Taseer *et al. (IJECI) 2022*

(IJECI)
ISSN: 2522-3429 (Print)
ISSN: 2616-6003 (Online)

International Journal for
Electronic Crime Investigation

DOI: https://doi.org/10.54692/ijeci.2022.0604121

Research Article

Vol. 6 issue 4 Year 2022

Cyber Security Incident Response and Reverse Engineering

Taseer Suleman and Nadia Liaquat

School of Electrical Engineering and Computer Sciences, NUST, Islamabad, Pakistan

Corresponding authores: nadialiaquat001@gmail.com, 12msccsmsuleman@seecs.edu.pk

Received: 11 September, 2022; Accepted: 14 November, 2022; Published: 20 December, 2022

Abstract

Although the incident response has always been a crucial component of information security, security administrators frequently ignore it. Whereas, Reverse engineering focuses on the difficult issue of analyzing legacy software code in the absence of appropriate documentation. This paper proposes an approach to understanding cyber security Incident Response and the services it provides followed by Reverse Engineering resources and the practical analysis of a malware named "Alice ATM Malware" in detail.

Key words: Cybersecurity Incident, Indicator of compromise, Digital defenses, Computer Security Incident Response Team, IDA de-compilers, Reverse Engineering.

1. Introduction

Highly skilled attackers provide a continuing threat to organizations. Attackers constantly discover new ways to breach organizations' digital defenses to steal information or destroy their operations. The threat landscape is rapidly changing (Ahmad et al. 2019). There is a lot of ongoing research on how to reinforce these digital defenses, but relatively little is done to improve the process that takes over when things go wrong: incident response (IR) [1].

When digital defenses fail, cybersecurity incident response teams are at the forefront and must intervene to reinstitute services andical problems are impeding the incident response team's capacity to respond to cyber-attacks (Nyre-Yu et al. 2019) [2]. By removing these obstacles, organizations may be better able to respond to incidents in general. One method to achieve this is through training, particularly employing training scenarios, which can boost team cybersecurity performance by developing skills and spotting possible flaws. The most significant information technology advancements are right in front of us right now. Given that approximately one billion computers in our world are currently connected to the Internet and that mobile telephony services and e-commerce have converged, enormous amounts of information can go from one network to another with only

one request, command, or click (Global Reach, 2005). To this degree, numerous technologies, platforms, and infrastructures are flourishing to give services to the end user, who now serves as the target point because it is the user who asks for services, accesses networks, and resources, and needs security and privacy [3].

Expertise in fields like forensic investigation and malware reverse engineering is frequently needed for incident response. Reverse engineering is the process through which a variety of items, including software, types of machinery, and architectural structures, are disassembled to obtain design data and is also known as back engineering. The reverse engineering method typically involves disassembling the parts of larger, more important products. The reader will learn about reverse engineering's common principles, applications, stages, and future in this article. It demonstrates how Reverse Engineering is constantly developing and influencing the idea of cyber security [4].

Software development can also benefit from reverse engineering since it allows developers to examine their code and identify potential flaws that were overlooked during software development but that an adversary could find through reverse engineering. In Cybersecurity it is important because it enables the extraction of Indicators of Compromise (IoC) from samples [5]. Typically IoC is file’s hashes, registry keys, import function and export functions, the programming language used, compilation date, IPs, emails, and even text strings that are present in the code, which are the traces left by attackers [6] [7].

2. Incident Response

The handling of diverse security incidents, cyber threats, and data breaches involves an organized technique called incident response. A cyber attack or live incident's cost is to be identified, contained, and reduced using incident response techniques [8]. Although it is not the end-all solution, a solid incident response (IR) plan can seal a potential weakness to avoid more attacks. The response is a part of incident handling, which in turn looks at the coordination, logistics, and planning required to deal with a problem. This kind of work is normally handled by the Computer Security Incident Response Team (CSIRT), with assistance from the Security Operation Center. While incident management is the primary function of CSIRT, it also has reporting, analysis, and reaction responsibilities. Before these phases, the incident must be located and promptly reported [9]. The function of a SOC Analyst becomes crucial at this point.

2.1 Incident Response Services

The most efficient incident response is carried out rapidly by trained responders. Organizations frequently lack the funding necessary to keep a fully functional incident response team on duty around the clock [10]. Working with an outside organization that provides qualified incident response services is one alternative.

Getting involved with these organizations offers the following advantages:

2.1.1 Availability

The cost and impact of an attack on the organization are reduced the faster the incident response team gets to work. Cybersecurity issues can happen at any time, and getting in touch with incident response team members

after hours might be challenging. To improve coverage and availability, professional incident response companies have numerous staff teams on hand.

2.1.2 Experience

Managing security issues improperly can increase costs and harm a company. For instance, a ransomware attack might cause a system to become unstable, making it unlikely that encrypted data will be restored after a system restart. Professional incident responders have the experience necessary to accurately and efficiently address such security issues.

2.1.3 Specialized expertise

Reverse engineering malware and forensic analysis are two skills that are frequently used in incident response. Even though the majority of businesses won't require these talents in-house, a specialized incident response team has access to the professionals they require to successfully manage cybersecurity problems.

2.1.4 Controlling all aspects of the incident response procedure

All of the organization's incident response requirements should be met by outsourced incident response providers. This includes putting incident response plans in place, controlling identified intrusions, and thwarting potential attacks.

3. Incident Response Plan Phases

It is a set of guidelines that must be followed during each stage of incident response. The components of a good incident response plan include a clear communication strategy, directives defining the duties and responsibilities of each person and organization, and protocols that must be followed at all times [11].

3.1 Preparation

The steps a business should take in the case of a disruptive incident are outlined in an effective incident response strategy. The plan starts by describing how a company should reduce the danger of a data leak [12]. Organizational data protection policies should be in line with security objectives and technology defenses throughout the preparation stage. You must, at the very least, guarantee that staff members have received training on information security. They should ideally also have specialized training in incident response. To make sure your sensitive data is adequately protected, you should audit your systems as well.

3.2 Identification

The second component of incident response planning deals with the measures an organization takes to ascertain when one of its systems has been compromised [13]. You are better able to stop you may quickly recognize an incursion from an assault. You can save time and money even if it isn't possible by limiting the damage and hastening the response effort.

The following inquiries should be addressed when determining a security incident:

- Who found the opening?
- How much of a breach is there?
- Had an impact on our operations?
- Where did the compromise originate?

3.3 Containment

The third phase discusses the steps you should

take to minimize harm after being infiltrated. Depending on the circumstances, this can require taking action to remove the criminal hacker from your networks or to isolate the already compromised data [14].

During this stage, you should consider whether systems need to be shut down or removed as well as whether there are any rapid fixes for vulnerabilities.

3.4 Eradication

Fixing the fault that caused the data breach is the aim of phase four of a cyber incident response strategy. Again, the specifics will depend on the type of incident, but right now you need to determine how the information was disclosed and how to get rid of the threat [15].

For instance, you would remove the malicious software and isolate the affected areas if your firm had been compromised by malware In the meantime, you would lock down a worker's account if the attack resulted from their login information being stolen by a malicious hacker.

3.5 Recovery

Getting your systems back online is the penultimate step in responding to a cyber incident once the threat has been eradicated. In some circumstances, this will be trickier to do than in others, but it's an important step that needs to be taken seriously. Without a strong recovery process, you might still be vulnerable to attacks, which would make the injury worse. As part of the recovery process, you should test and monitor the impacted systems after the issue has been fixed. This guarantees that the measures you implement the function as planned and offers you the chance to fix any errors [16].

3.6 Lessons Learned

The final phase in the cyber incident response strategy is reviewing the occurrence and identifying potential areas for improvement. Your incident response team needs to meet to go over the parts of the plan that worked and any problems you encountered.

It is important to review the process at every stage. Discuss what happened, why it happened, what you tried to stop it, and what may have been done differently. For instance, was the documentation effective and clear, and did the plan include any gaps?

Before having this conversation, one to two weeks should have passed after the security incident; this will allow everyone to think about the incident in retrospect while still keeping it fresh in everyone's memory [17].

Instead of berating team members for prior mistakes, this stage's objective is to avoid inefficiencies from occurring again. Failures in the processed signal that the documentation was either unclear, the proper course of action wasn't specified, or staff training wasn't adequate.

4 Reverse engineering

The procedure of obtaining knowledge or designing blueprints from everything created by humans is known as reverse engineering. The idea has likely existed since the Industrial

Revolution, long before computers or other contemporary technology. It closely resembles scientific research, in which an investigator seeks to identify the "blueprint" of an atom or the human mind. Reverse engineering is different from a traditional scientific study in that the artifact under investigation is human-made, as opposed to a natural event in scientific research [18]. When such information is lacking, reverse engineering is typically used to fill in the gaps in knowledge, ideas, and design philosophy.

4.1 Resources Used in Reverse Engineering

Multiple tools are used to perform reverse engineering. These tools can help to debug, decompile and disassemble the application.

4.1.1 Debuggers

GDB is a debugger for programming that may also be used to decipher binary code. While the assembly code is running, you can view the information in memory and registers [19]. Additionally, breakpoints can be added anywhere in the application using debuggers.

4.1.2 Disassembler

Machine code is converted into a human-readable format using a disassembler. Because disassembled code lacks programmers' comments and annotations, reading it is more challenging than reading source code.

4.1.3 De compilers

Although IDA is difficult to use and needs extensive programming knowledge, its technical level accurately captures the fundamental nature of reverse engineering (Seo et al., 2019). DE-compilation is the process of translating a compiled program into a higher-level symbolic language that humans can comprehend, and it particularly makes use of reverse engineering methods.

4.2 Stages in Reverse engineering

By creating models that describe the current program and the assumed goal, reverse engineering can be accomplished. Three main phases make up this process:

- Recovering from implementation. Prepare a preliminary model while learning about the application quickly.
- Adaptive design. Foreign key references should be resolved and the database's mechanics reversed.
- Retrieval of analysis. Eliminate any inaccuracies in the model and design artifacts.

4.2.1 Implementation Recovery

You prepare an early model for reverse engineering during implementation recovery. The first model should only reflect the implementation and contain no inferences because it will be used as a reference.

Reading through the most recent documentation and getting acquainted with a program is the first step. The resulting context makes it simpler to interact with application professionals and clarifies the developer's intention. This project ought to be finished in a few hours. Even though what you learn is unrelated to the actual reverse engineering, it is crucial since it enables you to make more

accurate observations as you go.

The database structure is then typed in manually or automatically into a modeling program. Some tools have can read an RDBMS's system tables and seed a model. If you utilize these tools, you ought to at the very least glance over the database architecture to get a sense of the development approach.

4.2.2 Design Recovery

You undo the database's mechanics during design recovery and just carry out simple operations. Conjecture and interpretation should wait until the analysis-recovery stage. In most cases, design recovery may be carried out independently, without assistance from application expertise. You fix three main problems at this stage.

Identity: For the prospective entity type keys, unique indexes will typically be defined. Otherwise, search for uncommon data combinations, which can point to a candidate key but not confirm it. You can also guess potential keys by looking at names and styling conventions. A suspected foreign key may imply a comparable candidate key.

Foreign keys: The most difficult aspect of design recovery is often identifying foreign keys or references between different tables. Foreign keys may be indicated by names and data types that match. Foreign keys and their referents can be declared by developers in some DBMSs, such as RDBMSs, but the majority of legacy applications do not use this feature.

Queries: When they are available, queries can be used to improve your understanding of identity and foreign keys.

Design reclamation may result in optimizations and flaws, but it still reflects the DBMS paradigm in its ultimate form. The model will rarely be finished in practice. Some of the structure's components might be unclear.

4.2.3 Analysis Recovery

The last step is analysis and recovery. The model is interpreted, improved, and made more abstract. During this stage is when you should speak with any available application professionals. Recovery from analysis consists of four basic steps.

Clarification: Eliminate any existing design artifacts. For instance, file and database access keys do not include any essential information and are purely design options, thus they are not required to be included in an analytical model.

Redundancy: Remove derived data type if it improves database design or if it was included for the wrong reasons. You might need to examine the data to determine whether a data structure is a duplicate.

Errors: Resolve any leftover database problems. This phase of analysis recovery is necessary since you need to completely understand the database before you can say that the developer committed a mistake. An apparent error in the early stages might have been a fair procedure or the result of inadequate database understanding [20].

Integration of a model. Various information sources can result in various models. For instance, structure and data analysis is typically used to develop a reverse-engineered

model. A forward-engineered model could be created using a user manual. The final analysis model must incorporate all independent models.

5. Practical Demonstration

The following shown is the 32-bit executable file. The Alice ATM malware was discovered for the first time in November 2016 as part of an ATM malware research study with Europol EC3, however, researchers believe it has been present since 2014.

property	value
md5	F1478AA747A976FB2AD526FA71ECA853
sha1	4292DF415C11F4155E8910EBCDE8BD2DA24E4426
sha256	04F25013EB088D5E8A6E55BDB005C464123E6605897BD80AC245CE7CA12A7A70
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z .. ..
file-size	18432 bytes
entropy	3.724
imphash	43CB8BEA4CA8C4F791841893ADD4E86A
signature	n/a
tooling	Visual Studio 5.0 - 5.12
entry-point	53 56 57 33 FF 57 E8 A4 00 00 00 A3 17 10 40 00 6A 03 E8 EF F3 FF FF 57 68 E9 1A 40 00 57 68 D0 07
file-version	1.0.0.0
description	Project Alice
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x5431DF9B (Mon Oct 06 00:17:31 2014 | UTC)
debugger-stamp	n/a
resources-stamp	0x00000000 (Thu Jan 01 00:00:00 1970 | UTC)
import-stamp	0x00000000 (Thu Jan 01 00:00:00 1970 | UTC)
exports-stamp	n/a

Figure 1: Executable file

This malware is detected on Virus total and 54 security vendors and 1 sandbox flagged this file as malicious on Virus Total.

Figure 2: Results of Virus Total

These are the libraries that are used by this malware.

library (5)	blacklist (1)	type (1)	functions (36)	description
ntdll.dll	-	implicit	6	NT Layer DLL
user32.dll	-	implicit	12	Multi-User Windows USER API Client DLL
kernel32.dll	-	implicit	7	Windows NT BASE API Client DLL
comctl32.dll	-	implicit	1	Common Controls Library
msxfs.dll	x	implicit	10	Extension for Financial Services (XFS)

Figure 3: Libraries used by the malware

Ntdll.dll:

A module containing NT system functionality is called ntdll.dll. The NT kernel functions are contained in the Microsoft-created file ntdll.dll, which is referred to as an "NT Layer DLL."

functions (36)	blacklist (3)	ordinal (0)	library (5)
RtlCaptureStackBackTrace	x	-	ntdll.dll
RtlMoveMemory	x	-	ntdll.dll
VerSetConditionMask	x	-	ntdll.dll
RtlUnwind		-	ntdll.dll
RtlZeroMemory		-	ntdll.dll
RtlFillMemory		-	ntdll.dll

Figure 4: Functions that are used from this library.

Comctl.dll:

A module called Comctl32.dll houses standard GUI elements used by Windows programs.

InitCommonControls		-	comctl32.dll

Msxfs.dll:

Microsoft didn’t provide much information on msxfs.dll.

WFSCleanUp		-	msxfs.dll
WFSOpen		-	msxfs.dll
WFSGetInfo		-	msxfs.dll
WFSExecute		-	msxfs.dll
WFSLock		-	msxfs.dll
WFSRegister		-	msxfs.dll
WFSFreeResult		-	msxfs.dll
WFSUnlock		-	msxfs.dll
WFSClose		-	msxfs.dll
WFSStartUp		-	msxfs.dll

Malicious Strings:

encoding (2)	size (bytes)	location	blacklist (3)	hint (31)	value (190)
ascii	24	0x000013C2	x	function	RtlCaptureStackBackTrace
ascii	13	0x000013DE	-	function	RtlFillMemory
ascii	13	0x000013EE	x	function	RtlMoveMemory
ascii	9	0x000013FE	-	function	RtlUnwind
ascii	13	0x0000140A	-	function	RtlZeroMemory
ascii	19	0x0000141A	x	function	VerSetConditionMask
ascii	13	0x00001446	-	function	AnimateWindow
ascii	9	0x00001468	-	function	EndDialog
ascii	10	0x00001474	-	function	GetDlgItem
ascii	8	0x0000149A	-	function	IsWindow
ascii	8	0x000014DA	-	function	SetFocus
ascii	11	0x00001504	-	function	ExitProcess
ascii	13	0x00001526	-	function	IsBadWritePtr
ascii	10	0x00001536	-	function	LocalAlloc
ascii	9	0x00001544	-	function	LocalFree
ascii	9	0x00001550	-	function	LocalSize
ascii	18	0x00001576	-	function	InitCommonControls
ascii	10	0x0000159A	-	function	WFSStartUp
ascii	8	0x000015A8	-	function	WFSClose
ascii	9	0x000015B4	-	function	WFSUnlock
ascii	13	0x000015C0	-	function	WFSFreeResult
ascii	11	0x000015D0	-	function	WFSRegister
ascii	10	0x000015E8	-	function	WFSExecute
ascii	10	0x000015F6	-	function	WFSGetInfo
ascii	10	0x0000160E	-	function	WFSCleanUp

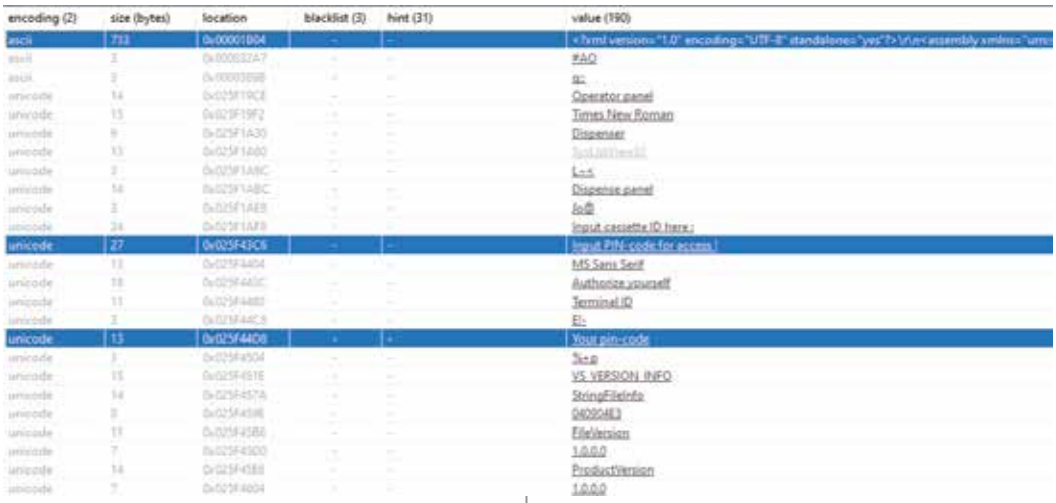


Figure 5: Malicious Strings

Dynamic Analysis:

When running this malware it asks for the pin and it only takes 123 as a pin. It didn't take any other number. When entering any other number it wasn't proceeding and when you entered 123 as pin it automatically processes the next step [21].

The "operator panel" is opened by entering a special PIN that is of 3 digits based on the terminal ID of the ATM.

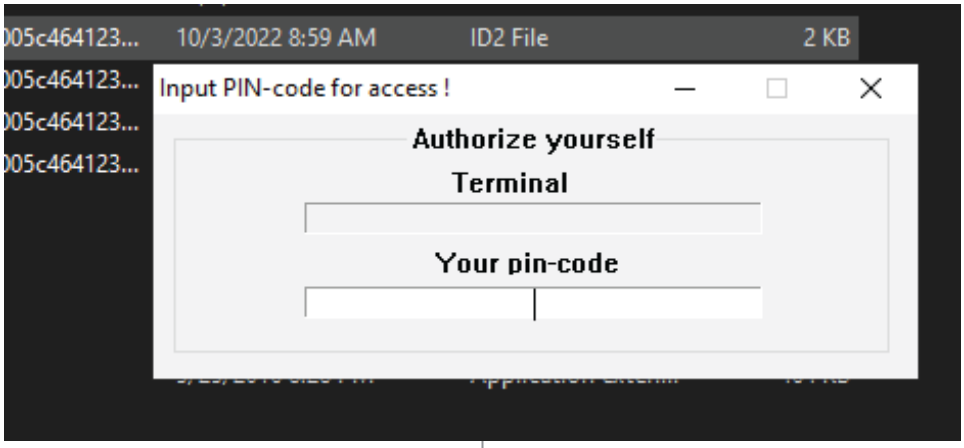


Figure 6: Running the malware

Maybe here it defines the password.

```
.text:00401031 ; int (__stdcall *dword_401031)(_DWORD)
.text:00401031 dword_401031    dd 0                    ; DATA XREF: DialogFunc+185↓r
.text:00401031                                         ; sub_401AE9+36↓r
.text:00401035                 align 4
.text:00401038                 db 0
.text:00401039 dword_401039    dd 0                    ; DATA XREF: DialogFunc+170↓r
.text:0040103D                 align 10h
.text:00401040                 db 0
.text:00401041 ; CHAR a123[4]
.text:00401041 a123            db '123',0              ; DATA XREF: sub_401AE9+D5↓o
.text:00401045 ; CHAR String2[2]
.text:00401045 String2         db '0',0                ; DATA XREF: DialogFunc+13C↓o
.text:00401045                                         ; sub_401AE9:loc_401BF6↓o
```

```
lea     ebx, [ebp+lParam]
push    ebx             ; lParam
push    104h            ; wParam
push    0Dh             ; Msg
push    7D5h            ; nIDDlgItem
push    [ebp+hDlg]      ; hDlg
call    SendDlgItemMessageA
push    offset a123     ; "123"
push    ebx             ; lpString1
call    lstrcmpiA
or      eax, eax
jnz     short loc_401BF6
```

Figure7: Password Defined

After entering the pin it asked to input the cassette id. The loaded cassettes holding the money are visible when the "operator panel" is opened. The values ID, Bills count, Bill value, Currency, and Result are shown. It displays all cassettes containing money in the machine.

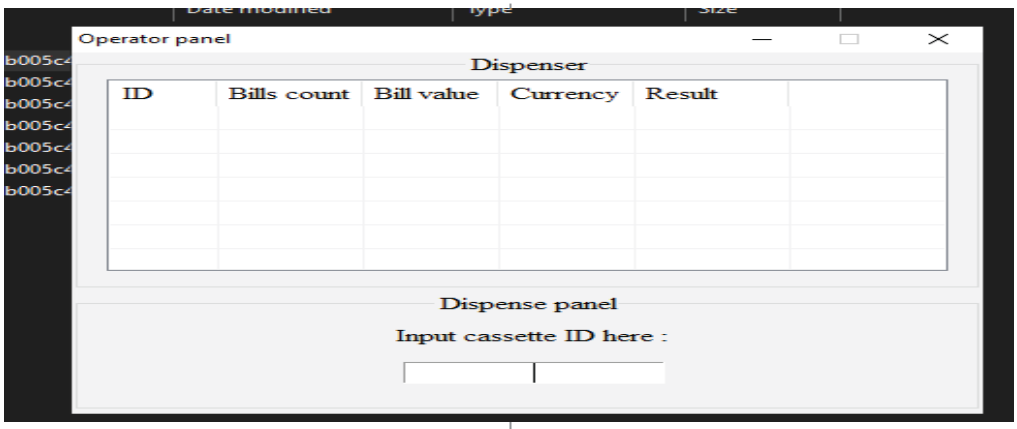


Figure 8: Entering the PIN

At this point, it detects that is it running on an ATM. When running on XFS (Extension for Financial Services)-based machines, it accepts input. For Microsoft Windows-based financial applications, particularly those that use specialized peripherals like ATMs, XFS offers a client-server architecture.

```
push    esi
push    edi
xor     edi, edi
push    edi                 ; lpModuleName
call    GetModuleHandleA
mov     hInstance, eax
push    3
call    sub_401047
push    edi                 ; dwInitParam
push    offset sub_401AE9   ; lpDialogFunc
push    edi                 ; hWndParent
push    7D0h                ; lpTemplateName
push    hInstance           ; hInstance
call    DialogBoxParamA
call    sub_401078
push    edi                 ; uExitCode
call    ExitProcess
start endp
```

Figure 9: Second call

In the second call, it calls the **sub_401078** function. In this function it calls **WFSStartUp,** a connection is made between an application and the XFS Manager by WFSStartUp. It has to be the first XFS API function a program calls. XFS functions cannot be supplied by an application until a successful WFSStartUp has finished.

After that, it made calls to the

DialogBoxParamA

```
xor     edi, edi
push    edi                 ; lpModuleName
call    GetModuleHandleA
mov     hInstance, eax
push    3
call    sub_401047
push    edi                 ; dwInitParam
push    offset sub_401AE9 ; lpDialogFunc
push    edi                 ; hWndParent
push    7D0h                ; lpTemplateName
push    hInstance           ; hInstance
call    DialogBoxParamA
call    sub_401078
push    edi                 ; uExitCode
call    ExitProcess
start endp
```

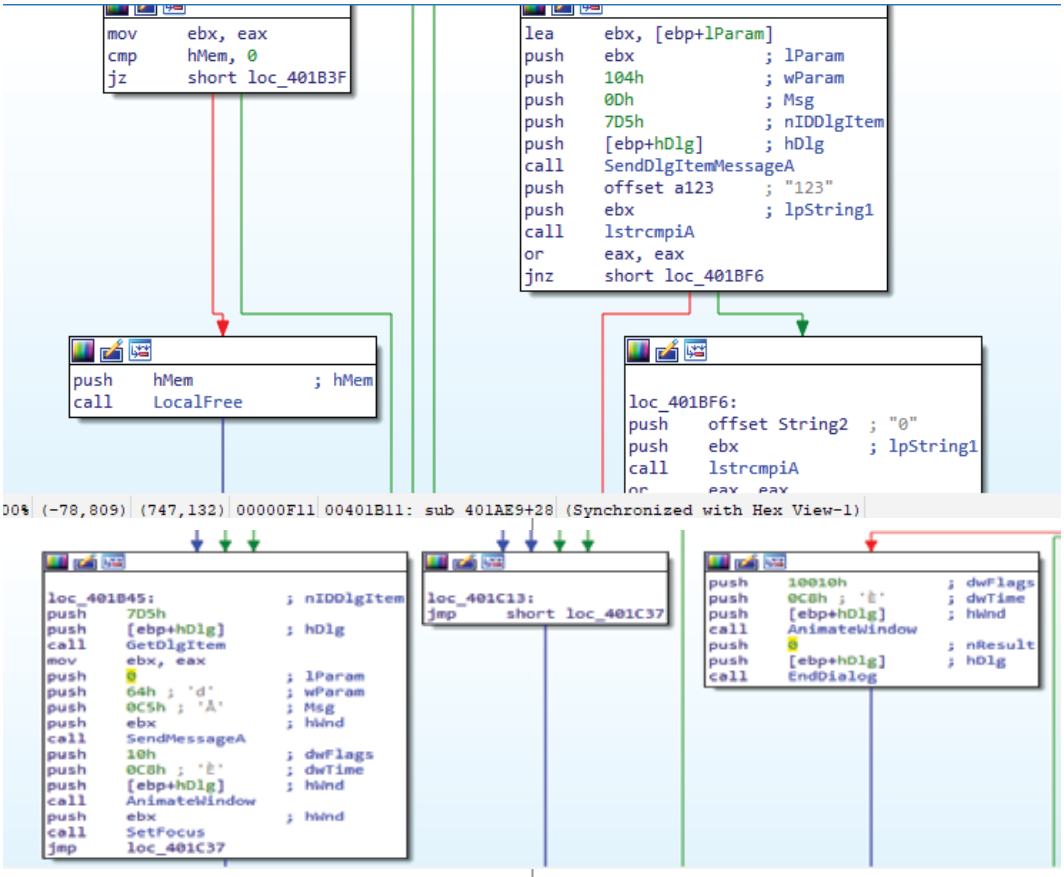
Figure 10: DialogBoxParamA

This function takes five parameters. It takes one function as a parameter.

```
void __noreturn start()
{
  hInstance = GetModuleHandleA(0);
  sub_401047(3);
  DialogBoxParamA(hInstance, (LPCSTR)0x7D0, 0, sub_401AE9, 0);
  sub_401078();
  ExitProcess(0);
}
```

Figure 11: Function as parameter

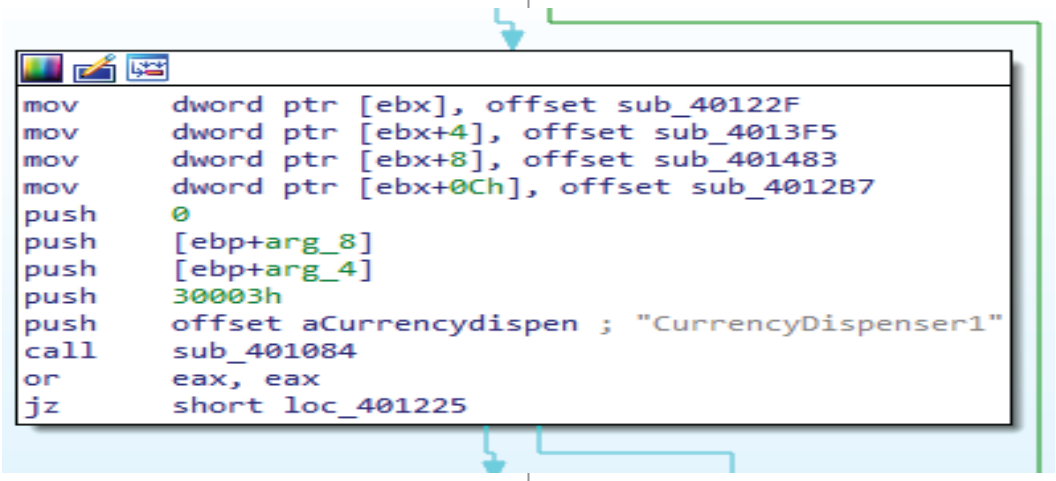
This function that is going as a parameter is managed message box that appears when we run the program and it takes decisions based on provided data is correct or not.



The highlighted function is responsible to connect the despenser1 of the ATM.

```
switch ( a2 )
{
  case 0x110u:
    InitCommonControls();
    if ( sub_4011D1(byte_40102D, 0, 0) )
    {
      v4 = dword_401031(byte_40102D);
      if ( v4 )
      {
        v5 = (void *)v4;
        if ( hMem )
          LocalFree(hMem);
        hMem = v5;
      }
    }
    DlgItem = GetDlgItem(hDlg, 2005);
    SendMessageA(DlgItem, 0xC5u, 0x64u, 0);
    AnimateWindow(hDlg, 0xC8u, 0x10u);
    SetFocus(DlgItem);
    break;
```

Alice connects to ATM's CurrenyDispenser1 peripheral, and no other hardware; therefore criminal does not need to issue any command via PIN pad.



```
mov     dword ptr [ebx], offset sub_40122F
mov     dword ptr [ebx+4], offset sub_4013F5
mov     dword ptr [ebx+8], offset sub_401483
mov     dword ptr [ebx+0Ch], offset sub_4012B7
push    0
push    [ebp+arg_8]
push    [ebp+arg_4]
push    30003h
push    offset aCurrencydispen ; "CurrencyDispenser1"
call    sub_401084
or      eax, eax
jz      short loc_401225
```

```
int __stdcall sub_4011D1(_DWORD *a1, int a2, int a3)
{
  int v3; // eax
  int v5; // [esp+214h] [ebp-4h]

  v5 = 0;
  if ( a1 )
  {
    *a1 = sub_40122F;
    a1[1] = sub_4013F5;
    a1[2] = sub_401483;
    a1[3] = sub_4012B7;
    v3 = sub_401084(aCurrencydispen, 196611, a2, a3, 0);
    if ( v3 )
    {
      a1[4] = v3;
      return 1;
    }
  }
  return v5;
}
```

```
.text:00401000 _text           segment para public 'CODE' use32
.text:00401000                 assume cs:_text
.text:00401000                 ;org 401000h
.text:00401000                 assume es:nothing, ss:nothing, ds:_text, fs:nothing, gs:nothing
.text:00401000 aCurrencydispen db 'CurrencyDispenser1',0
.text:00401000                                         ; DATA XREF: sub_4011D1+40↓o
.text:00401013 aPtr            db 'PTR',0
.text:00401017 ; HINSTANCE hInstance
.text:00401017 hInstance       dd 0                    ; DATA XREF: sub_401AE9+180↓r
.text:00401017                                         ; start+B4↓w ...
.text:00401018 dword_401018    dd 0                    ; DATA XREF: DialogFunc+324↓w
.text:0040101F ; CHAR Caption[]
.text:0040101F Caption         db 'Project Alice',0    ; DATA XREF: sub_401084+42↓o
.text:0040101F                                         ; sub_4012B7+108↓o ...
.text:0040102D byte_40102D     db 3 dup(0)             ; DATA XREF: DialogFunc+16B↓o
.text:0040102D                                         ; DialogFunc+1B0↓o ...
.text:00401030                 db 0
```

From here we can see the malware name **Project Alice**

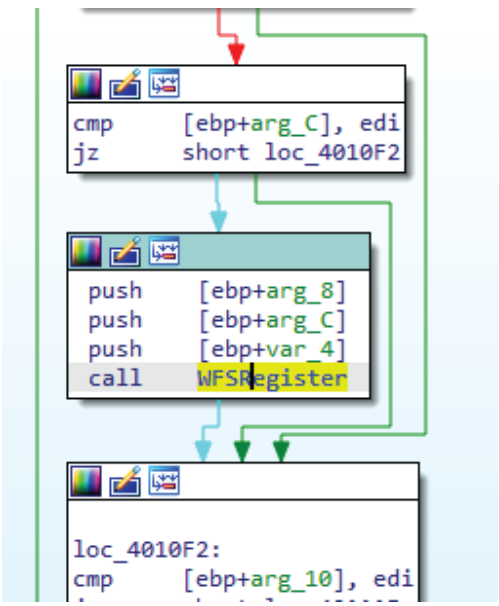
```
int __stdcall sub_4011AE(int a1)
{
  WFSClose(a1);
  return 0;
}
```

WFSClose is used to end a session or a series of service requests between the application and the designated service that was started using WFSOpen.

Then it calls WFSFreeResult, which informs the XFS manager that a memory buffer that was dynamically allocated by a service provider is ready to be released. An application uses this function to deallocate the memory.

```
            v10 = v4;
            v5 = v4;
            v6 = *(unsigned __int16 ***)(v2 + 4);
            do
            {
              v7 = *v6;
              *v5 = **v6;
              v5[1] = v7[6];
              v5[2] = v7[4];
              RtlMoveMemory(v5 + 3, (char *)v7 + 13, 3u);
              v5 += 4;
              ++v6;
              --v3;
            }
            while ( v3 );
          }
        }
        WFSFreeResult(v9);
      }
    }
    return v10;
}
```

A second call was placed. All messages of the defined classes are forwarded to the window indicated in the hWndReg argument by WFSRegister, which enables event monitoring for the specified service via the specified window. For instance, the application can call WFSRegister with the parameters SYSTEM EVENT and USER EVENT to receive data for both system and user events.



```
v15 = v5;
if ( *((_DWORD *)v4 + 1) >= 0x14u )
  v6 = 20;
else
  v6 = *((_DWORD *)v4 + 1);
*((_DWORD *)v5 + a3 - 1) = v6;
v11 = 1;
v12 = v13;
result = sub_40111F(*(_DWORD *)(a1 + 16), 302, Destination);
if ( !result || *(_DWORD *)(result + 22) )
{
  if ( result )
    v8 = *(_DWORD *)(result + 22);
  else
    v8 = -55;
  wsprintfA(Text, "Can't dispense requested amount. Error %d ocurred !", v8);
  MessageBoxA(hWnd, Text, Caption, 0x30u);
  LocalFree(v15);
  return v17;
}
return result;
```

At this point, it decides to see if input caste is available or not if available it calls the highlighted function if not then it moves on else part and displays an error.

WFSLock Establishes the application's sole authority over the designated service. Before beginning the transaction, the application needs to make sure that it has access to all the devices and that no other program will be able to utilize them until the transaction is finished. Utilizing the WFSLock function and its companion WFSUnlock allows for this.

WFSExecute communicates a command specified by the service to a service provider. To run commands supplied by the service, use this function.

A service that has been locked by a previous WFSLOCK FUNCTION is released by **WFSUnlock.**

```
int __stdcall sub_40111F(int a1, int a2, int a3)
{
  int v4; // [esp+Ch] [ebp-8h] BYREF
  int v5; // [esp+10h] [ebp-4h]

  v5 = 0;
  WFSLock(a1, 0, &v4);
  if ( v4 )
    WFSFreeResult(v4);
  v4 = 0;
  WFSExecute(a1, a2, a3, 0, &v4);
  v5 = v4;
  WFSUnlock(a1);
  return v5;
}
```

So here before starting the transaction, the application ensures that it has access to all devices and until a transaction is made, no other program can use them. For this, it uses **WFSLock** & **WFSUnlock.** After calling the function WFSLock it will call the next command for execution.

Examining the next function, it monitors if the selected caste is available or not. It gives an error if the input caste is unavailable

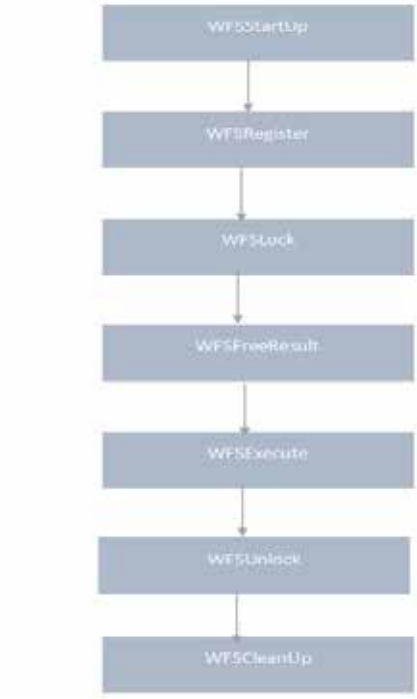
```
.text:00401260 ; CHAR Text[]
.text:00401260 Text            db 'Selected cassette is unavailable !',0
.text:00401260                                         ; DATA XREF: sub_4012B7+127↓o
.text:00401283 ; CHAR aCanTDispenseRe[]
.text:00401283 aCanTDispenseRe db 'Can',27h,'t dispense requested amount. Error %d ocurred !',0
.text:00401283                                         ; DATA XREF: sub_4012B7+F8↓o
.text:004012B7
```

Before exiting the process it calls the function.

```
void __noreturn start()
{
  hInstance = GetModuleHandleA(0);
  sub_401047(3);
  DialogBoxParamA(hInstance, (LPCSTR)0x7D0, 0, sub_401AE9, 0);
  sub_401078();
  ExitProcess(0);
}
```

In this function, it calls the **WFSCleanUp** function. An application is unplugged from the XFS manager via **WFSCleanUp**.

The Flow of XFS APIs used:



Key Points:

This executable file, when provided is not named when I start reverse engineering I found a titled 'Project Alice'. Alice ATM first asks for a pin after entering a pin opening the operator panel reveals the loaded cassettes that hold the money. While malware has one main function that it uses to connect to the currency dispenser peripheral in the ATM. During the reverse engineering process, I didn't see that if it attempts to connect to other ATM hardware such as a PIN pad. So one thing is clear at this point it is not controlled by commands issued via Pin pad.

How exactly it works:

1. It first calls **WFSStartUp** API to connect the application with the **XFS** manager.
2. After that it calls **WFSRegister** to enable event monitoring for the specified service.

3. Next, it makes a call to the WFSLock API to make sure that the application has access to all the devices before beginning the transaction and that no other application can use them until the transaction is finished. Utilizing the WFSLock function and its companion **WFSUnlock** allows for this.

4. Next, a call to **WFSFreeResult** alerts the XFS manager that a dynamically allocated memory buffer from a service provider has to be freed. An application uses this function to deallocate the memory.

5. Next, it calls **WFSExecute** to transmit a command specific to the service to a service provider. To run commands supplied by the service, use this function. Then it calls **WFSUnlock** to release a service that has been locked by a previous **WFSLOCK** function.

6. To disconnect the application from the XFS manager, it makes a final call to **WFSCleanUp.**

6. Conclusion

The globe has suddenly become a global village as a result of the extraordinary rise of information technology. As it happens, it has made the world smaller and knowledge flow more freely. Additionally, it has increased internet vulnerabilities, threats, scams, and criminal activity. The privacy of people, organizations, and states has been violated by the accessibility, user-friendly hacking tools, and complexity of cyberattacks. A good computer and network security life cycle, which comprises countermeasures, detection, and reaction, now includes incident response as a crucial component. An organization's information security policy should contain the necessary provisions, and from there, planning and organization are essential for a successful incident response effort. The planning and organizing for the incident response also includes developing a suitable incident response architecture, planning resource requirements, planning the use of technology, developing incident response procedures, cooperating with other teams and organizations, and developing appropriate metrics. Moreover, Reverse engineering encompasses a wide range of tasks, such as system data analysis and the DE compilation and disassembly of executable files and libraries. Reverse engineering is a technique used in computer security to analyze malware activities and develop solutions to stop them.

7. References

[1]. A. Javaid, “Incident Response Planning for Data Protection”. SSRN Electronic Journal. Vol. 3, no.4. pp. 21-32. 2013.

[2]. A. Ahmad, J. Hadgkiss, and A. B. Ruighaver. "Incident response teams–Challenges in supporting the organisational security function." Computers & Security. Vol. 31, no. 5. pp. 643-652. 2012.

[3]. M. Kevin, C. Prosise, and M. Pepe. “Incident response & computer forensics”. New York: McGraw-Hill, vol, 2. 2003.

[4]. M. Hausi. "Reverse engineering: a roadmap." Proceedings of the Conference on the Future of Software Engineering. 2000.

[5]. E. Eldad. “Reversing: secrets of reverse engineering”. John Wiley & Sons, 2011.

[6]. U. K. Sharath, K. Saumya and M. Madou. "Deobfuscation: Reverse engineering obfuscated code." 12th Working Conference on Reverse Engineering (WCRE'05). IEEE, 2005.

[7]. W. Wego. “Reverse engineering: Technology of reinvention”. Crc Press, 2010.

[8]. S. Bruce. "The future of incident response." IEEE Security & Privacy. Vol. 12, no.5, pp. 96-96. 2014.

[9]. W. Brown and J. Molra. Handbook for computer security incident response teams (CSIRTs). Carnegie-mellon univ pittsburgh pa software engineering inst, 2003.

[10]. R. Werlinger, K. Muldner, K. Hawkey and K. Beznosov. “Preparation, detection, and analysis: the diagnostic work of IT security incident response”. Information Management & Computer Security.vol.3, pp.13-56, 2010.

[11]. C. Rui. "Design principles for critical incident response systems." Information Systems and E-Business Management. Vol. 5, no. 3. Pp. 201-227. 2007.

[12]. W. Rodrigo. "Preparation, detection, and analysis: the diagnostic work of IT security incident response." Information Management & Computer Security. 2010.

[13]. G. George, W. B. Glisson, and T. Storer. "Rethinking security incident response: The integration of agile principles." arXiv preprint arXiv. Vol.3. pp. 1408.2431.2014.

[14]. F. Felix and B. Schwittay. "A common process model for incident response and digital forensics." Proceedings of the IMF2007. 2007.

[15]. L. Trevor. "A forensic approach to incident response”. Information Security Curriculum Development Conference. 2010.

[16]. S. Alexander. "A Cyber Incident Response and Recovery Framework to Support Operators of Industrial Control Systems." International Journal of Critical Infrastructure Protection. Vol. 37 . pp.100-105. 2022.

[17]. S. Daniel, M. Caselli, and G. Pernul. "A comparative study on cyber threat intelligence: the security incident response perspective." IEEE Communications Surveys & Tutorials. vol. 23, no. 4. Pp. 2525-2556. 2021.

[18]. H. A. Müller, J. H. Jahnke, D. B. Smith, and M. A. Storey. “Reverse engineering: a roadmap. InProceedings of the Conference on the Future of Software Engineering”. pp. 47-60. 2000.

[19]. H. Nikhil. "Where is the debugger for my software-defined network?." Proceedings of the first workshop on Hot topics in software defined networks. 2012.

[20]. E. Stroulia and T. Systä. Dynamic analysis for reverse engineering and program understanding. ACM SIGAPP Applied Computing Review, vol.10, no. 1. pp. 8-17. 2002.

[21]. S. Eleni, and T. Systa. "Dynamic analysis for reverse engineering and program understanding." ACM SIGAPP Applied Computing Review. Vol. 10, no.1. pp. 8-17. 2002.