Vol.2022-IOT-56 No.9
2022/3/7

UEFI モジュールを悪用したマルウェアの解析

新田耕大[†1]　　山之上卓[†1]

概要：ファームウェアに対するサイバー攻撃や問題が頻発している．2006 年に仕様が策定され，一般的に利用されているファームウェアの一種である UEFI BIOS では，ルートキットの Lojax の発見や UEFI モジュールを悪用した FPS ゲームのチートなどが発生している．UEFI BIOS のマルウェアは，OS 上で観測できないため検出が困難であり，問題になっている．本研究では，1.何もしない場合，2.チートでよく使われている FPS ゲームを実行させた場合，3.実行させた FPS ゲームのチートツールの一部を UEFI BIOS に実装した場合，の 3 通りの場合について UEFI Shell で UEFI 変数とその属性の内容の dump を取り，比較検討を行なった．

キーワード：セキュリティ，ファームウェア，UEFI BIOS，マルウェア

1. はじめに

ソフトウェアの一種でデバイスの制御を行うファームウェアに対するサイバー攻撃やセキュリティ上の問題が頻発している[1]．ファームウェアの実装を標準化し，レガシーBIOS から置き換えるため 2006 年に仕様が策定され，一般的に利用されているファームウェアの一種である UEFI BIOS においては，ルートキットの Lojax の発見や UEFI モジュールを悪用した FPS ゲームのチートなどが発生している．

UEFI BIOS には SPI フラッシュと呼ばれる，ハードディスクや SSD とは異なるデバイスにデータが保存されており，システムは実行時にそのデータを読み取り実行を開始している．UEFI BIOS には高い権限で実行される SMM モジュールや Secure Boot などのセキュリティ機能が実装されており，SMM モジュール以外からの SPI フラッシュへの書き込みを禁止させるレジスターが用意されている．しかし，多くの OEM と BIOS ベンダーがこれを行わないデバイスを出荷しており，SPI フラッシュへ OS やハイパーバイザーから書き込み可能な場合がある．2018 年に発見された Lojax はこの脆弱性を悪用し，カーネルモードから UEFI BIOS を書き換え，悪意のある UEFI モジュールをシステムの SPI フラッシュに書き込ませる攻撃である．書き込まれた悪意ある UEFI モジュールは，PC のブート処理中にマルウェアをストレージへ注入し，実行させる．この動作により，OS の再インストールや，ストレージの交換によっても排除できず，削除が非常に困難になる[2][3].

UEFI モジュールを悪用した FPS ゲームのチートは，UEFI BIOS にプログラムをダウンロードして実行しているためチートツールが SPI フラッシュメモリに保存される．そのため，OS の管理外にチートコードが配置されることになり，検出も排除も困難になる[4].

本研究では，1．何もしない場合，2．チートでよく使われ
ている FPS ゲームを実行させた場合，3．実行させた FPS ゲームのチートツールの一部を UEFI BIOS に実装した場合，の 3 通りの場合について UEFI Shell で UEFI 変数とその属性の内容の dump を取り，比較検討を行なった．

2. 使用するチートツールの概要

図 1 に実験に使ったチートツールの概要（推定）を示す．本研究で使用するチートツールは，CRZAIMBOT[5]で，memory.efi と CRZaimbot.dll の二つのプログラムで構成されている．通常は，アプリケーションから OS を経由して，BIOS で定義された低レベル入出力ルーチンが実行される．チートツールを導入するために，チートツールの memory.efi を loadすることによって，UEFI の変数の内容が書き代わり，元のルーチンの代わりに，後で導入する CRZaimbot.dll のルーチンが呼び出されるようになる．その後，CRZaimbot.dll を injection することで，チートプログラムが導入される．

3. 実験手法

本研究の実験手法は，次の通りである．

(1) **UEFI Shell を使えるようにする**

1．USB メモリを FAT32 でフォーマットする．

2．フォーマットした USB メモリ内に¥EFI¥BOOT のディレクトリを作成する．

3．URL[6]から Shell.efi をダウンロードし，2 で作成したディレクトリの下に配置する．

†1 福山大学

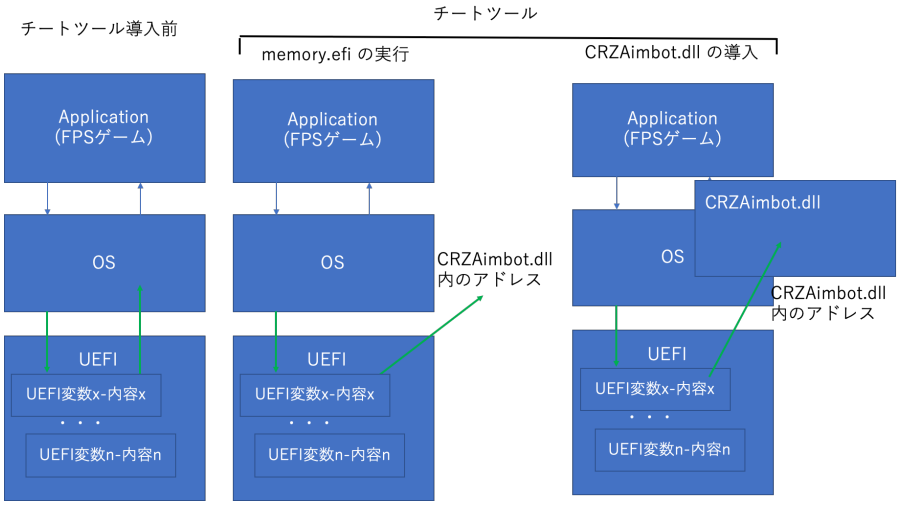

図 1　チートツールの概要（推定）

```
UEFI Interactive Shell v2.2
EDK II
UEFI v2.50 (American Megatrends, 0x0005000C)
Mapping table
      FS0: Alias(s):HD0a0b:;BLK1:
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x0,0x0)/HD(1,MBR,0xD40D2395,0x20,0xF47E0)
      FS1: Alias(s):HD1a65535a1:;BLK3:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(1,GPT,D03F7B37-2032-4853-8FEF-3731FFAFE
414,0x800,0x82000)
      FS2: Alias(s):HD1a65535a3:;BLK5:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(3,GPT,F13E0D6E-71BC-4279-B865-FFB31DEB8
8BA,0x8A800,0x1DAA6800)
      FS3: Alias(s):HD1a65535a4:;BLK6:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(4,GPT,244123ED-A86F-4E83-8180-AC11C2099
4D4,0x1DB31000,0x1C2000)
      FS4: Alias(s):HD1c65535a2:;BLK9:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)/HD(2,GPT,4610D570-A5BE-4ED0-AB49-192B0A977
6D5,0x40800,0x746C6000)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x0,0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)
     BLK7: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)
     BLK4: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(2,GPT,200EC4A5-05E2-4D6A-914B-1624B1FAA
BE5,0x82800,0x8000)
     BLK8: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)/HD(1,GPT,59E8FBAB-9745-4C04-8C39-8CDAFA317
297,0x22,0x40000)
Press ESC in 2 seconds to skip startup.nsh or any other key to continue.
```

図 2　UEFI Shell の実行画面

4. 配置したファイルをアーキテクチャに合わせて以下の通りにリネームする（表 1）.

表 1　アーキテクチャごとのリネーム内容

Architecture	File
x86 (32-bit)	bootia32. efi
x64 (64-bit)	bootx64. efi
ARM (32-bit)	bootarm. efi
ARM (64-bit)	bootaa64. efi

5. ダウンロードした EFI ファイルはマイクロソフトの署名がないため UEFI BIOS のセット画面を開き, Security メニューで Secure Boot を Disabled する.

6. Exit メニューで Save Changed し, Boot Override から USB メモリを選択して（図 3）UEFI Shell を実行させる（図 2）.

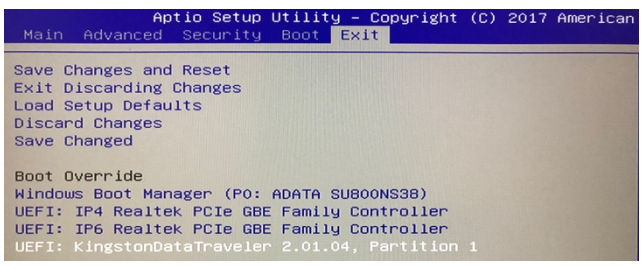

図 3　UEFI BIOS セット画面

```
FS0:\EFISCREEN\> map
Mapping table
      FS0: Alias(s):HD0a0b:;BLK1:
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x0,0x0)/HD(1,MBR,0xD40D2395,0x20,0xF47E0)
      FS1: Alias(s):HD1a65535a1:;BLK3:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(1,GPT,D03F7B37-2032-4853-8FEF-3731FFAFE
414,0x800,0x82000)
      FS2: Alias(s):HD1a65535a3:;BLK5:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(3,GPT,F13E0D6E-71BC-4279-B865-FFB31DEB8
8BA,0x8A800,0x1DAA6800)
      FS3: Alias(s):HD1a65535a4:;BLK6:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(4,GPT,244123ED-A86F-4E83-8180-AC11C2099
4D4,0x1DB31000,0x1C2000)
      FS4: Alias(s):HD1c65535a2:;BLK9:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)/HD(2,GPT,4610D570-A5BE-4ED0-AB49-192B0A977
6D5,0x40800,0x746C6000)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x0,0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)
     BLK7: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)
     BLK4: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(2,GPT,200EC4A5-05E2-4D6A-914B-1624B1FAA
BE5,0x82800,0x8000)
     BLK8: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)/HD(1,GPT,59E8FBAB-9745-4C04-8C39-8CDAFA317
297,0x22,0x40000)
```

図 4　map コマンドの実行結果

(2) **UEFI Shell から dump を取り，USB メモリ内にテキストファイルとして保存する**

1．map コマンドでマウント可能なドライブ一覧を表示し USB メモリを探す．図 4 の FS0 は USB メモリ，FS1 は EFI システムパーティション，FS2 は C ドライブ，FS3 は空のパーティション，FS4 は D ドライブである．BLK は動的パーティションである．

2．FS0 が USB メモリであるため FS0:と入力し，マウントする（図 5）．

```
Shell> fs0:
FS0:\> ls
Directory of: FS0:\
01/31/2022  02:11 <DIR>         4,096  EFISCREEN
01/31/2022  01:59 <DIR>         4,096  EFI
        0 File(s)          0 bytes
        2 Dir(s)
```

図 5　ドライブのマウント

3．dmpstore コマンドですべての UEFI 変数とその属性の内容がダンプされる．そのため，dump.txt にリダイレクトして USB メモリ内部に保存する（図 6）．

```
FS0:\> dmpstore > dump.txt
FS0:\> ls
Directory of: FS0:\
01/31/2022  02:11 <DIR>         4,096  EFISCREEN
01/31/2022  01:59 <DIR>         4,096  EFI
01/31/2022  03:03             142,116  dump.txt
        1 File(s)     142,116 bytes
        2 Dir(s)
```

図 6　ダンプの取得

(3) **パターンに分けてダンプを取り，テキストファイルを比較して差分を取る**

テキストファイルの比較は図 7 に示す．

4. 実験内容

4.1 ダンプ取得パターン

本実験でダンプを取得するパターンは以下の通りである

(1) ゲーム実行前
(2) ゲームプレイ後
(3) チートをダウンロードした時
(4) (3)からゲームプレイ後

4.2 使用するゲームとチート

本実験では，”APEX LEGENDS”という FPS ゲームと CRZAIMBOT という Aimbot を実現するチートツールを使用する．

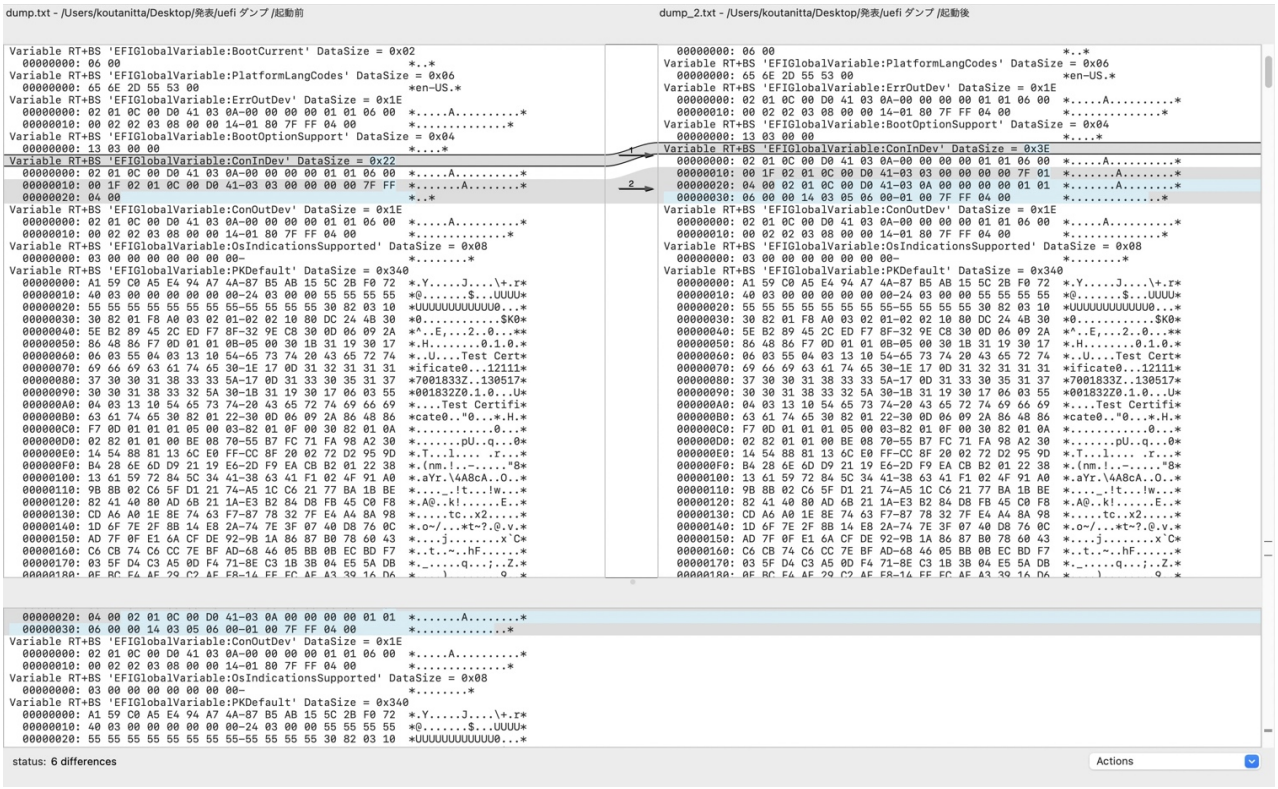

図 7　テキストファイルの比較

4.3 差分分析

本実験では，3.1 で取得したデータからそれぞれ差分を取り UEFI 変数とその属性の把握し，(1)~(4)の過程でどのような変化があるのかを分析する．

5. 実験結果

本実験で使用した CRZAIMBOT は EFI ファイルの実行までは成功したが，FPS ゲームの現在のシーズンに対応していなかったため導入に失敗し 3.1 の(3)と(4)を行うことができなかった．

そのため，1.何もしない場合，2.チートでよく使われている FPS ゲームを実行させた場合，3.memory.efi の実行後のダンプを取り， 1 と 2 との差分，1 と 3 の差分を検出して分析を行なった．

6. 差分分析

6.1 ゲーム実行前とゲーム実行後の差分（1 と 2 の差分）

その 1.

変数“Variable RT+BS ‘EFIGlobalVariable:ConInDev’”の内容に差分が現れた（図 8，9）．ここで，RT はランタイムアクセスのことであり，UEFI ブートサービスが完了した後でも，OS の中などから問い合わせることができることを示す．BS は，ブートサービスアクセスのことであり，EFI ファイルなど、ブートサービス実行ファイルから問い合わせ可能であることを示す．“EFIGlobalVariable”は，EFI グローバル変数であることを示す．“CoinDev”は，使用可能であるコンソール入力デバイスのデバイスパスを示す．従って，この変数は揮発性で実行のたびに動的に設定されるコンソール入力デバイスのデバイスパスを示している[7].

その 2.

変数“Variable NV+RT+BS ‘EFIGlobalVariable:OsIndications’”の位置に差分が現れた（図 10，11）．実行前は，“Variable NV+RT+BS 'EFIGlobalVariable:Boot0006' DataSize = 0xE”の後ろ，実行後は前に位置している．ここで，NV は不揮発性であることを示す．従って，永続的なストレージに保存されるため，ブートに関係なく利用可能である．“OsIndications”は，OS がファームウェアに有効にさせたい機能および OS がファームウェアに行わせたいアクションを示すために使われる変数であることを示す．“Boot0006”は，名前 Boot に 4 桁の 16 進数の固有番号を付加したものであり，属性やファイルパスのリストの長さが格納されている[7].

その 3.

変数“Variable NV+RT+BS ’EFIGlobalVariable:ConIn’”の内容に差分が現れた（図 12，13）．ここで，“ConIn”は，デフォルトの入力コンソールのデバイスパスを示す．属性 NV が付いているためこの変数は不揮発性である[7].

6.2 memory.efi の実行前と実行後の差分（1 と 3 の差分）

6.1 のその 2 と同じ差分結果になり，その他の差分は見られなかった（図 14，15）．

6.3 分析のまとめ

6.1 の差分から推測とは異なり，memory.efi を実行せずとも FPS ゲームを実行することで UEFI 変数の値は書き換わることがわかった．Momory.efi のソースプログラムによると，多くの変数の中身が書き換わるはずであるが今回の実験では 6.2 の差分しか見ることができなかった．その原因は，UEFI BIOS には EFI グローバル変数以外にも変数があり，それらを dump することができなかったからだと思われる [5].

```
Variable RT+BS 'EFIGlobalVariable:ConInDev' DataSize = 0x22
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 1F 02 01 0C 00 D0 41-03 03 00 00 00 00 7F FF  *.......A........*
  00000020: 04 00                                            *..*
```

図 8 ゲーム実行前その 1

```
Variable RT+BS 'EFIGlobalVariable:ConInDev' DataSize = 0x3E
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 1F 02 01 0C 00 D0 41-03 03 00 00 00 00 7F 01  *.......A........*
  00000020: 04 00 02 01 0C 00 D0 41-03 0A 00 00 00 00 01 01  *.......A........*
  00000030: 06 00 00 14 03 05 06 00-01 00 7F FF 04 00        *..............*
```

図 9 ゲーム実行後その 1

```
Variable NV+RT+BS 'EFIGlobalVariable:Boot0006' DataSize = 0xEA
  00000000: 01 00 00 00 80 00 55 00-45 00 46 00 49 00 3A 00  *......U.E.F.I.:.*
  00000010: 20 00 4B 00 69 00 6E 00-67 00 73 00 74 00 6F 00  * .K.i.n.g.s.t.o.*
  00000020: 6E 00 44 00 61 00 74 00-61 00 54 00 72 00 61 00  *n.D.a.t.a.T.r.a.*
  00000030: 76 00 65 00 6C 00 65 00-72 00 20 00 32 00 2E 00  *v.e.l.e.r. .2...*
  00000040: 30 00 31 00 2E 00 30 00-34 00 2C 00 20 00 50 00  *0.1...0.4.,. .P.*
  00000050: 61 00 72 00 74 00 69 00-74 00 69 00 6F 00 6E 00  *a.r.t.i.t.i.o.n.*
  00000060: 20 00 31 00 00 00 02 01-0C 00 D0 41 03 0A 00 00  * .1........A....*
  00000070: 00 00 01 01 06 00 00 14-03 05 06 00 00 00 04 01  *................*
  00000080: 2A 00 01 00 00 00 20 00-00 00 00 00 00 00 E0 47  **..... ........G*
  00000090: 0F 00 00 00 00 00 95 23-0D D4 00 00 00 00 00 00  *.......#........*
  000000A0: 00 00 00 00 00 00 01 01-7F FF 04 00 01 04 36 00  *..............6.*
  000000B0: EF 47 64 2D C9 3B A0 41-AC 19 4D 51 D0 1B 4C E6  *.Gd-.;.A..MQ..L.*
  000000C0: 30 00 44 00 38 00 30 00-44 00 35 00 36 00 31 00  *0.D.8.0.D.5.6.1.*
  000000D0: 37 00 31 00 43 00 31 00-36 00 39 00 36 00 33 00  *7.1.C.1.6.9.6.3.*
  000000E0: 00 00 7F FF 04 00 00 00-42 4F                    *........BO*
Variable NV+RT+BS 'EFIGlobalVariable:OsIndications' DataSize = 0x08
  00000000: 00 00 00 00 00 00 00 00-                         *........*
```

図 10 ゲーム実行前その 2

```
Variable NV+RT+BS 'EFIGlobalVariable:OsIndications' DataSize = 0x08
  00000000: 00 00 00 00 00 00 00 00-                         *........*
Variable NV+RT+BS 'EFIGlobalVariable:Boot0006' DataSize = 0xEA
  00000000: 01 00 00 00 80 00 55 00-45 00 46 00 49 00 3A 00  *......U.E.F.I.:.*
  00000010: 20 00 4B 00 69 00 6E 00-67 00 73 00 74 00 6F 00  * .K.i.n.g.s.t.o.*
  00000020: 6E 00 44 00 61 00 74 00-61 00 54 00 72 00 61 00  *n.D.a.t.a.T.r.a.*
  00000030: 76 00 65 00 6C 00 65 00-72 00 20 00 32 00 2E 00  *v.e.l.e.r. .2...*
  00000040: 30 00 31 00 2E 00 30 00-34 00 2C 00 20 00 50 00  *0.1...0.4.,. .P.*
  00000050: 61 00 72 00 74 00 69 00-74 00 69 00 6F 00 6E 00  *a.r.t.i.t.i.o.n.*
  00000060: 20 00 31 00 00 00 02 01-0C 00 D0 41 03 0A 00 00  * .1........A....*
  00000070: 00 00 01 01 06 00 00 14-03 05 06 00 00 00 04 01  *................*
  00000080: 2A 00 01 00 00 00 20 00-00 00 00 00 00 00 E0 47  **..... ........G*
  00000090: 0F 00 00 00 00 00 95 23-0D D4 00 00 00 00 00 00  *.......#........*
  000000A0: 00 00 00 00 00 00 01 01-7F FF 04 00 01 04 36 00  *..............6.*
  000000B0: EF 47 64 2D C9 3B A0 41-AC 19 4D 51 D0 1B 4C E6  *.Gd-.;.A..MQ..L.*
  000000C0: 30 00 44 00 38 00 30 00-44 00 35 00 36 00 31 00  *0.D.8.0.D.5.6.1.*
  000000D0: 37 00 31 00 43 00 31 00-36 00 39 00 36 00 33 00  *7.1.C.1.6.9.6.3.*
  000000E0: 00 00 7F FF 04 00 00 00-42 4F                    *........BO*
```

図 11 ゲーム実行後その 2

```
Variable NV+RT+BS 'EFIGlobalVariable:ConIn' DataSize = 0x22
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 1F 02 01 0C 00 D0 41-03 03 00 00 00 00 7F FF  *.......A........*
  00000020: 04 00                                            *..*
```

図 12 ゲーム実行前その 3

```
Variable NV+RT+BS 'EFIGlobalVariable:ConIn' DataSize = 0x3E
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A.........*
  00000010: 00 1F 02 01 0C 00 D0 41-03 03 00 00 00 00 7F 01  *.......A........*
  00000020: 04 00 02 01 0C 00 D0 41-03 0A 00 00 00 00 01 01  *.......A........*
  00000030: 06 00 00 14 03 05 06 00-01 00 7F FF 04 00        *..............*
```

図 13　ゲーム実行後その 3

```
Variable NV+RT+BS 'EFIGlobalVariable:Boot0006' DataSize = 0xEA
  00000000: 01 00 00 00 80 00 55 00-45 00 46 00 49 00 3A 00  *......U.E.F.I.:.*
  00000010: 20 00 4B 00 69 00 6E 00-67 00 73 00 74 00 6F 00  * .K.i.n.g.s.t.o.*
  00000020: 6E 00 44 00 61 00 74 00-61 00 54 00 72 00 61 00  *n.D.a.t.a.T.r.a.*
  00000030: 76 00 65 00 6C 00 65 00-72 00 20 00 32 00 2E 00  *v.e.l.e.r. .2...*
  00000040: 30 00 31 00 2E 00 30 00-34 00 2C 00 20 00 50 00  *0.1...0.4.,. .P.*
  00000050: 61 00 72 00 74 00 69 00-74 00 69 00 6F 00 6E 00  *a.r.t.i.t.i.o.n.*
  00000060: 20 00 31 00 00 00 02 01-0C 00 D0 41 03 0A 00 00  * .1........A....*
  00000070: 00 00 01 01 06 00 00 14-03 05 06 00 00 00 04 01  *................*
  00000080: 2A 00 01 00 00 00 20 00-00 00 00 00 00 00 E0 47  **..... ........G*
  00000090: 0F 00 00 00 00 00 95 23-0D D4 00 00 00 00 00 00  *.......#........*
  000000A0: 00 00 00 00 00 00 01 01-7F FF 04 00 01 04 36 00  *..............6.*
  000000B0: EF 47 64 2D C9 3B A0 41-AC 19 4D 51 D0 1B 4C E6  *.Gd-.;.A..MQ..L.*
  000000C0: 30 00 44 00 38 00 30 00-44 00 35 00 36 00 31 00  *0.D.8.0.D.5.6.1.*
  000000D0: 37 00 31 00 43 00 31 00-36 00 39 00 36 00 33 00  *7.1.C.1.6.9.6.3.*
  000000E0: 00 00 7F FF 04 00 00 00-42 4F                    *........BO*
Variable NV+RT+BS 'EFIGlobalVariable:OsIndications' DataSize = 0x08
  00000000: 00 00 00 00 00 00 00 00-                         *........*
```

図 14　memory．efi 実行前

```
Variable NV+RT+BS 'EFIGlobalVariable:OsIndications' DataSize = 0x08
  00000000: 00 00 00 00 00 00 00 00-                         *........*
Variable NV+RT+BS 'EFIGlobalVariable:Boot0006' DataSize = 0xEA
  00000000: 01 00 00 00 80 00 55 00-45 00 46 00 49 00 3A 00  *......U.E.F.I.:.*
  00000010: 20 00 4B 00 69 00 6E 00-67 00 73 00 74 00 6F 00  * .K.i.n.g.s.t.o.*
  00000020: 6E 00 44 00 61 00 74 00-61 00 54 00 72 00 61 00  *n.D.a.t.a.T.r.a.*
  00000030: 76 00 65 00 6C 00 65 00-72 00 20 00 32 00 2E 00  *v.e.l.e.r. .2...*
  00000040: 30 00 31 00 2E 00 30 00-34 00 2C 00 20 00 50 00  *0.1...0.4.,. .P.*
  00000050: 61 00 72 00 74 00 69 00-74 00 69 00 6F 00 6E 00  *a.r.t.i.t.i.o.n.*
  00000060: 20 00 31 00 00 00 02 01-0C 00 D0 41 03 0A 00 00  * .1........A....*
  00000070: 00 00 01 01 06 00 00 14-03 05 06 00 00 00 04 01  *................*
  00000080: 2A 00 01 00 00 00 20 00-00 00 00 00 00 00 E0 47  **..... ........G*
  00000090: 0F 00 00 00 00 00 95 23-0D D4 00 00 00 00 00 00  *.......#........*
  000000A0: 00 00 00 00 00 00 01 01-7F FF 04 00 01 04 36 00  *..............6.*
  000000B0: EF 47 64 2D C9 3B A0 41-AC 19 4D 51 D0 1B 4C E6  *.Gd-.;.A..MQ..L.*
  000000C0: 30 00 44 00 38 00 30 00-44 00 35 00 36 00 31 00  *0.D.8.0.D.5.6.1.*
  000000D0: 37 00 31 00 43 00 31 00-36 00 39 00 36 00 33 00  *7.1.C.1.6.9.6.3.*
  000000E0: 00 00 7F FF 04 00 00 00-42 4F                    *........BO*
```

図 15　memory．efi 実行後

7. 関連研究

7.1 「UEFI BIOS セキュリティ」の講義および事前学習資料

丹田氏は UEFI BIOS セキュリティに関する講義資料を作成している．本研究は，この講義資料の内容を実証しようとしたものである[4].

7.2 Lojax に関する ESET の研究白書

ESET の研究白書によると，少なくとも 2017 年から Lojax が見つかっている．この研究白書で Lojax が SPI フラッシュメモリの脆弱性又は設定ミスを突いて悪意ある UEFI モジュールをインストールすること，この UEFI モジュールがファームウェアに侵入し Windows の再インストールやハードディスクの交換では除去できないことについて述べられている．本研究では，Lojax ではなく FPS ゲームのチートツールについて研究しようとしている[8].

7.3 機械学習を用いた UEFI マルウェアの検出

ESET は，自らが所有する UEFI スキャナーを用いて収集したテレメトリデータ（遠隔的に取得したデータ）を活用し，UEFI 実行ファイルのための特別な処理用パイプラインを考案した。これは機械学習を利用して，受信したサンプルの中から異常を検出するものである[9].

本研究では，これらの研究も参考にして最終的には UEFI マルウェアへの理解を深め，安全なファームウェアの構築を行いたい．

8. おわりに

本研究では，UEFI Shell を使用したが，今までに UEFI BIOS を触れることがなく，事前知識では UEFI マルウェアに関するもののみであったため，当初，UEFI Shell でハード

ディスク内のファイルへリダイレクトができないことに気がつかなかった．今回，FPS ゲームのシーズン違いによってチートツールの導入に失敗したが，チートツールにはゲームの対応バージョンが記載されていなかったため，FPS ゲームのシーズンは実際にゲームをプレイするか，ゲームの最新情報を知っていないとわからなかった．参考となる UEFI に関する論文が日本語ではほとんど見つからなかった．今回，6．2 の差分をほとんど見ることができなかった．その原因は，UEFI BIOS には EFI グローバル変数以外にも変数があったからである．どうやら，dmpstore コマンドにはオプションがあり dmpstore -all と入力することで EFI グローバル変数以外の変数も dump できることを知ったが，この論文には間に合わなかった．

参考文献

[1] Microsoft Security Team，” New Security Signals study shows firmware attacks on the rise; here’s how Microsoft is working to help eliminate this entire class of threats”，March 30， 2021
[2] 丹田賢．https://github．com/tandasat/SecurityCamp/blob/main/%5BC4%5DUEFI%20BIOS%E3%82%BB%E3%82%AD%E3%83%A5%E3%83%AA%E3%83%86%E3%82%A3%E3%81%AE%E4%BA%8B%E5%89%8D%E5%AD%A6%E7%BF%92．md，[C4]UEFI BIOS セキュリティの事前学習．md，SecurityCamp2021 資料，(参照 2022-01-31).
[3] 佐藤 岳大．OS 再インストールや HDD 交換でも排除できない、UEFI ルートキット「LoJax」，https://pc．watch．impress．co．jp/docs/news/1145336．html，(参照 2022-01-31).
[4] 丹田賢．https://github．com/tandasat/SecurityCamp/blob/main/GCC2022/UEFI%20BIOS%20Security．pdf，UEFI BIOS Security．pdf，SecurityCamp2021 資料，(参照 2022-01-31).
[5] Rasyid Abdul Halim S．Tr．kom． https://github．com/rasyidabdulhalim/CRZAIMBOT，(参照 2022-01-31).
[6] https://github．com/tianocore/edk2/blob/vUDK2017/ShellBinPkg/UefiShell/X64/Shell．efi，(参照 2022-01-31).
[7] https://uefi．org/sites/default/files/resources/ UEFI_Spec_2_8_final．pdf，(参照 2022-01-31).
[8] https://www．welivesecurity．com/2018/09/27/ lojax-first-uefi-rootkit-found-wild-courtesy-sednit-group/，(参照 2022-01-31).
[9] https://ascii．jp/elem/000/002/006/2006911/，(参照 2022-01-31).