Informe Código Dañino
CCN-CERT ID-05/21

Pysa ransomware

Marzo 2021

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

Fecha de Edición: marzo de 2021

LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

AVISO LEGAL

ÍNDICE

1. SOBRE CCN-CERT, CERT GUBERNAMENTAL NACIONAL

El CCN-CERT es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN, adscrito al Centro Nacional de Inteligencia, CNI. Este servicio se creó en el año 2006 como **CERT Gubernamental Nacional español** y sus funciones quedan recogidas en la Ley 11/2002 reguladora del CNI, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre.

Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas, incluyendo la coordinación a nivel público estatal de las distintas Capacidades de Respuesta a Incidentes o Centros de Operaciones de Ciberseguridad existentes.

Todo ello, con el fin último de conseguir un ciberespacio más seguro y confiable, preservando la información clasificada (tal y como recoge el art. 4. F de la Ley 11/2002) y la información sensible, defendiendo el Patrimonio Tecnológico español, formando al personal experto, aplicando políticas y procedimientos de seguridad y empleando y desarrollando las tecnologías más adecuadas a este fin.

De acuerdo a esta normativa y la Ley 40/2015 de Régimen Jurídico del Sector Público es competencia del CCN-CERT la gestión de ciberincidentes que afecten a cualquier organismo o empresa pública. En el caso de operadores críticos del sector público la gestión de ciberincidentes se realizará por el CCN-CERT en coordinación con el CNPIC.

2. INFORMACIÓN DEL CÓDIGO DAÑINO

El presente documento recoge el análisis sobre los componentes con las siguientes firmas:

Hash SHA-1
425209B891142704462BAF14048D0DD59D0C7561
6B6855931E69D27F5F2E2D828FBEB4DB91688996

Dado que ambos componentes son prácticamente similares, el análisis se enfocará en el primero, con fecha de compilación más reciente, y en un apartado final se detallarán sus diferencias más relevantes.

3. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Es compatible con sistemas Windows de 32 y 64 bits.
- Cifra los ficheros de las unidades de disco duro o flash.
- Emplea cifrado simétrico (AES) y asimétrico (RSA).
- No requiere de conexión a internet para funcionar.
- Escribe el mensaje de rescate en el registro.
- Se autodestruye al finalizar.

4. DETALLES GENERALES

La muestra analizada utiliza el formato PE EXE (Portable Executable), es decir, se corresponde con un ejecutable para sistemas operativos Windows, concretamente para 32 bits (por lo que puede funcionar también en sistemas de 64 bits), compilado con “Microsoft Visual C/C++ 2015”.

La fecha interna de creación del programa fue el 30 de septiembre de 2020 a las 20:11:58 (UTC) horas, según se observa en la siguiente imagen. No obstante, hay que tener en cuenta que esta información puede ser fácilmente alterada.

pFile	Data	Description	Value
0000010C	014C	Machine	IMAGE_FILE_MACHINE_I386
0000010E	0007	Number of Sections	
00000110	5F74E68E	Time Date Stamp	2020/09/30 mié 20:11:58 UTC
00000114	00000000	Pointer to Symbol Table	
00000118	00000000	Number of Symbols	
0000011C	00E0	Size of Optional Header	
0000011E	0102	Characteristics	
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0100	IMAGE_FILE_32BIT_MACHINE

Figura 1. Información del código dañino.

5. CARACTERÍSTICAS TÉCNICAS

5.1 MUTEX

Comienza creando un mutex con nombre “Pysa” para garantizar su ejecución exclusiva y así evitar que existan múltiples instancias del proceso en ejecución.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  _DWORD h; // esi
  int result; // eax

  FreeConsole();
  if ( !OpenMutexA(MUTEX_ALL_ACCESS, 0, "Pysa") )
  {
    h = CreateMutexA(0, 0, "Pysa");
    EncryptFiles(0);
    EncryptFiles(1);
    SetRegKeyValues();
    ReleaseMutex(h);
    AutoDelete();
  }
  return 0;
}
```

Figura 2. Creación de mutex al inicio.

5.2 CIFRADO DE FICHEROS

El proceso de cifrado se realiza en dos rondas:

1. Cifra todos los ficheros que tengan alguna de las extensiones que mantiene en una lista predefinida.
2. Cifra el resto de los ficheros, además de escribir el mensaje de rescate en cada directorio.

Para obtener las unidades del sistema utiliza el API “GetLogicalDriveStringsW”, y selecciona únicamente las que pertenecen a unidades de disco duro o unidades flash. Después, por cada unidad detectada, crea un hilo encargado de enumerar sus ficheros y cifrarlos.

```
if ( lpRootPathName[5] >= (LPCWSTR)8 )
  v7 = lpRootPathName[0];
if ( GetDriveTypeW(v7) == DRIVE_FIXED )   // hard disk drive or flash drive.
  append_str(a1, (int)v3, (unsigned int)lpRootPathName);
LOBYTE(v16) = 1;
std::wstring::_Tidy(lpRootPathName, 1, 0);
```

Figura 3. Comprobación tipo de unidad.

Para evitar cifrar zonas críticas o funciones básicas del sistema operativo, comprueba que el nombre del fichero o directorio a procesar no contenga ninguna de las cadenas que se aprecian en la siguiente imagen:

```
whitelist[0] = L":\\Windows\\";
whitelist[1] = L"\\Boot\\";
whitelist[2] = L"\\BOOTSECT";
whitelist[3] = L"\\pagefile";
whitelist[4] = L"\\System Volume Information\\";
whitelist[5] = L"bootmgr";
whitelist[6] = L"\\Recovery";
whitelist[7] = L"\\Microsoft";
wcscpy_s(Destination, 0x400u, Source);
wsprintfW(FileName, L"%s\\*.*", Destination);
v2 = FindFirstFileW(FileName, &FindFileData);
```

Figura 4. Lista blanca de ficheros y directorios.

Asimismo, comprueba si la extensión de los ficheros coincide con alguna de la siguiente lista para omitirlos durante el cifrado:

.pysa
.README
.exe
.dll
.search-ms
.sys

En la primera ronda de cifrado, procesa únicamente los ficheros que coincidan con las siguientes extensiones, asegurándose así de cifrar primero los ficheros que considera de valor:

.doc	.001	.pbf
.xls	.acr	.qic
.docx	.bac	.sqb
.xlsx	.bak	.tis
.pdf	.backupdb	.vbk
.db	.bkf	.win
.db3	.bkup	.sql
.frm	.bup	.pst
.ib	.fbk	.mdb
.mdf	.mig	.7z
.mwb	.spf	.zip
.myd	.vhdx	.rar
.ndf	.vfd	.cad
.sdf	.avhdx	.dsd
.sql	.vmcx	.dwg
.trc	.vmrs	.pla
.wrk	.vbm	.pln
.bck	.vrb	

En la segunda ronda, cifra el resto de los ficheros sin importar su extensión, siempre y cuando no se encuentren en las listas blancas mencionadas anteriormente. Además, por cada directorio detectado, crea un fichero "Readme.README" con el mensaje de rescate.

Para realizar las operaciones criptográficas, utiliza la librería open-source "Crypto++" (http://svn.code.sf.net/p/cryptopp/code/), que mantiene enlazada estáticamente en el código.

Function name	Segment	Start
CryptoPP::BufferedTransformation::Peek(uchar &)	.text	009DC790
CryptoPP::BufferedTransformation::PeekWord16(ushort &,Crypto...	.text	009DC880
CryptoPP::SimpleKeyingInterface::SetKey(uchar const *,uint,Crypt...	.text	009DCB80
CryptoPP::BufferedTransformation::SetRetrievalChannel(std::strin...	.text	009DCC90
CryptoPP::BufferedTransformation::SkipAll(void)	.text	009DCD30
CryptoPP::TheBitBucket(void)	.text	009DCDD0

Figura 5. Métodos de librería Crypto++.

A continuación, se detalla el esquema de cifrado aplicado para cada fichero:

- Comprueba que el tamaño del fichero sea superior a 1024 bytes.
- Renombra el fichero, añadiendo la extensión ".pysa".
- Genera una clave y vector de inicialización (IV) aleatorios de 16 bytes.
- Cifra el contenido en bloques de 1024 bytes, usando el algoritmo AES 128 bits en modo CBC y usando las claves generadas en el paso anterior.
- Si el tamaño del último bloque es inferior a 1024 bytes, no lo cifra.
- Finalmente, cifra la clave y vector de inicialización (32 bytes) usando la clave pública RSA que lleva embebida en el código, y añade el resultado codificado en hexadecimal al final del fichero cifrado (últimos 2048 bytes).

Este esquema de cifrado, muy común en familias de ransomware, garantiza a los atacantes que los ficheros secuestrados solamente puedan ser descifrados usando la clave privada RSA que tienen en su posesión.

```
.rdata:010FA364                 align 8
.rdata:010FA368 rsa_public_key  db '30820220300D06092A864886F70D0101010500038202 0D00308202080282020 10'
.rdata:010FA368                                         ; DATA XREF: .data:off_1115004↓o
.rdata:010FA368                 db '0DBC6834DDC9631C363D90295C3A491913620D1D5C8566C5957A7F940A37FCE0A'
.rdata:010FA368                 db '2D77ADF5DC9984B3EB4C2F2B8C04D81A3C18DEAE63F2593A8645203E3182AD345'
.rdata:010FA368                 db '3898EDB3A3ED2AD819D66305C0150247F65D95CB131DC9F83C0D8189BC516C26E'
.rdata:010FA368                 db '370DB6B59D980BE138C156F6E78325EB16B190B8C848586B67548C2264072DF96'
.rdata:010FA368                 db '7AC12B73E38236259944907 2B8657E03B88F6B577BE4024B5DD30567916E43A93'
.rdata:010FA368                 db '82AAB1070A0B47D1AC0935CD508E80453611A1C520B83EB9291658B88B164420F'
.rdata:010FA368                 db '4CFA261871022DB7E3436051DC803C5970131CE3F7B942E6356C98F08C7C8B8F5'
.rdata:010FA368                 db '7F7E7ED93613571F00CFC6FC03CE75B5122341C50DAC1509BED63F073C5D72300'
.rdata:010FA368                 db '97D9E75A4B4A816155B2A7FD0B40C658B9E4F115EC0BFEB2AC2277666D21A5324'
.rdata:010FA368                 db '5A1E323E601D0D76E61553EC217EF110FFCB3ECF21E552878B0DD262C10257A79'
.rdata:010FA368                 db 'A9A374A164EF89A12FD3E4A489E949C67FD0463323E0FE85BAC26AEECD854EE0B'
.rdata:010FA368                 db 'EE26A1B596738952C30BAA4FD78393AF92BDEC19EE01E641B449D0AC66DEEFFBD'
.rdata:010FA368                 db '31628D4C1DA8D7BE939F005D8A79B392B019A06246512DFB8193017FE76F4DCF0'
.rdata:010FA368                 db '9E9D0597EF159F7D05A45EE20FE1E6F92E4BCC8C58D8B208FF1C24D4A14819A5D'
.rdata:010FA368                 db '1D7D150B9D42333DE76BD9E8009680E03F6C0981E75A74D58530AB5FF0B38DD95'
.rdata:010FA368                 db '511840AA8C925E28870AC558E2196BD949F356718F1C52EBDF020111',0
.rdata:010FA7B1                 align 4
```

Figura 6. Clave pública RSA 4096 bits embebida.

```
ctx = GetCTX();
result = CryptGenRandom(*ctx, dwLen, pbBuffer);
if ( !result )
{
  v7 = 15;
  v6 = 0;
  v5[0] = 0;
  std::string::assign("CryptGenRandom", 0xEu);
  v8 = 0;
```

Figura 7. Generación de clave e IV.

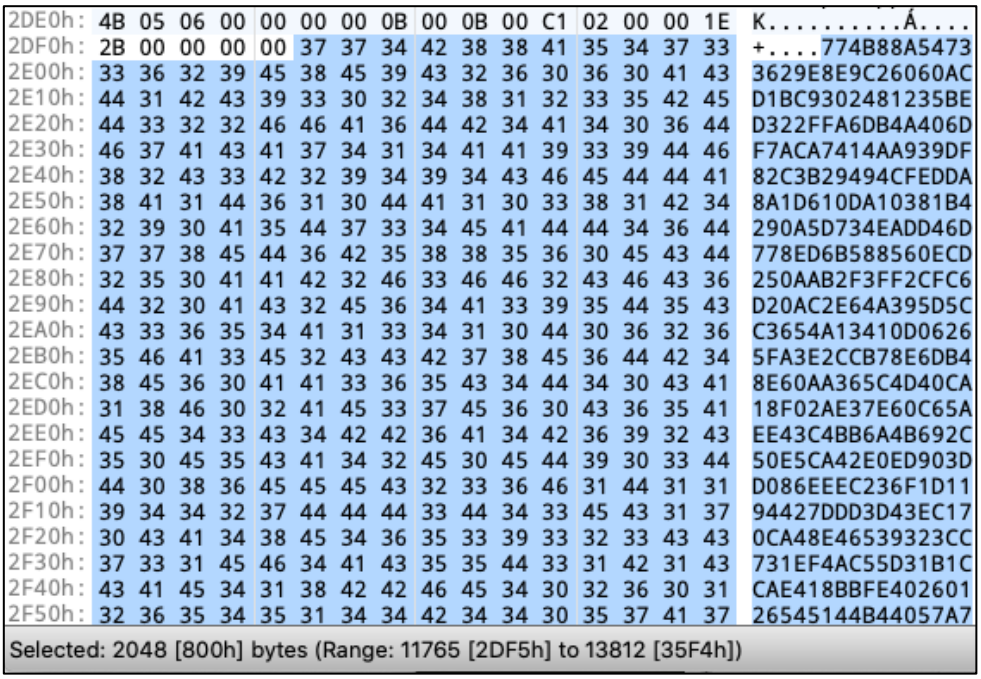

Figura 8. Clave e IV cifrados con RSA al final del fichero.

5.3 MENSAJE DE RESCATE

Con el objetivo de mostrar el mensaje de rescate durante el inicio de sesión de Windows, el código dañino modifica la siguiente clave de registro:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
Valor: legalnoticetext
Valor: legalnoticecaption

Para poder efectuar estos cambios en el registro, el programa debe ejecutarse como administrador.

```
1 LSTATUS SetRegKeyValues()
2 {
3   HKEY phkResult; // [esp+4h] [ebp-8h] BYREF
4
5   RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System", 0, 2u, &phkResult);
6   RegSetValueExA(phkResult, "legalnoticetext", 0, 7u, lpData, strlen((const char *)lpData) + 1);
7   RegSetValueExA(phkResult, "legalnoticecaption", 0, 7u, "PYSA", 5u);
8   return RegCloseKey(phkResult);
9 }
```

Figura 9. Valores de registro modificados con mensaje de rescate.

FilterAdministratorToken	REG_DWORD	0x00000000 (0)
legalnoticecaption	REG_MULTI_SZ	PYSA
legalnoticetext	REG_MULTI_SZ	Hi Company,Every byte on any types of your devices ...
PromptOnSecureDesktop	REG_DWORD	0x00000001 (1)
scforceoption	REG_DWORD	0x00000000 (0)

Figura 10. Modificación valores clave de registro.

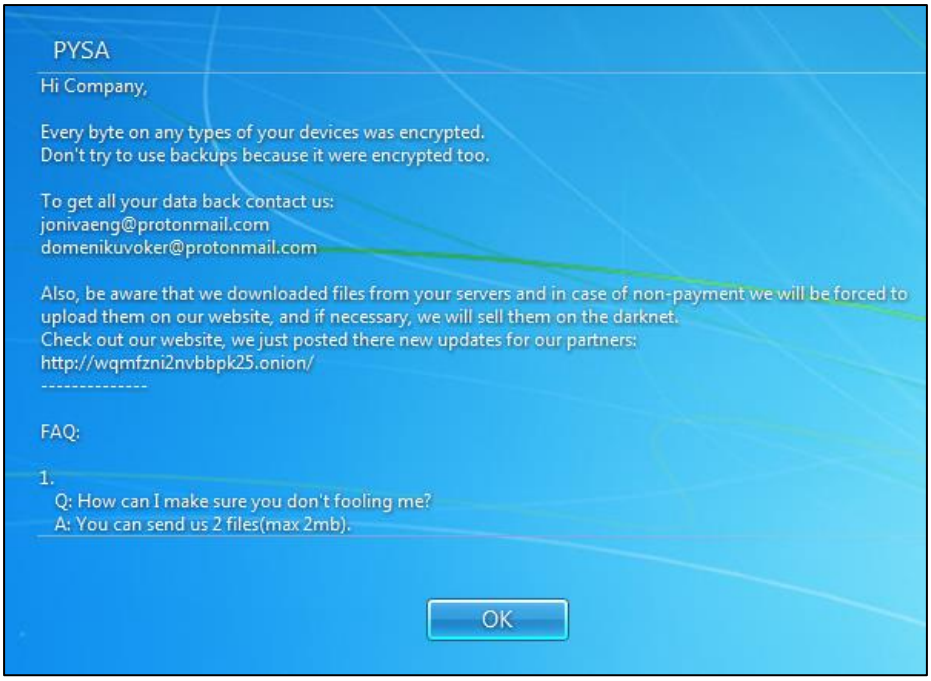

Figura 11. Mensaje de rescate al inicio de sesión.

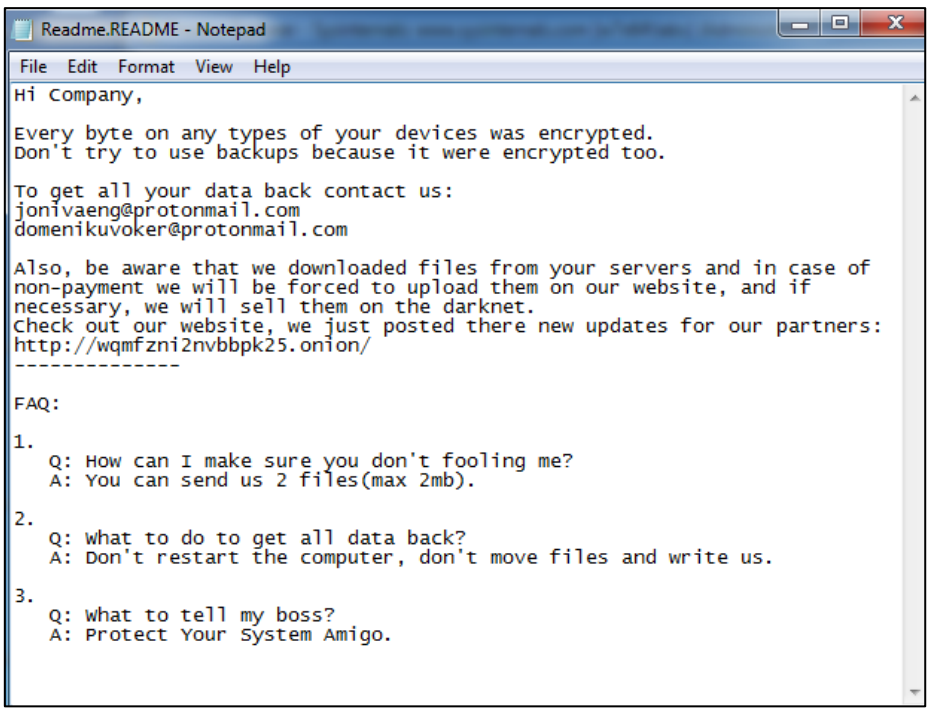

Figura 12. Mensaje de rescate “Readme.README”.

5.4 AUTOBORRADO

Una vez termina de cifrar los ficheros y crear la clave de registro, se autoborra escribiendo y ejecutando un fichero por lotes (batch file) con nombre “update.bat” en el directorio temporal.

```
hFile = CreateFileA(Buffer, 0x40000000u, 0, 0, 2u, 0x80u, 0);
if ( hFile != (void *)INVALID_HANDLE_VALUE )
{
  v7 = alloca(strlen(batch_file) + 20 + strlen(Buffer) + 2 * strlen(Filename));
  wsprintfA(
    v8,
    ":Repeat\r\ndel \"%s\"\r\nif exist \"%s\" goto Repeat\r\nrmdir \"%s\"\r\ndel \"%s\"",
    Filename,
    Filename,
    Str,
    Buffer);
  WriteFile(hFile, v8, &v8[strlen(v8) + 1] - &v8[1], &NumberOfBytesWritten, 0);
  CloseHandle(hFile);
  hFile = ShellExecuteA(0, "open", Buffer, 0, 0, 0);
}
return hFile;
```

Figura 13. Escritura y ejecución del fichero para su autoborrado.

5.5 DIFERENCIAS ENTRE MUESTRAS

A continuación, se detallan las diferencias más relevantes entre las muestras analizadas:

Hash SHA1	425209B891142704462BAF14048D0DD59D0C7561
Fecha compilación	30/09/2020 20:11:58 UTC
Algoritmo cifrado	AES 128 CBC (clave de 16 bytes e IV de 16 bytes).
Clave pública RSA	Ver en ANEXOS
Mensaje de rescate	Ver en ANEXOS

Hash SHA1	6B6855931E69D27F5F2E2D828FBEB4DB91688996
Fecha compilación	18/01/2020 23:58:47 UTC
Algoritmo cifrado	AES 256 OFB (clave de 32 bytes e IV de 16 bytes)
Clave pública RSA	Ver en ANEXOS
Mensaje de rescate	Ver en ANEXOS

6. REGLAS DE DETECCIÓN

6.1 REGLAS YARA

```
import "pe"
rule PYSA_Ransomware
{
        meta:
                author      = "Centro Criptológico Nacional (CCN)"
                date        = "15/03/2021"
                description = "PYSA ransomware"
        strings:
                $1 = "PYSA"
                $2 = "update.bat"
                $3 = "Crypto++"
                $4 = {2E0070007900730061000000}
                $5 = {4576657279206279746520616E79207479706573}
         condition:
                uint16(0) == 0x5A4D and
                pe.machine == pe.MACHINE_I386 and
                pe.number_of_sections == 7 and
                all of them
}
```

7. ANEXOS

7.1 Readme.README

7.1.1 425209B891142704462BAF14048D0DD59D0C7561

```
Hi Company,

Every byte on any types of your devices was encrypted.
Don't try to use backups because it were encrypted too.

To get all your data back contact us:
jonivaeng@protonmail.com
domenikuvoker@protonmail.com

Also, be aware that we downloaded files from your servers and in case of non-payment we will be
forced to upload them on our website, and if necessary, we will sell them on the darknet.
Check out our website, we just posted there new updates for our partners:
http://wqmfzni2nvbbpk25.onion/
--------------

FAQ:

1.
   Q: How can I make sure you don't fooling me?
   A: You can send us 2 files(max 2mb).

2.
   Q: What to do to get all data back?
   A: Don't restart the computer, don't move files and write us.

3.
   Q: What to tell my boss?
   A: Protect Your System Amigo.
```

7.1.2 6B6855931E69D27F5F2E2D828FBEB4DB91688996

```
Hi Company,

Every byte on any types of your devices was encrypted.
Don't try to use backups because it were encrypted too.

To get all your data back contact us:
raingemaximo@protonmail.com
gareth.mckie3l@protonmail.com
--------------

FAQ:

1.
   Q: How can I make sure you don't fooling me?
   A: You can send us 2 files(max 2mb).

2.
   Q: What to do to get all data back?
   A: Don't restart the computer, don't move files and write us.

3.
   Q: What to tell my boss?
   A: Protect Your System Amigo.
```

7.2 RSA public key

7.2.1 425209B891142704462BAF14048D0DD59D0C7561

```
-----BEGIN PUBLIC KEY-----
MIICIDANBgkqhkiG9w0BAQEFAAOCAg0AMIICCAKCAgEA28aDTdyWMcNj2QKVw6SR
kTYg0dXIVmxZV6f5QKN/zgotd6313JmEs+tMLyuMBNgaPBjermPyWTqGRSA+MYKt
NFOJjts6PtKtgZ1mMFwBUCR/ZdlcsTHcn4PA2BibxRbCbjcNtrWdmAvhOMFW9ueD
JesWsZC4yEhYa2dUjCJkBy35Z6wStz44I2JZlEkHK4ZX4DuI9rV3vkAktd0wVnkW
5DqTgqqxBwoLR9GsCTXNUI6ARTYRocUguD65KRZYuIsWRCD0z6JhhxAi2340NgUd
yAPFlwExzj97lC5jVsmPCMfIuPV/fn7ZNhNXHwDPxvwDznW1EiNBxQ2sFQm+1j8H
PF1yMAl9nnWktKgWFVsqf9C0DGWLnk8RXsC/6yrCJ3Zm0hpTJFoeMj5gHQ125hVT
7CF+8RD/yz7PIeVSh4sN0mLBAlenmpo3ShZO+JoS/T5KSJ6UnGf9BGMyPg/oW6wm
ruzYVO4L7iahtZZziVLDC6pP14OTr5K97BnuAeZBtEnQrGbe7/vTFijUwdqNe+k5
8AXYp5s5KwGaBiRlEt+4GTAX/nb03PCenQWX7xWffQWkXuIP4eb5LkvMjFjYsgj/
HCTUoUgZpdHX0VC51CMz3na9noAJaA4D9sCYHnWnTVhTCrX/CzjdlVEYQKqMkl4o
hwrFWOIZa9lJ81ZxjxxS698CARE=
-----END PUBLIC KEY-----
```

7.2.2 6B6855931E69D27F5F2E2D828FBEB4DB91688996

```
-----BEGIN PUBLIC KEY-----
MIICIDANBgkqhkiG9w0BAQEFAAOCAg0AMIICCAKCAgEA6dYN+TogNihncAJNXRht
Ueyj7EQ/BIGbupIMq5PRI3a1+HqMXEk5vdb3NhzFBUoVhY/jTEE71flTwHM73q9P
rgovaYSl8HeXZaU+HkqjF7Ofu4Qf+SDkoPxcubX4cFYV1r97z9vcFgFehzk+9Cof
EnHWEo2N656QGRXeO0PaJX/riiL672KHzMDNKzfZQnmpMHL+KzeyJaaPVVz7V9qC
CkjT+IT26xtG2jY5tggepfLQfB6ExxaoJ1j0GapQMIZ3k6F1AtBmfcNvyu3cW29a
bIOCsu1QRzfq6iSau2xx0ZaRz0l3vgU79PCLtsGw7BNPtKZdDL9dA879aKWlDBIi
zc3lg4IpHxdf5MOTmpQR0kst3kyOieNlIjEAyewyRQ788o3qs8k9SS+89CD916AM
EVqRcQH8ugBv5ocs0xAf+2bHe13ogIRciTz9ALTvtMSqhNptEBP/z+lIhuMTs2Mr
JRTaQLpVHUIlqAcQuLm8AHIYdGmBXEvUqPjRIo+L9Jb+P1XUcXYHvOZUBV0VFSOo
yQeqiBeaYS+PhCV6TmTRHsH/8XkPt/eGXm3Dk4feYNaZ5a9uQKYc9Akt6G0N+P8T
7zobyAWfQNqGFJhklh6JEAJw58XCJNdmETT68kfwtQ+XFB4caUHessaJ369lprAj
4TjDUFfYkkm74ntG4nVtL+sCARE=
-----END PUBLIC KEY-----
```