Minefield

(a.k.a. Booby-Trap, Tripwire)

Abstract

The *Minefield* pattern will trick, detect, and block attackers during a break-in attempt. Attackers often know more than the developers about the security aspects of standard components. This pattern aggressively introduces variations that will counter this advantage and aid in detection of an attacker.

Problem

Any server that is accessible from the Internet will be attacked. No matter how much effort you expend in protecting it, the possibility exists that an attacker will penetrate the system. Once inside, an attacker who is familiar with the operation of a system will be at a huge advantage over one who has to perform lengthy, detectable reconnaissance activities.

Using standard Commercial-off-the-Shelf (COTS) software is an economic necessity. But attackers have access to the same COTS software, and thus are likely to know the operation of a system as well as its developers and administrator. Being intimately familiar with the internals of a system allows an attacker to operate with confidence, carefully avoiding standard detection mechanisms and covering his/her tracks afterwards. If a site is too cookie-cutter, attackers will even be able to use existing scripted tools to automate this process.

Solution

The *Minefield* pattern involves making slight customizations to a system. These modifications can have a number of different effects, depending on the attacker's skill and risk aversion:

- They might break compatibility with existing scripted attack tools.
- They might alert the operator to the presence of an intruder without the intruder's knowledge.
- They might cause the attacker to become uncomfortable enough with the prospect of being caught to cease attacking.

There is no cookie-cutter approach to developing minefields, as that would defeat the purpose. But there are a few basic approaches, including the following:

- Rename common, exclusively administrative commands on the server and replace them with instrumented versions that alert the administrator to an intruder before executing the requested command.
- Alter the file system structure.

- Introduce controlled variation using tools such as the Deception Toolkit.

- Add application-specific boobytraps that will recognize tampering with the site and prevent the application from starting.

Taken together, these techniques provide higher assurance that an attacker will not be able to attack the application server without being detected.

Issues

Customization

The single most significant security customization is to remove (or not install) the hundreds of tools, sample files, and potentially dangerous utilities that are part of any default installation, but not necessary for the server to execute. (See the *Build the Server From the Ground Up* pattern for more details.)

Simply moving files from the default locations can be enough to break many automated attack scripts. For example, many tools that probe for Web vulnerabilities are hard-coded to request files in the cgi-bin directory. Creating an alternative directory and placing cgi-bin files in that directory will cause many tools to fail. While something this simple will not deter a skilled adversary, it can cause unskilled attackers to leave a system alone.

Another useful technique is to change the disk location of the Web server's configuration and document files. If the apache configuration file is always located in /etc/httpd/config, a scripted attack could cause the file to be overwritten with a less secure version. Likewise, if the root Web page is stored at /var/www/html/index.html, it will be easy for a scripted attack to deface the site by replacing that page. Simply moving these files will thwart many tool-based attacks.

A more aggressive strategy is to use customization to make the server inhospitable to an attacker who actually gains access to the system. An excellent example of this described by Marcus Ranum is to rename all of the standard system utilities ("ls", "cd", etc.) to bizarre, non-standard names, and replace them with boobytrapped versions that will shut the system down. If legitimate administrators need access, they can load a secret alias file that maps all the names back to their standard value. An attacker who attempts to do anything on the system will cause it to be shut down almost immediately.

As noted in Security Assertion, it is also possible to develop application-specific tripwire-like functionality that shuts the system down if application data fails internal consistency checks.

Any customizations should be fully documented as part of the *Document the Server Configuration* pattern.

Detection

Custom boobytraps are most effective when carefully monitored for signs of an intrusion. Given enough time, a good attacker will learn to avoid the various mines and boobytraps. But initially,

the attacker will stumble across some of these devices. It is at this point that discovery of the attacker's presence is most possible.

In the example of the renamed cgi-bin directory, the system administrator should be aware of any attempts to access files in the original cgi-bin directory. Because no valid URL from the site will reference that directory, this is an obvious sign of mischief. Such activities should immediately result in an entry in a network address blacklist. If not blocked, they should be closely monitored for further evidence of attempted misuse.

In the example of the relocated apache configuration and html files, the original file locations should be carefully monitored. If these files change, the system administrator will know that a potentially devastating vulnerability exists on the Web server. At that point, monitoring should be increased in order to understand how these files were altered and to close the security vulnerabilities before they are exploited further.

Likewise, in the example of the boobytrapped system, if a boobytrap is triggered, the system administrator should perform a complete forensic analysis of the server in order to understand the extent of the damage.

In all of these cases, the existence of customizations allows for early discovery of security problems, but only if appropriate monitoring is present. A conventional intrusion detection system will not uncover these problems, unless it is customized appropriately.

Manageability

The biggest weakness of this approach is the adverse affect that it can have on system manageability. An unwary system administrator could easily set off the various boobytraps.

The extent of this problem will depend on the degree to which the server configuration is controlled. When the *Build the Server from the Ground Up* and *Use a Staging Server* patterns are properly employed, there should be no on-the-fly system administration performed on the production server. In this case, the customizations have no real affect on manageability. However, when the server must be actively administered, this pattern will be less appropriate.

Possible Attacks

Standard attack tools will cause a commercial intrusion detection systems to light up like a Christmas tree. When the system administrator has no recourse but to ignore these attacks, they represent a huge drain of personnel resources.

Distributed denial-of-service attacks against these systems can also limit the availability of the system.

Examples

There are products that implement some of the concepts in this pattern, such as Tripwire, Tripwire for Web Pages [2], and the Deception Toolkit [1].

Trade-Offs

Accountability	No direct effect.
Availability	If implemented in a fail-secure manner, this will adversely affect availability, by causing the system to halt when an intruder is detected.
Confidentiality	No direct effect.
Integrity	This pattern improves system integrity by increasing confidence that scripted attacks will fail and that human attackers will be detected before they can cause damage.
Manageability	This pattern will increase the management overhead of the application because it will require that log auditing be performed along with other security tasks. If the various sensors are too finely tuned, a significant administration burden will result. Customizations that are meant to trip up an intruder or break compatibility with attack tools can also trip up administrators and break compatibility with management and development tools.
Usability	Usability will suffer if sensors are overly sensitive, causing users to be denied access for too many legitimate mistakes.
Performance	No direct effect.
Cost	Development and documentation of custom configuration and monitoring mechanism is an additional one-time cost.

Related Patterns

- *Build the Server from the Ground Up* – a related pattern that provides more details about the trade-offs between developing custom components and using standard parts.

- *Network Address Blacklist* – a related pattern that discusses a mechanism for blocking network access to remote systems that have triggered numerous alerts.

References

[1] Cohen, F. and Associates. "The Deception Toolkit Home Page and Mailing List". http://all.net/dtk/dtk.html, 1998.

[2] Gillespie, G. "Saving face: Get Tripwire for Web Pages to protect your site against vandalism". http://www.computeruser.com/articles/daily/7,3,1,0620,01.html, June 2001.

[3] Ranum, M. “Intrusion Detection and Network Forensics”. USENIX '99, Monterey, CA, June 1999.

[4] U.S. Department of Energy Computer Incident Advisory Capability (CIAC). “J-042: Web Security”. http://ciac.llnl.gov/ciac/bulletins/j-042.shtml, May 1999.