Web Application Security

Rajendra Kachhwaha
rajendra1983@gmail.com

November 2, 2015

Outline

Extended Stored Procedures:

1. Extended stored procedures are essentially compiled Dynamic Link Libraries (DLLs) thatuse a SQL Server specific calling convention to run exported functions.
2. They allow SQL Server applications to have access to the full power of C/C++, and are an extremely useful feature.
3. A number of extended stored procedures are built in to SQL Server, and perform various functions such as sending email and interacting with the registry.

Further Access:

Once an attacker has control of the database, they are likely to want to use that access to obtain further control over the network. This can be achieved in a number of ways:

1. Using the xp_cmdshell extended stored procedure to run commands as the SQLserver user, on the database server.
2. Using the xp_regread extended stored procedure to read registry keys, potentially including the SAM (if SQL Server is running as the local system account).
3. Use other extended stored procedures to influence the server.

Further Access:

4. Run queries on linked servers.
5. Creating custom extended stored procedures to run exploit code from within the SQL Server process.
6. Using the sp_OACreate, sp_OAMethod and sp_OAGetProperty system stored procedures to create Ole Automation (ActiveX) applications that can do everything an ASP script can do.

Examples:

1. exec master..xp_cmdshell 'dir' : will obtain a directory listing of the current working directory of the SQL Server process.
2. exec master..xp_cmdshell 'net1 user' : will provide a list of all users on the machine.
3. exec master..xp_availablemedia : reveals the available drives on the machine.
4. exec master..xp_loginconfig : reveals information about the security mode of the server.

Examples:

1. exec master..xp_cmdshell 'dir' : will obtain a directory listing of the current working directory of the SQL Server process.
2. exec master..xp_cmdshell 'net1 user' : will provide a list of all users on the machine.
3. exec master..xp_availablemedia : reveals the available drives on the machine.
4. exec master..xp_loginconfig : reveals information about the security mode of the server.
5. You can google on how to use xp_regread extended stored procedure and how to create Ole Automation (ActiveX) applications.

Definition:

1. In computer science, canonicalization (sometimes standardization or normalization) is a process for converting data that has more than one possible representation into a "standard", "normal", or canonical form.
2. This can be done to compare different representations for equivalence, to count the number of distinct data structures, to improve the efficiency of various algorithms by eliminating repeated calculations, or to make it possible to impose a meaningful sorting order.
3. Canonicalization of file names is important for computer security.

Definition:

4. For example, a web server may have a security rule stating "only execute files under the cgi directory" (`C:\inetpub\wwwroot\cgi-bin`).
5. The rule is enforced by checking that the path starts with `C:\inetpub\wwwroot\cgi-bin`, and if it does, the file is executed.
6. Should file `C:\inetpub\wwwroot\cgi-bin\..\..\..\Windows\System32\cmd.exe` be executed ?

Definition:

4. For example, a web server may have a security rule stating "only execute files under the cgi directory" `C:\inetpub\wwwroot\cgi-bin`).
5. The rule is enforced by checking that the path starts with `C:\inetpub\wwwroot\cgi-bin`, and if it does, the file is executed.
6. Should file `C:\inetpub\wwwroot\cgi-bin\..\..\..\Windows\System32\cmd.exe` be executed ?
7. No, because this trick path goes back up the directory hierarchy (through use of the '..' path specifier), not staying within cgi-bin. This type of fault is called a directory traversal vulnerability.

Directory Traversal Vulnerability:

1. A directory traversal (or path traversal) consists in exploiting insufficient security validation of user-supplied input file names, so that characters representing "traverse to parent directory" are passed through to the file APIs.
2. The goal of this attack is to order an application to access a computer file that is not intended to be accessible.
3. This attack exploits a lack of security (the software is acting exactly as it is supposed to) as opposed to exploiting a bug in the code.
4. Directory traversal is also known as the `../` (dot dot slash) attack, directory climbing, and backtracking.

Directory Traversal Vulnerability:Example

```
<?php
$template = 'red.php';
if (isset($_COOKIE['TEMPLATE']))
   $template = $_COOKIE['TEMPLATE'];
include ("/home/users/phpguru/templates/" . $template);
?>
```

An attack against this system could be to send the following HTTP request:

```
GET /vulnerable.php HTTP/1.0
Cookie: TEMPLATE=../../../../../../../../../../etc/passwd
```

Generating a server response such as:

```
HTTP/1.0 200 OK
Content-Type: text/html
Server: Apache

root:fi3sED95ibqR6:0:1:System Operator:/:/bin/ksh
daemon:*:1:1::/tmp:
phpguru:f8fk3j1OIf31.:182:100:Developer:/home/users/phpguru/:/bin/csh
```

The repeated `../` characters after `/home/users/phpguru/templates/` has caused include() to traverse to the root directory, and then include the Unix password file `/etc/passwd`.

Variations of directory traversal:

1. **Directory traversal on Unix:** Common Unix-like directory traversal uses the `../` characters.
2. **Directory traversal on Microsoft Windows:** Microsoft Windows or DOS directory traversal uses the `..\` characters. Today, many Windows programs or APIs also accept Unix-like directory traversal characters.
3. **URI encoded directory traversal:**
   - `%2e%2e%2f` which translates to `../`
   - `%2e%2e/` which translates to `../`
   - `..%2f` which translates to `../`
   - `%2e%2e%5c` which translates to `..\`

Possible methods to prevent directory traversal:

1. Process URI requests that do not result in a file request.
2. When a URI request for a file/directory is to be made, build a full path to the file/directory if it exists, and normalize all characters.
3. It is assumed that a 'Document Root' fully qualified, normalized, path is known, and this string has a length N. Assume that no files outside this directory can be served.
4. Ensure that the first N characters of the fully qualified path to the requested file is exactly the same as the 'Document Root'. If so, allow the file to be returned. If not, return an error, since the request is clearly out of bounds from what the web-server should be allowed to serve.