**Null Byte Injection**

[Page history](http://projects.webappsec.org/w/page-revisions/13246949/Null%20Byte%20Injection)last edited by [Robert Auger](javascript:alert('Please%20join%20this%20workspace%20to%20see%20more%20details%20about%20this%20user.');) 10 years, 9 months ago

**Project:**[**WASC Threat Classification**](http://projects.webappsec.org/Threat-Classification)

***Threat Type:***[***Attack***](http://projects.webappsec.org/Threat-Classification-Glossary)

***Reference ID: WASC-28***

**Null Byte Injection**

Null Byte Injection is an active exploitation technique used to bypass sanity checking filters in web infrastructure by adding URL-encoded null byte characters (i.e. %00, or 0x00 in hex) to the user-supplied data. This injection process can alter the intended logic of the application and allow malicious adversary to get unauthorized access to the system files.

Most web applications today are developed using higher-level languages such as, PHP, ASP, Perl, and Java. However, these web applications at some point require processing of high-level code at system level and this process is usually accomplished by using ‘C/C++’ functions. The diverse nature of these dependent technologies has resulted in an attack class called ‘Null Byte Injection’ or ‘Null Byte Poisoning’ attack. In C/C++, a null byte represents the string termination point or delimiter character which means to stop processing the string immediately. Bytes following the delimiter will be ignored. If the string loses its null character, the length of a string becomes unknown until memory pointer happens to meet next zero byte. This unintended ramification can cause unusual behavior and introduce vulnerabilities within the system or application scope. In similar terms, several higher-level languages treat the ‘null byte’ as a placeholder for the string length as it has no special meaning in their context. Due to this difference in interpretation, null bytes can easily be injected to manipulate the application behavior.

URLs are limited to a set of US-ASCII characters ranging from 0x20 to 0x7E (hex) or 32 to 126 (decimal)[5][8]. However, the aforementioned range uses several characters that are not permitted because they have special meaning within HTTP protocol context. For this reason, the URL encoding scheme was introduced to include special characters within URL using the extended ASCII character representation. In terms of “null byte”, this is represented as %00 in hexadecimal. The scope of a null byte attack starts where web applications interact with active ‘C’ routines and external APIs from the underlying OS. Thus, allowing an attacker to manipulate web resources by reading or writing files based on the application's user privileges.

Let’s take some examples to demonstrate a real-world attack:

**Example#1 Perl**

Perl is written on the top of ‘C’ and ‘C’ language handles the null byte as a string terminator, while Perl itself does not. If the Perl script is processing user-supplied malicious data (i.e. %00 embedded), it will be passed to the system call function “open FILE ( )” and furthermore passed onto the ‘C’ engine for final processing. This allows the underlying ‘C’ processor to reject anything beyond the “%00” null byte character. As in the case mentioned below, the user supplied filename will be filtered against basic acceptable characters set and then passed on to be read from the user(s) directory with a pre-defined extension (JPG). From here an attacker can manipulate this request to execute or read a system file (e.g. /etc/passwd) by embedding a ‘null byte %00’ with a valid extension to fool the code into processing the request successfully.

Code Snippet:

$buffer = $ENV{'QUERY\_STRING'};

$buffer =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

$fn = '/home/userx/data/' .$buffer. '.jpg';

open (FILE,"<$fn");

Exploitation:

Normal Mode: <http://www.example.host/read.pl?page=userphoto.jpg>

Attacking Mode: <http://www.example.host/read.pl?page=../../../../etc/passwd%00jpg>

**Example#2 PHP**

The scenario mentioned above is also true with PHP technology. For instance, if a user requests a personal data file from the server, it will be validated by appending ‘.DAT’ extension to the filename. This script itself appears to be secure by enforcing the file extension but the request for the resource can be manipulated by appending a “%00” null byte at the end of URL. Thus, a malicious adversary can take advantage of this vulnerability to read almost any system file through a simple browser.

Code Snippet:

$file = $\_GET['file'];

require\_once("/var/www/images/$file.dat");

Exploitation:

Normal Mode: <http://www.example.host/user.php?file=myprofile.dat>

Attacking Mode: <http://www.example.host/user.php?file=../../../etc/passwd%00>

**Example#3 Java**

The trend of null byte injection attack is also common in Java. For instance, by examining the details of a vulnerable function "File ( )" inside "java.io.File" which passes its argument to the underlying 'C' API to process the user request failed to determine the actual file extension because it treats the occurrence of first null byte as a string terminator. Let us assume the following example in which a user is requesting access to the specific file where the extension is enforced as “.db” by the developer for validation purposes. The same request can be simulated by the attacker but in a different way to access the system resource by embedding a “%00” null byte with a valid filename and extension.

Code Snippet:

String fn = request.getParameter("fn");

if (fn.endsWith(".db"))

{

File f = new File(fn);

//read the contents of “f” file

…

}

Exploitation:

Normal Mode: <http://www.example.host/mypage.jsp?fn=report.db>

Attacking Mode: <http://www.example.host/mypage.jsp?fn=serverlogs.txt%00.db>

**References**

"Prevent PHP NULL byte or upload file security hole", Nitin Gupta (2009)

[1] <http://www.fruitnotes.com/blogs/Prevent_php_NULL_byte_or_upload_file_security_hole_1762>

"Null byte attacks are alive and well", Portswigger (2008)

[2] <http://blog.portswigger.net/2008/05/null-byte-attacks-are-alive-and-well.html>

"CGI Security and the null byte problem", Ovid (2000)

[3] <http://www.perlmonks.org/index.pl?node_id=38548>

"Test cases for null-byte injections in Java", Arshan Dabirsiaghi

[4] <http://i8jesus.com/stuff/Test.java>

[5] "The Web Application Hackers Handbook", Dafydd Stuttard, Marcus Pinto (2008)

[6] "Apache Security", Ivan Ristic (2005)

[7] "The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities", Mark Dowd, John McDonald, Justin Schuh (2006)

Request for Comments: 2396 - "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter (1998)

[8] <http://www.ietf.org/rfc/rfc2396.txt>

CAPEC-52: Embedding NULL Bytes

[9] <http://capec.mitre.org/data/definitions/52.html>

Perl CGI problems: Phrack Magazine Vol.9 Issue-55

[10] <http://www.phrack.com/issues.html?issue=55&id=7#article>