools nobody. Granny's malware is listening to what you type on the keyboard, not looking at the text box, so you need to eliminate the keyboard, and the typing.
  Take the uppercase and lowercase alphabets, scramble all the letters, and display them for the user to choose those making up his password.
  Wait! Granny's malware can read the machine's frame buffer, and identify the letters as they're selected. Remember what happened at Target? That's how the criminals managed to lay hands on millions of credit card numbers and PIN codes.
  That's why you display both halves of the mixed alphabets as two touch/click-sensitive bitmaps. That's better. Now, reading the frame buffer reveals no ASCII characters, meaning that, to get the password data, the malware would need to run the bitmap through an OCR, and figure out the coordinates of each letter. Having done that, there would be 26 candidates for each letter of the password, which would give 208 possible passwords. Of course, there's an incrementing time delay for every wrong login, and a lockout after too many of these.
  The next time the user logs in, the letters are scrambled in a different order, so the hacker has to start from scratch.
  Oh, yes, nearly forgot. You don't transmit the sequence of clicks as such, just the hash of their sum.
- Don't use biometrics
  All biometric data is a combination of your username and your password, since it uniquely identifies you.
  Despite it sounding like the perfect uncrackable method of identifying a user, just remember that biometric data is stored and transmitted in digital format. This means that the simplest man-in-the-middle attack (as performed by my granny, earlier) can save and store for reuse, a username and its related biometric data. Much more alarming, however, is the fact that, if the biometric data is stolen, the user is permanently compromised, with no socially-acceptable way of changing the biometrics. Not long ago, the FBI had its entire biometric database stolen, which not only meant that they had to fall back on username/password logins, but also made most of their agents identifiable by possible enemies. Not a good scenario.
- Two-factor authentication at its most basic is a card and PIN
  Ask customers of the companies mentioned in the opening paragraph of this piece how secure that is.

Many banks will send out the second factor as a random number in an SMS message to the user, which he then has to enter in a special box on the authentication/authorisation form. This is highly secure, but inconvenient, in that there's always a delay, and it involves the user in an extra operation. How about a different, transparent second factor? How about using the user's device as the second factor?
Each device, as viewed by the authentication system, is a unique combination of device type, operating system, CPU speed, graphics adapter and internet browser. These parameters can be combined into a signature, which is so unique, that an operating system update, or even a browser update will alter it. It's totally transparent to the user, and just means that he has to register all the devices he's planning to use to access his account.

Intrusion Detection System

- Use content-based, not rule-based systems
  A rule-based system refers to a huge table of known hack sites, before deciding whether to allow the connection. Apart from the fact that a huge number of available systems are just badly-written interpreted scripts, with the response time of an offshore call centre, they suffer from the obvious drawback that the number of "known" hack addresses is far outweighed by the unknown ones. Yes, you do get updates, but the hackers get more than you.
  A content based IDS looks at the incoming query itself, rather than trying to guess whether the source address is good or bad. If it sees, for instance a string like "../../../" in the query, it's a fair bet that it's not a potential customer. Similarly, if it sees a query partly written in hexadecimal ASCII codes, it again assumes that the sender is after your savings.
  Best of all, having identified a hacker, the IDS can add a firewall rule to block his address. No need to wait for a third-party update.
  Such an IDS is described in detail here:
  http://www.linkedin.com/pulse/article/20140927080143-57394917-a-gentleman-s-guide-to-intrusion-detection-and-protection?trk=mp-edit-rr-posts

In Conclusion

A few years ago, I was on a training course on the island of Guernsey.
Lured by the descriptions of the wild nightlife, on the neighbouring island of Jersey, a few of us decided to take an evening flight, on one of the strange wood-and-paper aeroplanes, called 'Islanders', that used to make the trip, at the time.
While we sat in the hotel bar, waiting for the airport bus, a storm of enormous proportions blew up, with lightning, howling winds and horizontal rain. Then, a figure came towards us, and said,
"Look, I have to go, since I'm the pilot. You guys still have a choice".
Fuelled with copious quantities of the local ethylene hydroxide, we ignored his advice, piled on the bus, and made the trip.
This article is a bit like that. My company is in the security business, so we can't afford not to implement all of the measures mentioned above. Your company probably isn't, so you can choose what works for you.