Universidad de Los Andes
Departamento de Sistemas y Computación
ECOS – CSOF6202 Arquitectura de Software

Lista con los principales patrones de ataque tomados del libro: "Exploiting Software – How to break Code". Greg Hoglund and Gary McGraw. Addison-Wesley. 2004 y The Common Attack Pattern Enumeration and Classification (CAPEC) - http://capec.mitre.org/data/index.html

Attack Pattern: Make the Client Invisible

Remove the client from the communications loop by talking directly with the server. Explore to determine what the server will and will not accept as input. Masquerade as the client.

Attack Pattern: Target Programs That Write to Privileged OS Resources

Look for programs that write to the system directories or registry keys (such as HKLM which stores a number of critical Windows environment variables). These are typically run with elevated privileges and have usually not been designed with security in mind. Such programs are excellent exploit targets because they yield lots of power when they break.

Attack Pattern: Use a User-Supplied Configuration File to Run Commands That Elevate Privilege

A setuid utility program accepts command-line arguments. One of these arguments allows a user to supply the path to a configuration file. The configuration file allows shell commands to be inserted. Thus, when the utility starts up, it runs the given commands. One example found in the wild is the UUCP (or UNIX-to-UNIX copy program) set of utilities. The utility program may not have root access, but may belong to a group or user context that is more privileged than that of the attacker. In the case of UUCP, the elevation may lead to the dialer group, or the UUCP user account. Escalating privilege in steps will usually lead an attacker to a root compromise (the ultimate goal). Some programs will not allow a user-supplied configuration file, but the system wide configuration file may have weak permissions. The number of vulnerabilities that exist because of poorly configured permissions is large. A note of caution: As an attacker, you must consider the configuration file as an obvious detection point. A security process may monitor the target file. If you make changes to a configuration file to gain privilege, then you should immediately clean the file when you are finished. You can also run certain utilities to set back file access dates. The key is not to leave a forensic trail surrounding the file you exploited.

Attack Pattern: Make Use of Configuration File Search Paths

If you place a copy of the configuration file into a previously empty location, the target program may find your version first and forgo any further searching. Most programs are not aware of security, so no check will be made against the owner of the file. The UNIX environment variable for `PATH` will sometimes specify that a program should look in multiple directories for a given file. Check these directories to determine whether you can sneak a Trojan file into the target.

Attack Pattern: Direct Access to Executable Files

A privileged program is directly accessible. The program performs operations on behalf of the attacker that allow privilege escalation or shell access. For Web servers, this is often a fatal issue. If a server runs external executables provided by a user (or even simply named by a user), the user can cause the system to behave in unanticipated ways. This may be accomplished by passing in command- line options or by spinning an interactive session. A problem like this is almost always as bad as giving complete shell access to an

attacker. The most common targets for this kind of attack are Web servers. The attack is so easy that some attackers have been known to use Internet search engines to find potential targets. The Altavista search engine is a great resource for attackers looking for such targets. Google works too.

Attack Pattern: Embedding Scripts within Scripts

The technology that runs the Internet is diverse and complex. There are hundreds of development languages, compilers, and interpreters that can build and execute code. Every developer has a sense for only part of the overall technology. Investments in time and money are made into each particular technology. As these systems evolve, the need to maintain backward compatibility becomes paramount. In management speak, this is the need to capitalize on an existing software investment. This is one reason that some newer scripting languages have backward support for older scripting languages. As a result of this rapid and barely controlled evolution, much of the technology found in the wild can embed or otherwise access other languages and technologies in some form. This adds multiple layers of complexity and makes keeping track of all the disparate (yet available) functionality difficult at best. Filtering rules and security assumptions get swamped by the flow of new stuff. Looking for unanticipated functionality forgotten in the nooks and crannies of a system is an excellent technique.

Attack Pattern: Leverage Executable Code in Nonexecutable Files

Attackers usually need to upload or otherwise inject hostile code into a target processing environment. In some cases, this code does not have to be inside an executable binary. A resource file, for example, may be loaded into a target process space. This resource file may contain graphics or other data and may not have been intended to be executed at all. But, if the attacker can insert some additional code sections into the resource, the process that does the loading may be none the wiser and may just load the new version. An attack can then occur.

Attack Pattern: Argument Injection

User input is directly pasted into the argument of a shell command. A number of thirdparty programs allow passthrough to a shell with little or no filtering.

Attack Pattern: Command Delimiters

Using the semicolon or other off-nominal characters, multiple commands can be strung together. Unsuspecting target programs will execute all the commands.

Attack Pattern: Multiple Parsers and Double Escapes

A command injection will sometimes pass through several parsing layers. Because of this, meta- characters sometimes need to be "double escaped." If they are not properly escaped, then the wrong layer may consume them.

Attack Pattern: User-Supplied Variable Passed to File System Calls

File system calls are very common in software applications. In many cases, user input is consumed to specify filenames and other data. Without proper security control this leads to a classic vulnerability whereby an attacker can pass various parameters into file system calls.

Attack Pattern: Postfix NULL Terminator

In some cases, especially when a scripting language is used, the attack string is supposed to be postfixed with a NULL character. Using an alternate representation of `NULL` (i.e.,

%00) may result in a character translation occurring. If strings are allowed to contain NULL characters, or the translation does not automatically assume a null-terminated string,then the resulting string can have multiple embedded NULL characters. Depending on the parsing in the scripting language, NULL may remove postfixed data when an insertion is taking place.

Attack Pattern: Postfix, Null Terminate, and Backslash

If a string is passed through a filter of some kind, then a terminal NULL may not be valid. Using alternate representation of NULL allows an attacker to embed the NULL midstring while postfixing the proper data so that the filter is avoided. One example is a filter that looks for a trailing slash character. If a string insertion is possible, but the slash must exist, an alternate encoding of NULL in midstring may be used.

Attack Pattern: Relative Path Traversal

Usually the CWD for a process is set in a subdirectory. To get somewhere more interesting in the file system, you can supply a relative path that traverses out of the current directory and into other, more interesting subdirectories. This technique saves you from having to supply the fully qualified path (i.e., one that starts from the root). A nice feature of the relative path is that once you hit the root of the file system, additional moves into a parent directory are ignored. This means that if you want to make sure you start from the root of the file system, all you have to do is put a large number of "../" sequences into the injection.

Attack Pattern: Client-Controlled Environment Variables

The attacker supplies values *prior* to authentication that alter the target process environment variables. The key is that the environment variables are modified before any authentication code is used.

Attack Pattern: User-Supplied Global Variables (DEBUG=1, PHP Globals, and So Forth)

In seriously broken languages like PHP, a number of default configurations are poorly set. Trying these out is only prudent.

Attack Pattern: Session ID, Resource ID, and Blind Trust

When session and resource IDs are simple and available, attackers can use them to their advantage. Many schemes are so simple that pasting in another known ID in a message stream
works.

Attack Pattern: Analog In-band Switching Signals (aka "Blue Boxing")

Many people have heard of 2600, the frequency used in the United States to control telephone switches during the 1960s and 1970s. (Come to think of it, probably more people have heard of the hacker 'zine *2600* and its associated club than have heard of the reason for the name of the club.) Most systems are no longer vulnerable to ancient phreaking attacks. However, older systems are still found internationally. Overseas trunk lines that use trans-Atlantic cabling are prone to the in-band signal problem and they aretoo expensive a resource to abandon. Thus, many overseas (home-country direct) 800/888 numbers are known to have in-band signal problems even today. Consider the CCITT-5 (C5) signaling system that is used internationally. This system does not use the commonly known 2,600 Hz, but instead uses 2,400 Hz as a control signal. If you have ever heard the "pleeps" and chirps on the Pink Floyd album "The Wall,"then you have heard C5 signals. There are millions of phone lines still in operation today that are routed through switches with in-band signaling. This attack pattern involves playing specific control commands across a normal voice link, thus seizing control of the line, rerouting calls, and

so on.

Attack Pattern: Simple Script Injection

As a normal user of a system there are opportunities to supply input to the system. This input may include text, numbers, cookies, parameters, and so forth. Once these values are accepted by the system, they may be stored and used later. If the data are used in a server response (such as a message board, where the data are stored and then displayed back to users), an attacker can "pollute" these data with code that will be interpreted by unsuspecting client terminals.

Attack Pattern: Embedding Script in Nonscript Elements

Script does not need to be inserted between `<script>` tags. Instead, script can appear as part of another HTML tag, such as the image tag. The injection vector is

```
<img src=javascript:alert(document.domain)>
```

Attack Pattern: XSS in HTTP Headers

The HTTP headers of a request are always available to a server for consumption. No matter the context or where data are positioned, if the data are from the client, they should clearly be untrusted. However, in many cases programmers overlook header information. For some reason header information is treated as holy ground that cannot be controlled by the user. This pattern takes advantage of this oversight to inject data via a header field.

Attack Pattern: HTTP Query Strings

A query string takes variable = value pairs. These are passed to the target executable or script designated in the request. A variable can be injected with script. The script is processed and stored in a way that is later visible to a user.

Attack Pattern: User-Controlled Filename

An unfiltered, user-controlled filename can be used to construct client HTML. Perhaps HTML text is being built from filenames. This can be the case if a Web server is exposing a directory on the file system, for example. If the server does not filter certain characters, the filename itself can include an XSS attack.

Attack Pattern: Passing Local Filenames to Functions That Expect a URL

Use local filenames with functions that expect to consume a URL. Find interesting connections.

Attack Pattern: Meta-characters in E-mail Header

Meta-characters can be supplied in an e-mail header and may be consumed by the client software to interesting effect.

Attack Pattern: Client-side Injection, Buffer Overflow

Acquire information about the kind of client attaching to your hostile service. Intentionally feed malicious data to the client to exploit it. Possibly install backdoors.

Attack Pattern: Cause Web Server Misclassification

A very famous set of classification problems occurs when a Web server examines the last few characters of a filename to determine what kind of file it is. There are many ways to take advantage of these kinds of problems—appending certain strings to file names, adding dots, and so forth.

Attack Pattern: Alternate Encoding the Leading Ghost Characters

Some APIs will strip certain leading characters from a string of parameters. Perhaps these characters are considered redundant, and for this reason they are removed. Another possibility is the parser logic at the beginning of analysis is specialized in some way that causes some characters to be removed. The attacker can specify multiple types of alternative encodings at the beginning of a string as a set of probes. One commonly used possibility involves adding ghost characters—extra characters that don't affect the validity of the request at the API layer. If the attacker has access to the API libraries being targeted, certain attack ideas can be tested directly in advance. Once alternative ghost encodings emerge through testing, the attacker can move from lab-based API testing to testing real-world service implementations.

Attack Pattern: Using Slashes in Alternate Encoding

Slash characters provide a particularly interesting case. Directory-driven systems, such as file systems and databases, typically use the slash character to indicate traversal between directories or other container components. For murky historical reasons, PCs (and, as a result, Microsoft OSs) choose to use a backslash, whereas the UNIX world typically makes use of the forward slash. The schizophrenic result is that many MS-based systems are required to understand both forms of the slash. This gives the attacker many opportunities to discover and abuse a number of common filtering problems. The goal of this pattern is to discover server software that only applies filters to one version, but not the other.

Attack Pattern: Using Slashes in Alternate Encoding

Slash characters provide a particularly interesting case. Directory-driven systems, such as file systems and databases, typically use the slash character to indicate traversal between directories or other container components. For murky historical reasons, PCs (and, as a result, Microsoft OSs) choose to use a backslash, whereas the UNIX world typically makes use of the forward slash. The schizophrenic result is that many MS-based systems are required to understand both forms of the slash. This gives the attacker many opportunities to discover and abuse a number of common filtering problems. The goal of this pattern is to discover server software that only applies filters to one version, but not the other.

Attack Pattern: Unicode Encoding

Unicode is a system for encoding character strings in a 16-bit representation so that characters from a number of different languages can be represented. Unicode involves using 2 bytes for every character instead of the customary single byte found in ASCII encoding. Any system that is unicode aware may be capable of converting unicode strings into ASCII byte strings. If the native language of the system or the APIs that are being used require normal byte strings, then the system may provide for a translation from unicode. The advantage to an attacker begins when some of the components of the system are not unicode aware. In this case, the attacker may provide a unicode string in the hopes that a filtering mechanism or classifying mechanism will fail to understand the request. This can result in slipping past a content filter and/or possibly causing the application to route a request incorrectly.

Attack Pattern: URL Encoding

In many cases, a character can be encoded as `%HEX-CODE` in URL strings. This has led to a number of classic filtering problems.

Attack Pattern: Alternative IP Addresses

IP address ranges can be represented using alternative methods. Here are some examples

Attack Pattern: Overflow Variables and Tags

In this case, the target is a program that reads formatted configuration data and parses a tag or variable into an unchecked buffer. The attacker crafts a malicious HTML page or configuration file that includes oversized strings, thus causing an overflow.

Attack Pattern: HTTP Cookies

Because HTTP is a stateless protocol, cookies (small files that are stored in a client browser) were invented, mostly to preserve state. Poor design of cookie handling systems leaves both clients and HTTP daemons susceptible to buffer overflow attack.

Attack Pattern: Filter Failure through Buffer Overflow

In this attack, the idea is to cause an active filter to fail by causing an oversized transaction. If the filter fails "open" you win.

Attack Pattern: Buffer Overflow with Environment Variables

Programs consume a huge number of environment variables, but they often do so in unsafe ways. This attack pattern involves determining whether a particular environment variable can be used to cause the program to misbehave.

Attack Pattern: Buffer Overflow in an API Call

Libraries or shared code modules can suffer from buffer overflows too. All clients that make use of the code library thus become vulnerable by association. This has a very broad effect on security across a system, usually affecting more than one software process.

Attack Pattern: String Format Overflow in `syslog()`

The `syslog` function is typically misused, and user-supplied data are passed as a format string. This is a common problem, and many public vulnerabilities and associated exploits have been posted.