The Application Security Pyramid - Neighborhood Watch

Posted At : April 25, 2006 1:01 AM | Posted By : Mark Kruger
Related Categories: Security

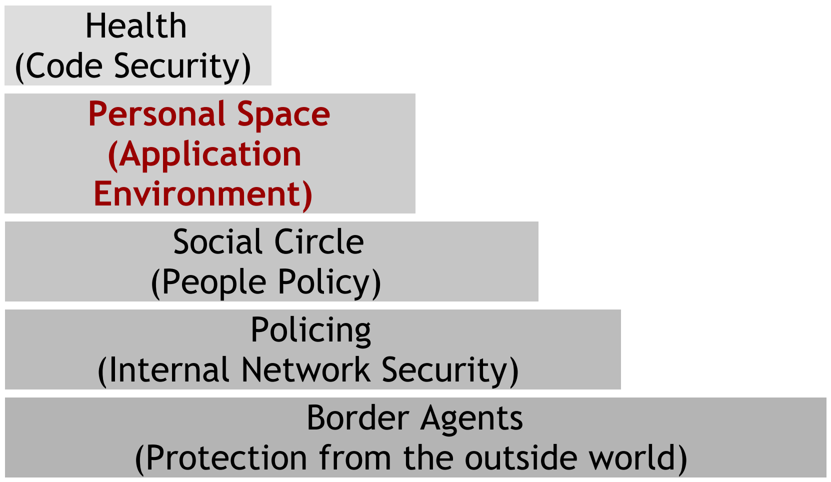
This post is a continuation of a 5 part series on security called "The Application Security Pyramid". The introduction introduced a new metaphor for dealing with security that loosely mimics Maslow's heirarchy of self-actualization. In Part I I discussed the importance of "border patrol" technology to safeguard your network. In part II I discussed internal Policing and People Policy. In this post we will deal with the importance of maintainaing a secure "environment" for your application.

- Intro
- Part I
- Part II
- Part IV

Neighborhood Watch - Securing you system

So you have effective border controls, and effective internal policing. Your network is safe from the outside and your network is controlled on the inside. Good for you. Let's talk about *your personal safety*. In the real world to really feel secure you need more than just a nice country, effective policing and civil services. You also need a *sense of internal security*. Your house and neighborhood should be a safe haven for you.

There are different kinds of security when you are talking about your personal well-being. Not only are you concerned about safety from crime, but you are probably also concerned with your health (feeling secure from disease) and your shelter (safe from weather and disaster). In the same way, *your application environment* and your *code security* have a direct effect on the level of vulnerability in your application. Let's look at the environment.



Application Environment

It's easy to forget that the *neighborhood* where your application lives is important. By neighborhood I mean the systems that your application interacts with directly. In planning where your application lives it is important to consider and define exactly what controls are in place for security of the whole system.

Define the Level of Exposure

Some web servers are set up as bastion hosts. A bastion host is typically a server hosted in a DMZ with a high level of exposure to the public internet. A DMZ is a separate subnet on your network that is fire walled away from the rest of your network and is allowed access only by very specific firewall rules. It should be noted that even though you are running a web server as a bastion host it does not mean that there is no firewall protection for this server.

The idea behind a DMZ is not to leave the servers completely exposed. Rather, it is a segregation technique that allows a *higher level of exposure*. Since this higher level of exposure naturally engenders higher risk, a separate subnet (a DMZ) with restrictions allows a network administrator to minimize any damage that might result from a breach. *Note: we will be using Windows 2000 server in all of our examples*.

Define Rules Between Systems

Inventory the processes and systems needed to run your applications and enable only those permissions that are necessary for the specific tasks involved. A good example is the database. If you need access to the database server you should open only the port or ports you need. Your username and password should *not grant a higher level of permissions than needed*. This is quite important. I have worked with many enterprise class Coldfusion applications that used SQL server where the username for DSN's was "sa". Using "sa" exposes your entire database server to *any conceivable action* that can be taken on a SQL server - including shell commands. A malicious user gaining access to the database connection has access to your entire database server - right down to low level system processes.

Watch out for remote services that allow high levels of access. RDS is a service that is overlooked on a Coldfusion server - especially by novice administrators. An RDS user often has full access to the file system and can accomplish just about anything. The same is true of RDP access. Terminal services are useful and in conjunction with domain controls they may be appropriate, but they need to be carefully audited and planned out.

Remove *all unneeded services* from your server. Why are you running the print spooler? Is this a print server? Why are you running the MS indexing services? Is it a file server? Extra services slow your server down and provide additional targets of opportunity to both malicious and incidental code.

No More Default Configurations or Unneeded Services

It has been said that it is impossible to secure a windows server. People often think this (erroneously) because windows servers are so often *installed in the default configuration*. This configuration includes many unneeded services like those listed above. If you choose to allow a windows server to be installed with *all the things it can do available* you will end up with a server broadcasting it's many friendly (and sometimes insecure) resources to the surrounding network. Instead, install the server with no services except those you specifically need. Change any username or password that exists in the "default" to something else right away - and *document* your changes in a log book. A 3 ring binder with a section for each server on the rack is a good idea. Make a note each time you change a configuration.

One of the side benefits of having a server with a very narrow set of services is that you can ignore a large number of security patches and updates. For example, if you are installing a MS SQL Database server without IIS, FTP, Media Serivces, File Sharing, DNS etc. (in other words a windows kernel plus networking plus MS SQL), you will be able to ignore about 75 to 80 percent of the patches that are issued. Why worry about the media player 9 patch if you don't even have it installed? I can tell you from experience that a database server configured this way can stay running securly without a reboot (planned or otherwise) for months.

Don't settle for the default installation of the AP server either. **No application server will install in an appropriate (and secure) configuration right off the shelf.** All of them will require changes for the sake of both security and optimization. Coldfusion for example, will require disabling RDS, changing the service account (optional), setting up sandboxes, disabling unnecessary tags, possibly configuring multiple instances etc.

Domain and Service Security

What about the old question of security permissions for the application server itself? If you are running Coldfusion (for example) you have a choice of running the CF Service in 3 separate configurations.

local system account - this is the way that Coldfusion installs by default. The *local system account* has access to all the files and services of the local machine (the server on which CF is installed). It does not have access to any resources on the domain. For example, it cannot access external file resources via UNC paths. Why… because a username/password to those external resources would be required. So this level of access is restrictive in that it allows no access outside of the local machine. That's restrictive enough to minimize damage to the local machine in most cases (there are exceptions), but it does expose the application server itself.

Local User Account - You can run the CF server as a regular local user. This requires creating a user, setting up appropriate permissions and changing the login settings. Using this method you can effectively restrict CF to just a subset of resources that exist on this server only - largely through the use of file permissions (ACLs).

Domain User Account - Using a domain user account can be helpful if your server needs access to active directory resources that are on the domain but *not* on this server. This should be done with great care. When you choose a domain account you are opening your domain up to the Ap server. Make sure that you are using groups and users appropriately and are *explicitly authorizing individual resources*. In other words, don't just give up on getting it right and authorize whole file systems for access by your ap server. Don't laugh, getting domain permissions correct can be very challenging.

It should be noted at this point that getting the service permissions right for your application server is only a part of what you need to look at. Database connections are an important part of almost every application. It does you no good to implement a restrictive policy with your service accounts if your database connections are wide open to shell commands and DML. Other external access implemented in code may contain separate permissions in application code or config files (FTP calls for example). The very best advice I can give is to never ever treat an unknown application in a *cookie cutter* fashion when it comes to installation. Each application is unique and will come with it's own unique security and optimization challenges.

While you are at it, don't forget to secure the CF administrator. Don't allow Anonymous access to it. Put it on an "internal network" only site (one with no external access). This will mean that a user has to know both a local or domain account *and* the CF Admin password to have access to the administrator. They would also need access to your "inside the firewall" network.

What's Next

Finally we have come full circle. The final, most overlooked, and in many ways most important aspect of securing your application is to write secure code. In my final post in this series I will cover writing secure, maintainable code and protecting against common vulnerabilities like XSS, SQL injection and session high jacking. For now I will only say that even if you perfectly implement all of the previous "in-depth" defense mechanisms you will not be secure unless you also examine the code. Hackers and Crackers attack your firewall, hardware, domain security and servers in order to find targets of opportunity to attack your code. You can put in deadbolts and security alarms at your house too - but it does little good if you leave the front door open. Stay tuned.