GuLoader? No, CloudEyE.

research.checkpoint.com/2020/guloader-cloudeye

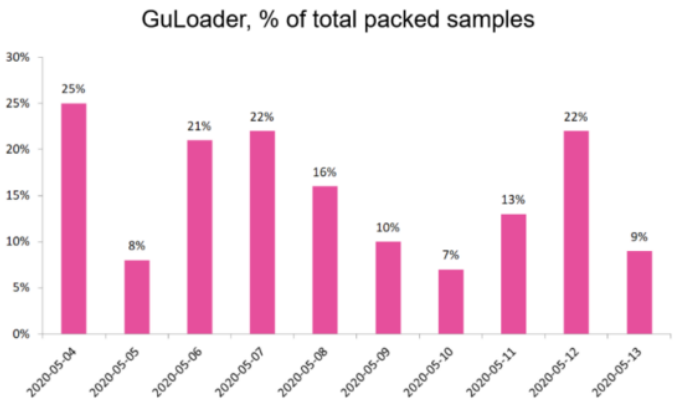
June 8, 2020

June 8, 2020

Italian company exposed on Clearnet earned up to $ 500,000 helping cybercriminals to deliver malware using cloud drives.

Recently, we wrote about the network dropper known as GuLoader, which has been very actively distributed in 2020 and is used to deliver malware with the help of cloud services such as Google Drive. The delivery of malware through cloud drives is one of the fastest growing trends of 2020.

We see hundreds of attacks involving GuLoader every day; up to 25% of all packed samples are GuLoaders. The dropper delivers a huge number of malware types, including different malicious campaigns apparently related to many different threat actors.



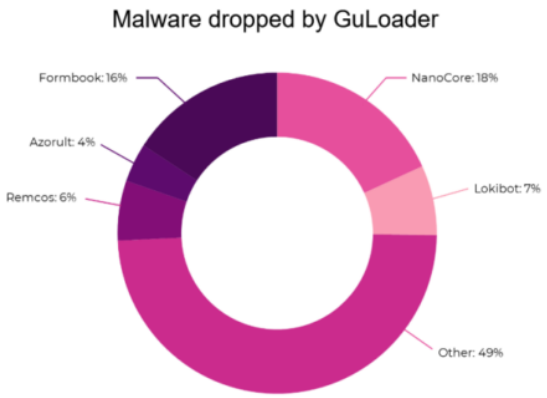


Figure 1 – Percentage of GuLoader samples and malware distributed by GuLoader.

The dropper is constantly updated: we see new versions with sandbox evasion techniques, code randomization features, C&C URL encryption, and additional payload encryption. As a result, we can reasonably assume that behind GuLoader there is a major new service aiming to replace traditional packers and crypters.

We did indeed manage to find this service, which is created and maintained by an Italian company that pretends to be completely legitimate and aboveboard, and even has a website in Clearnet that uses the .eu domain zone. But first things first.

DarkEyE

While monitoring GuLoader, we repeatedly encountered samples that were detected as GuLoader, but they did not contain URLs for downloading the payload. During manual analysis of such samples, we found that the payload is embedded in the sample itself. Those samples appear to be related to **DarkEyE Protector**:

```
部J  巷0 肃□  o\Projects\Vagodepressor.frm" -W 3 -Gy -G5 -Gs4096 -dos -Zl
-Fo"C:\Program Files (x86)\DarkEyE Protector -
Professional_Neo\Projects\Vagodepressor.OBJ"
-QIfdi景鳙 Ä*Ѣ2asic (x8炊鳟 Ä*Ѣ2C:\Program Files
```

Figure 2 – DarkEyE sample.

The DarkEyE samples have a lot in common with the GuLoader samples. They both are written in VisualBasic, contain a shellcode encrypted with 4-bytes XOR key, and have the same payload decryption procedure:

```
:003C3B27                 neg     ebx
:003C3B29                 push    edi             ; junk instruction
:003C3B2A                 xor     edi, 93B0F962h  ; junk instruction
:003C3B30                 pop     edi             ; junk instruction
:003C3B31                 mov     edi, ebx
:003C3B33                 fnop
:003C3B35
:003C3B35 decryption_loop:                        ; CODE XREF: ab_decrypt_paylod+7D↓j
:003C3B35                 mov     eax, [edx+ecx]  ; encrypted data
:003C3B38                 add     ebx, esi
:003C3B3A                 movd    mm0, eax        ; encryted bytes
:003C3B3D                 movd    mm1, dword ptr [ebx] ; decryption_key
:003C3B40                 mov     edi, edi
:003C3B42                 pxor    mm0, mm1
:003C3B45                 cmp     ecx, 698ACC2Bh  ; junk instruction
:003C3B4B                 push    ecx
:003C3B4C                 add     edi, 0C606035Bh ; junk instruction
:003C3B52                 sub     edi, 0C606035Bh ; junk instruction
:003C3B58                 movd    ecx, mm0
:003C3B5B                 mov     al, cl
:003C3B5D                 mov     edi, edi        ; junk instruction
:003C3B5F                 pop     ecx
:003C3B60                 sub     ebx, esi
:003C3B62                 add     ebx, 1
:003C3B65                 jnz     short loc_3C3B69
:003C3B67                 mov     ebx, edi
:003C3B69
:003C3B69 loc_3C3B69:                             ; CODE XREF: ab_decrypt_paylod+73↑j
:003C3B69                 mov     [edx+ecx], eax
:003C3B6C                 add     ecx, 1
:003C3B6F                 jnz     short decryption_loop
```

GuLoader

```
:00320CD2                 neg     ebx
:00320CD2
:00320CD2
:00320CD2
:00320CD4                 mov     edi, ebx
:00320CD6
:00320CD6 decryption_loop:                        ; CODE XREF: ab_decrypt_payload+91↓j
:00320CD6                 mov     eax, [edx+ecx]  ; encrypted data
:00320CD9                 add     ebx, esi
:00320CDB                 pxor    mm0, mm0
:00320CDE                 pxor    mm1, mm1
:00320CE1                 movd    mm0, eax        ; encryted bytes
:00320CE4                 movd    mm1, dword ptr [ebx] ; decryption_key
:00320CE7                 pxor    mm0, mm1
:00320CE7
:00320CEA                 push    ecx
:00320CEA
:00320CEA
:00320CEB                 movd    ecx, mm0
:00320CEE                 mov     al, cl
:00320CEE
:00320CF0                 pop     ecx
:00320CF1                 sub     ebx, esi
:00320CF3                 add     ebx, 1
:00320CF6                 jnz     short loc_320CFA
:00320CF8                 mov     ebx, edi
:00320CFA
:00320CFA loc_320CFA:                             ; CODE XREF: ab_decrypt_payload+87↑j
:00320CFA                 mov     [edx+ecx], eax
:00320CFD                 add     ecx, 1
:00320D00                 jnz     short decryption_loop
```

DarkEyE

Figure 3 – Comparison of GuLoader and DarkEyE samples.

We searched for “DarkEyE Protector” on the web and easily found a very old thread from 2014 in which it was advertised by a user known as “**xor**”:

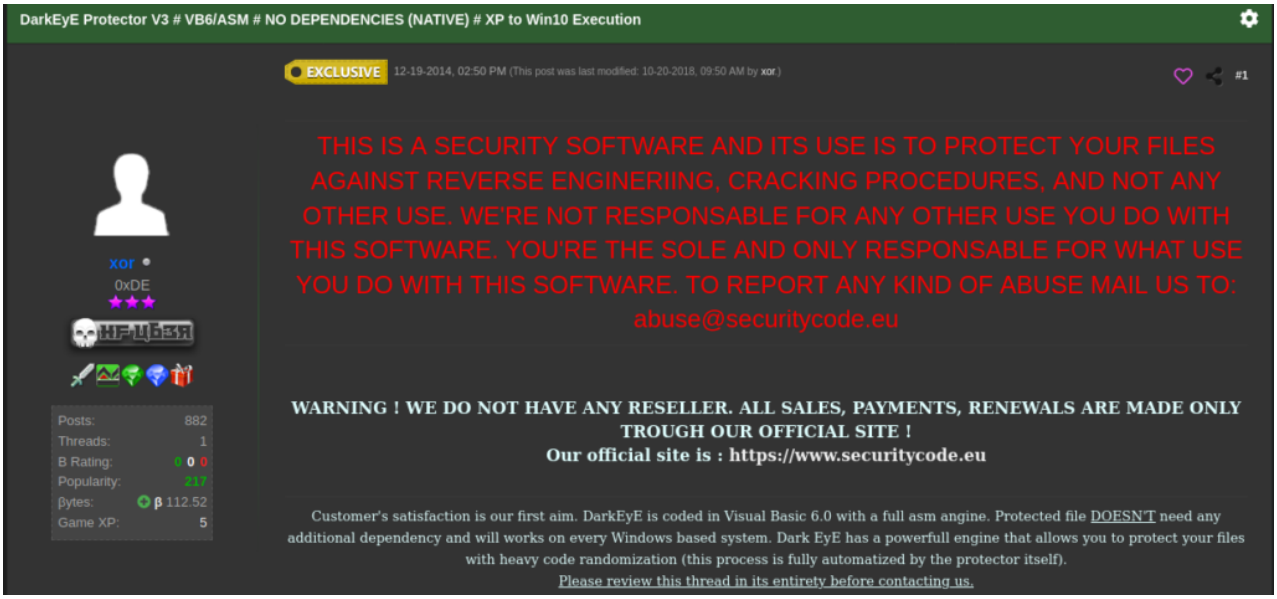


Figure 4 – DarkEyE advertisement on a hacker forum.

We also found some earlier ads for DarkEyE on the same website, these posted by the user "**sonykuccio.**" The ads describe DarkEyE as a crypter that can be used with different malware such as stealers, keyloggers, and RATs (remote access Trojans), and makes them fully undetectable for antiviruses (FUD). This left us with no doubt that this software was developed to protect malware from discovery by anti-viruses, as the authors didn't forget to emphasize that they "don't take any responsibility for the use" of DarkEyE:

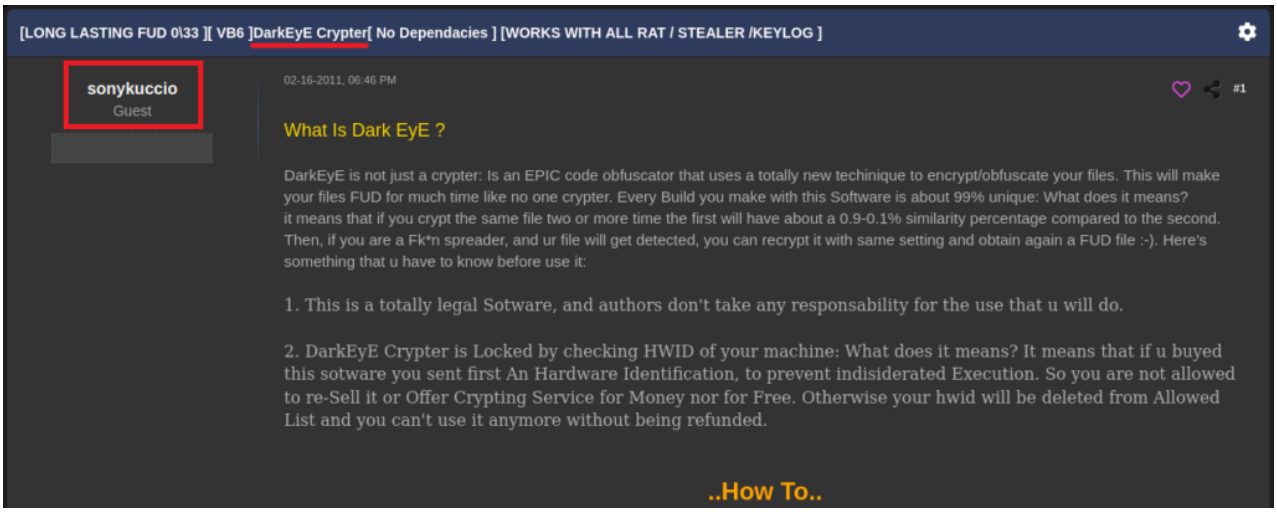


Figure 5 – DarkEyE advertisement on a hacker forum.

The user "**sonykuccio**" also posted contact emails for anyone interested in buying DarkEyE (remember this for later):

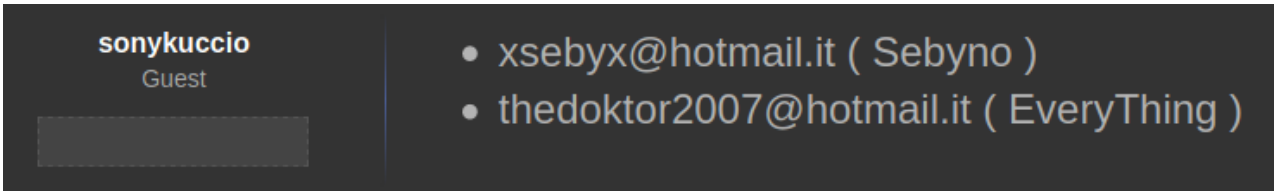


Figure 6 – Contact emails mentioned in DarkEyE ads.

Finally, we found the website **securitycode.eu**, whose URL is mentioned in one of the ads above.

DarkEyE evolved into CloudEyE

Indeed, the website securitycode.eu is connected to DarkEyE. However, currently this website focuses on another product – CloudEyE:

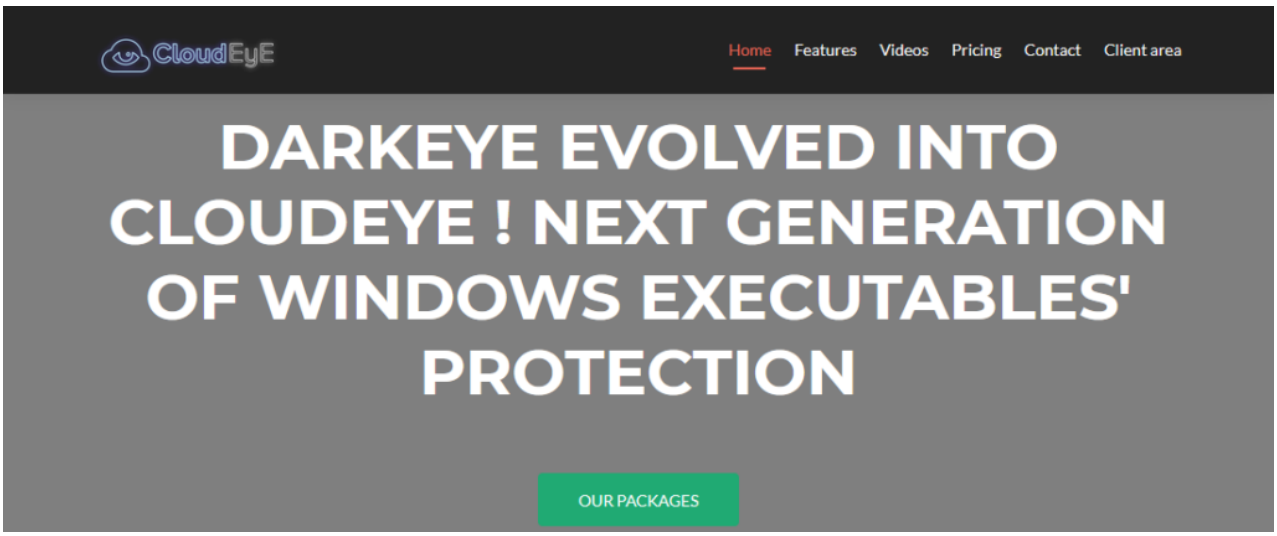


Figure 7 – securiticode.eu website.

The company selling CloudEyE pretends to be legitimate. As said on their website, CloudEyE is security software intended for "*Protecting windows applications from cracking, tampering, debugging, disassembling, dumping*."

But let's look at the rest of the securitycode.eu website. It contains several YouTube video tutorials on how to use CloudEyE, and, as it turned out, how to abuse Google Drive and OneDrive:

- "Protecting an application using google drive." (https://youtu.be/TOdfOBmeAx8)
- "Protecting using an already exsisting project, with a saved profile." (https://youtu.be/8siii5x0Q3k)
- "Protecting file using VPS/Cloud or any dedicated server." (https://youtu.be/4JLEXGevpfg)
- "Protecting file using backup domains." (https://youtu.be/4JJWL4-OCDM)
- "CloudEyE avoiding debugging of application." (https://youtu.be/v1CS_Q7LZpg)

- "Protecting 'putty' application using OneDrive." (https://youtu.be/Y2ZNLVC6yfk)
- "CloudEyE memory protection in action!" (https://youtu.be/76IVgS88WTg)

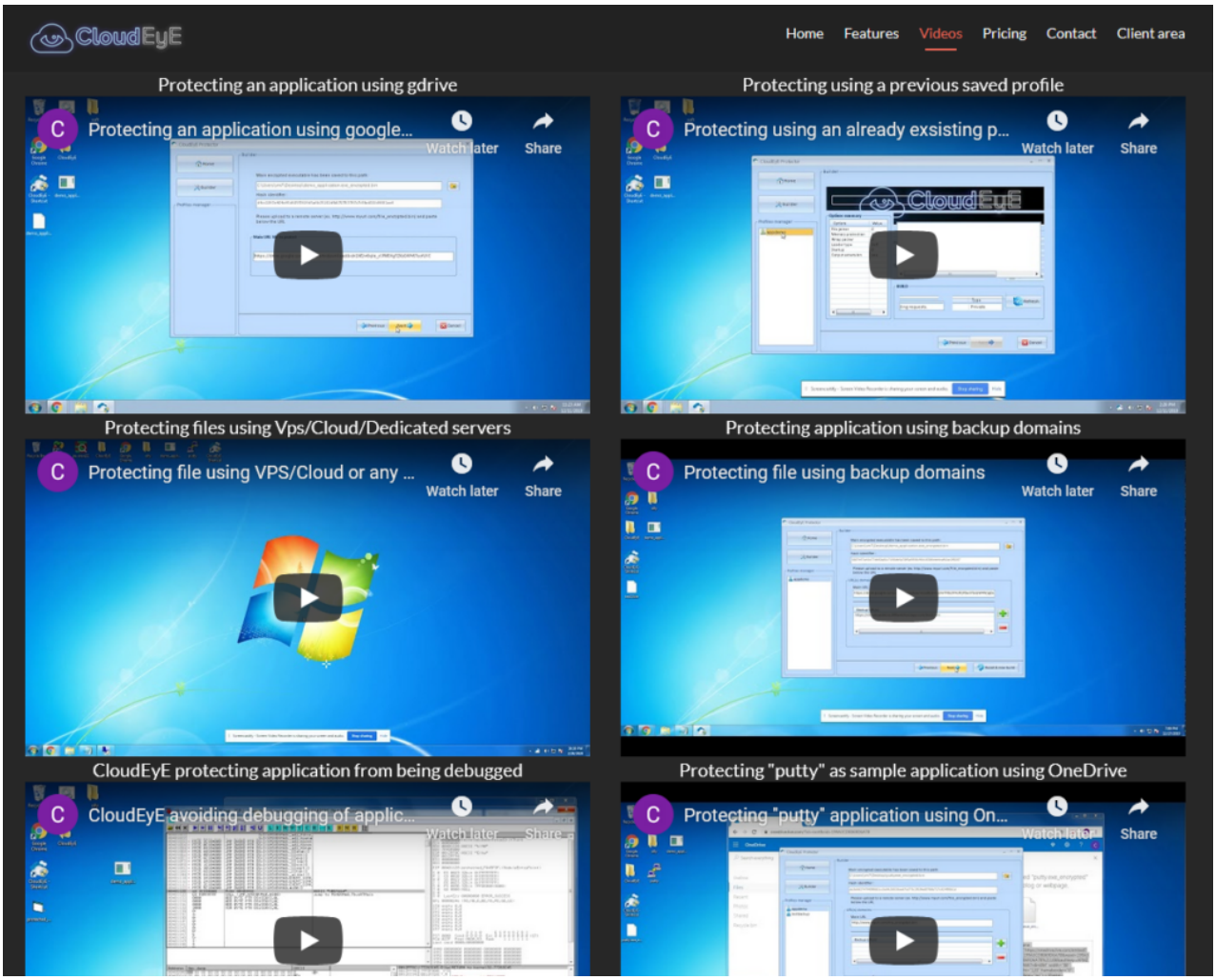


Figure 8 – YouTube videos published on the securitycode.eu website.

Watching one of the videos on this website (https://youtu.be/TOdfOBmeAx8?t=74), we noticed the same URL patterns as we have seen earlier in GuLoader:

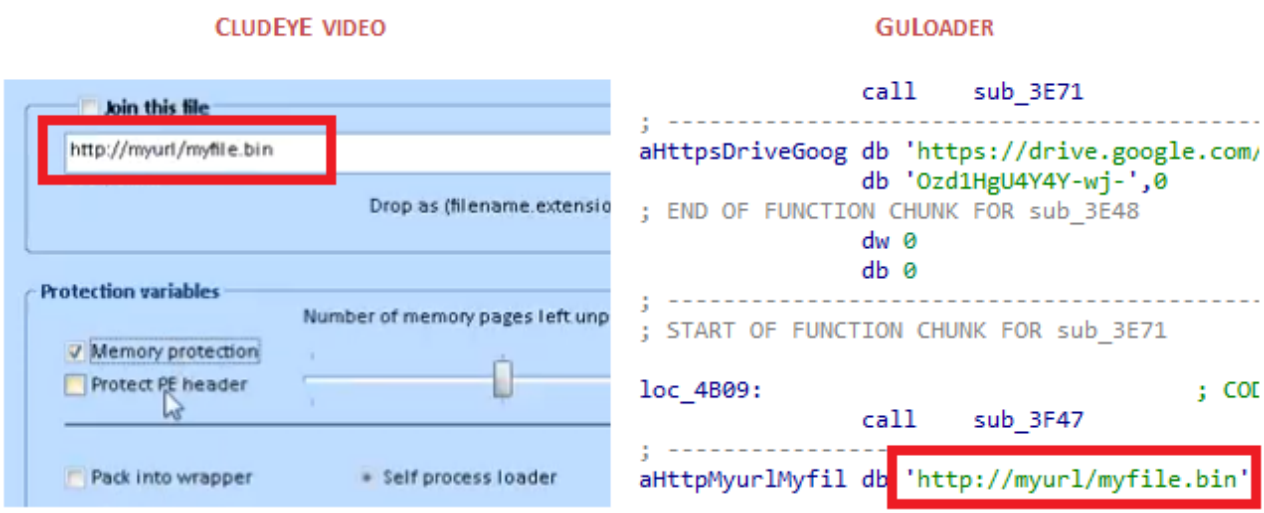


Figure 9 – The same URL pattern in the CloudEyE YouTube video and GuLoader samples.

This is a placeholder for a URL that is used in some of GuLoader samples for downloading joined files (decoy images in our previous research). Way too much coincidence for us to find it here!

We decided to obtain CloudEyE to see for ourselves if it is related to GuLoader.

CloudEyE

To test CloueEyE Protector, we decided to encrypt the **calc.exe** application:

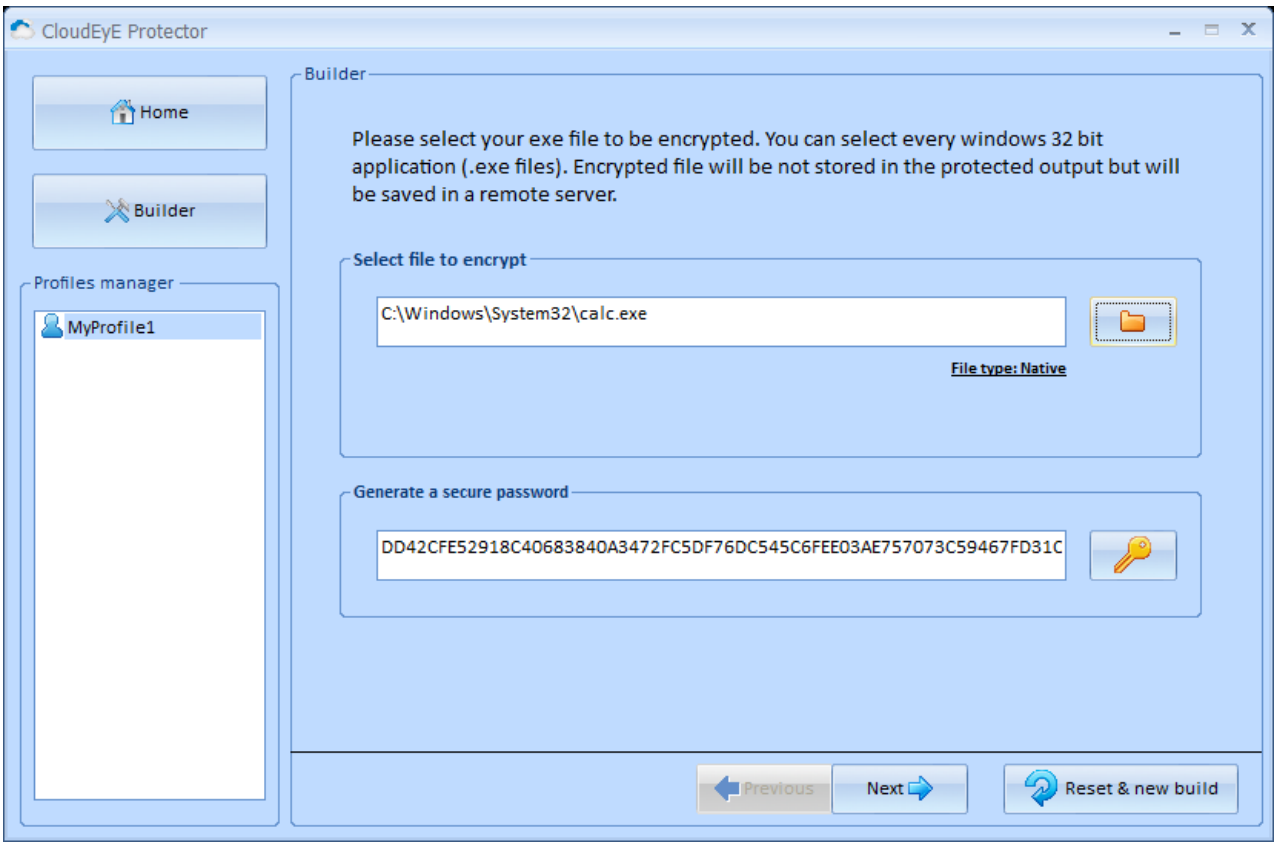


Figure 10 – CloudEyE builder: choosing a file to protect.

The XOR encryption key (password) is generated automatically and can't be entered manually.

After clicking "Next", we got the encrypted file. Then we placed it on a local HTTP server and put the URL in the next window:

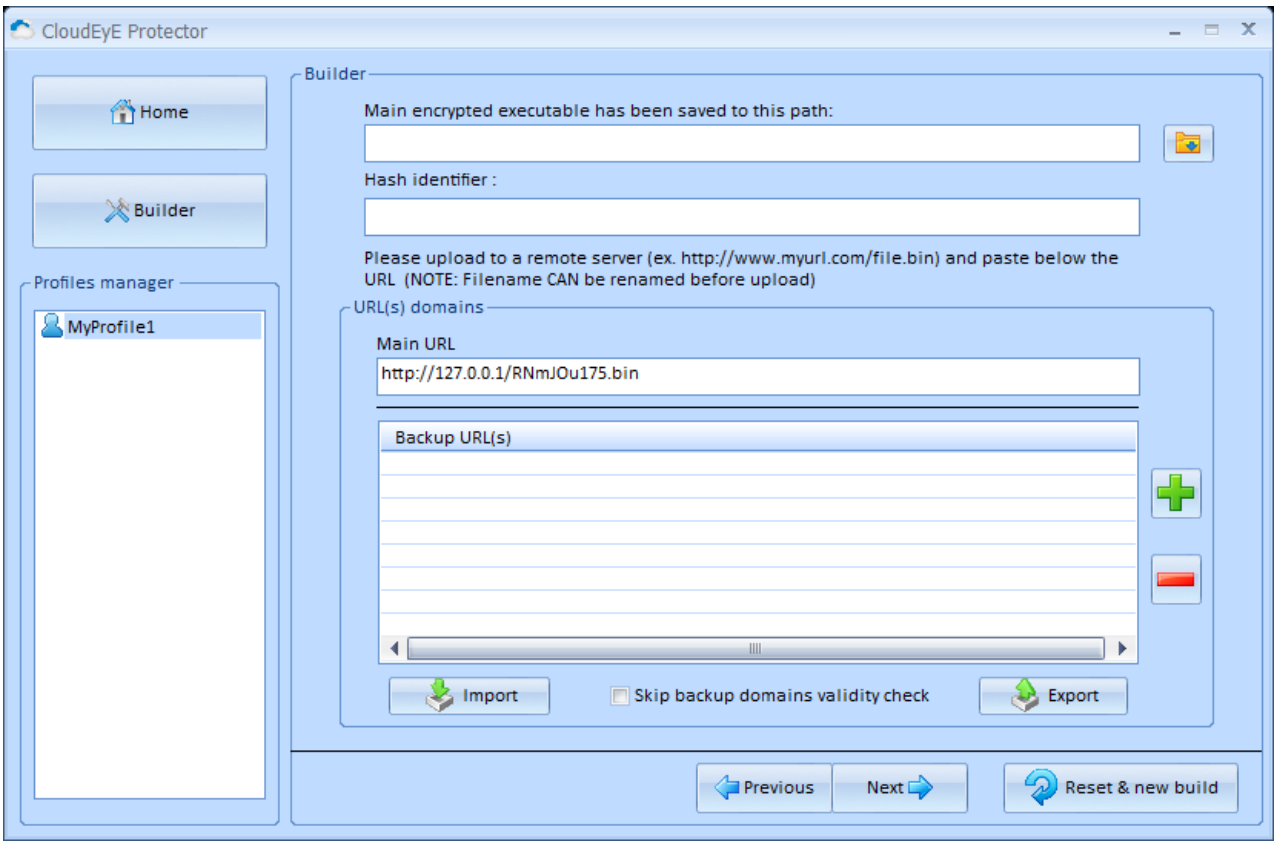


Figure 11 – CloudEyE builder: choosing a URL where the protected file will be downloaded from.

After clicking "Next", we see the window with the known URL template `http://myurl/myfile.bin` :

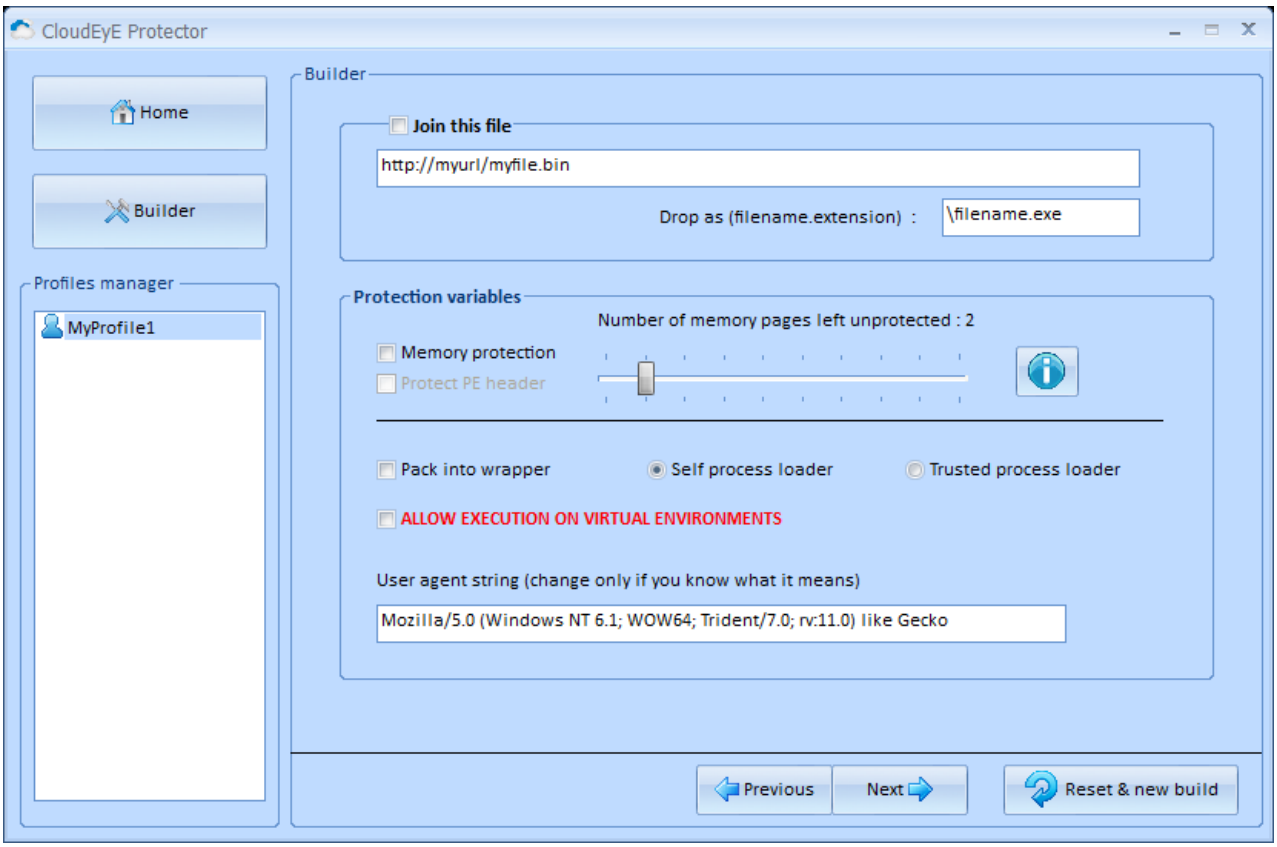


Figure 12 – CloudEyE builder: protection options.

We assumed that most customers don't use additional options, so we decided to leave everything else as the default value.

CloudEyE also allows you to set up autorun, select an icon, change the file size and choose the extension:

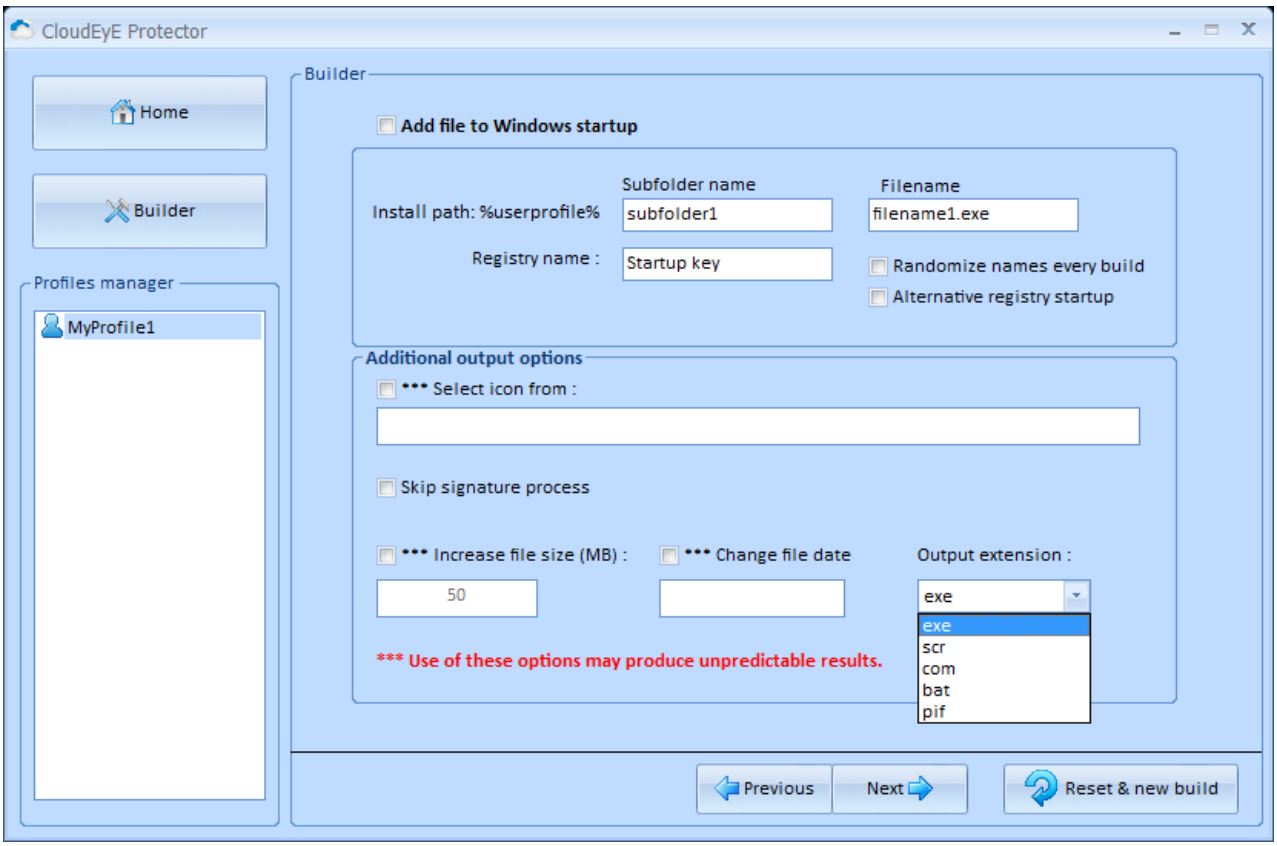
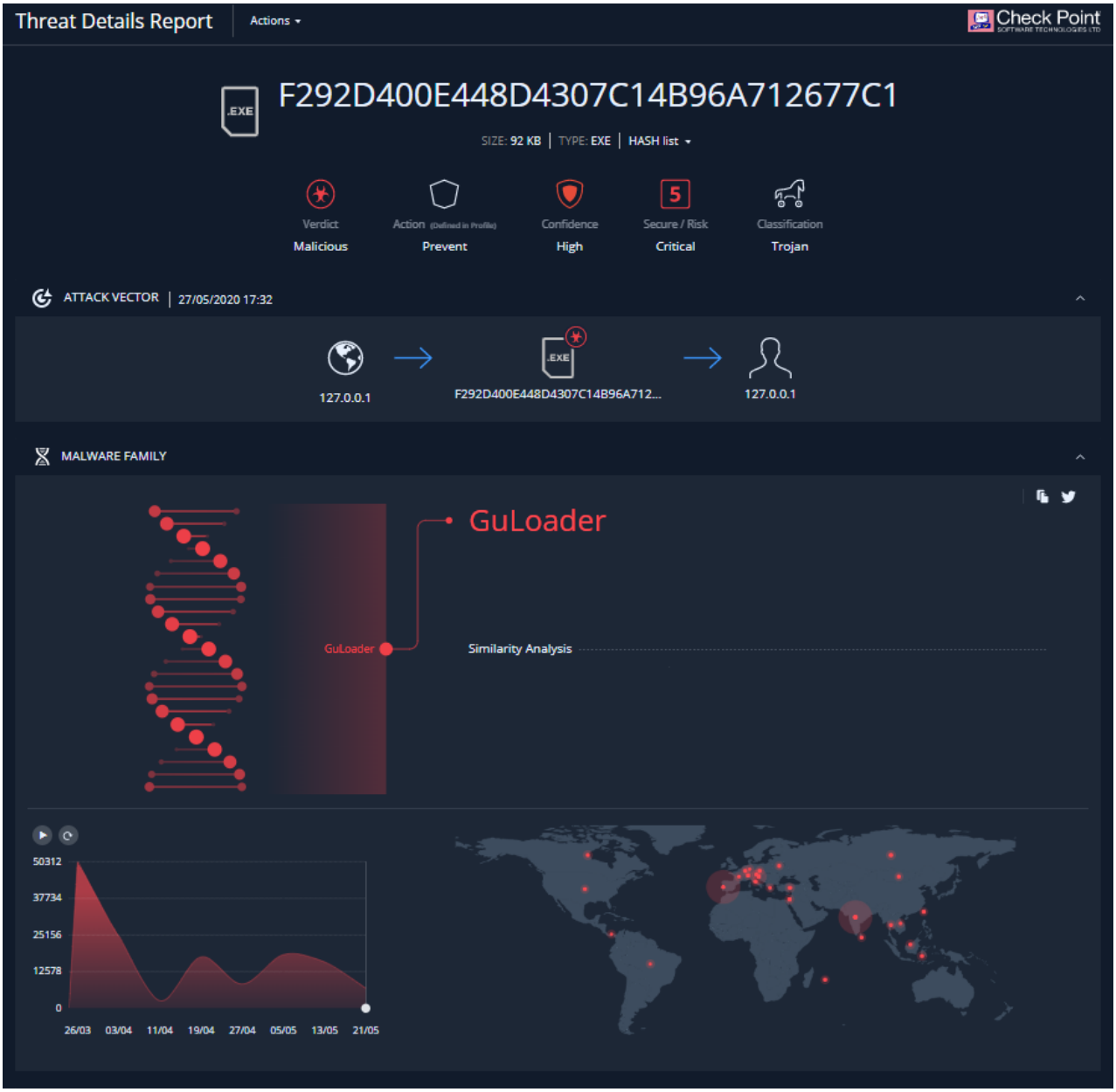
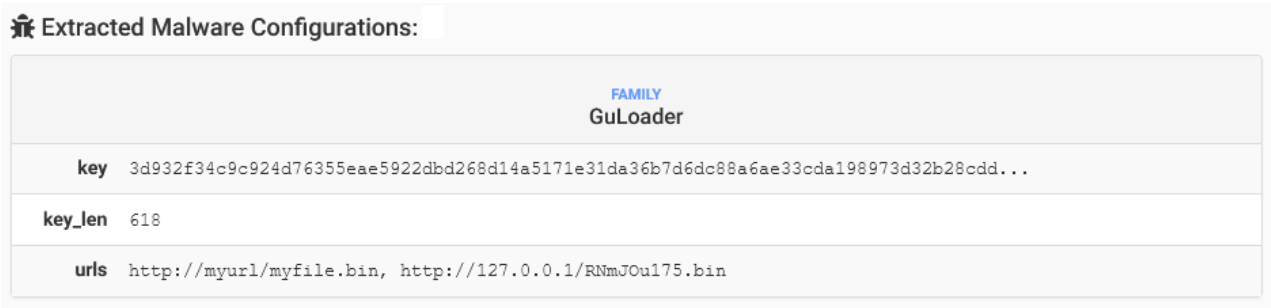


Figure 13 – CloudEyE builder: additional options.

Finally, we got the build.

At the next step, we submitted the build to our sandbox and, unsurprisingly, we got the expected verdict:





Extracted Malware Configurations:

	FAMILY GuLoader
key	3d932f34c9c924d76355eae5922dbd268d14a5171e31da36b7d6dc88a6ae33cda198973d32b28cdd...
key_len	618
urls	http://myurl/myfile.bin, http://127.0.0.1/RNmJOu175.bin

Figure 14 – Emulation results of the CloudEyE-produced sample.

However, to be completely sure that CloudEyE produces samples that are universally acknowledged as GuLoader malware, we decided to analyze it manually and compare with a real GuLoader sample that we saw in the wild.

GuLoader was slightly upgraded a few weeks ago. Therefore, we chose one of the recent samples which downloads the Formbook malware:

GuLoader MD5:	**3d1fd9bcef7cbe915bb49857461ad781**
Payload URL:	hxxps://drive.google.com/uc?export=download&id=1cs40Db_dgZugASem90KebWJ2m-Vl6LmjR
Encrypted Payload MD5:	95f29abac9c887639efc2d4e22b5350f
Decrypted Payload MD5:	3b72bf861b5d2907bb2d76d3d4d9d816

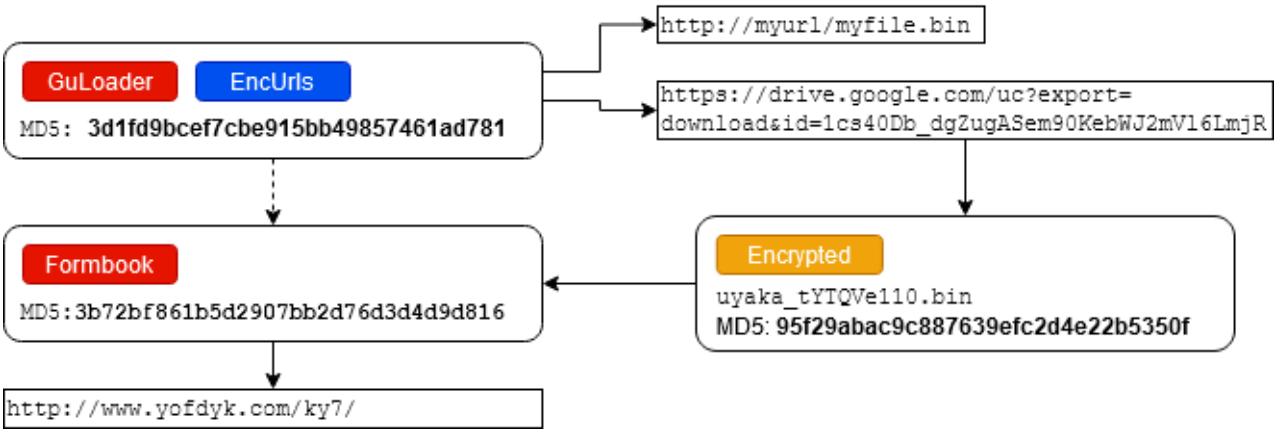


Figure 15 – Researched GuLoader sample details.

The CloudEyE-produced sample that we got has the same structure as GuLoader. Just like GuLoader, it is compiled with Visual Basic and contains shellcode encrypted with a random 4-bytes XOR key. Therefore, we decrypted the shellcode from both samples (CloudEyE and GuLoader).

To make it harder for automatic analysis and probably also to prevent automatic decryption, the shellcode starts from a random stub and is prepended with a jump over this stub. In both samples, the same space on the stack is reserved for a structure with global variables.

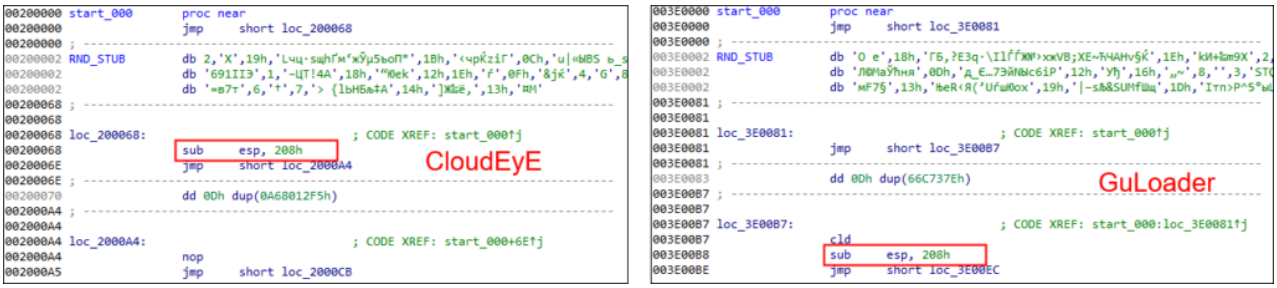


Figure 16 – Comparison of CloudEyE and GuLoader samples: shellcode randomization.

Variables in the structure have the same offset. Most of the code chunks differ only due to the applied randomization techniques. The useful code is the the same in both samples.

```
CloudEyE
ab_decrypt_joined_URL proc near          ; CODE XREF: sub_2012A8:loc_2045B5↓p
                pop     ebx
                push    edi
                clc
                push    ebx
                jmp     short loc_2016B0
; ---------------------------------------------------------------------------
                dd 10h dup(0A68012F5h)
; ---------------------------------------------------------------------------

loc_2016B0:                              ; CODE XREF: ab_decrypt_joined_URL+4↑j
                clc
                call    sub_204AA4
                clc
                clc
                push    dword ptr [ebp+64h] ; XOR Key
                push    618             ; Key length
                push    23              ; URL length
                push    dword ptr [ebp+0B8h] ; URL
                cld
                call    ab_xor_decrypt
                push    0
                jmp     short loc_201702
; ---------------------------------------------------------------------------
                dd 0Ch dup(12F5A680h)
; ---------------------------------------------------------------------------

loc_201702:                              ; CODE XREF: ab_decrypt_joined_URL+66↑j
                cld
                push    dword ptr [ebp+0B8h] ; URL
                nop
                call    sub_203A60
```

```
GuLoader
ab_decrypt_joined_URL proc near          ; CODE XREF: SEG000:loc_3E47BC↓p
                pop     ebx
                push    edi
                push    ebx
                jmp     short loc_3E15E2
; ---------------------------------------------------------------------------
                dd 0Dh dup(6C737E06h)
; ---------------------------------------------------------------------------

loc_3E15E2:                              ; CODE XREF: ab_decrypt_joined_URL+3↑
                fnop
                call    sub_3E4D85
                push    dword ptr [ebp+64h] ; XOR Key
                nop
                push    580             ; Key length
                push    23              ; URL length
                nop
                push    dword ptr [ebp+0B8h] ; URL
                jmp     short loc_3E1625
; ---------------------------------------------------------------------------
                dd 0Ah dup(737E066Ch)
; ---------------------------------------------------------------------------

loc_3E1625:                              ; CODE XREF: ab_decrypt_joined_URL+52
                cld
                call    ab_xor_decrypt
                cld
                push    0
                push    dword ptr [ebp+0B8h] ; URL
                call    sub_3E395C
                push    0
                push    eax
```

Figure 17 – Comparison of CloudEyE and GuLoader samples: URL decryption.

The URLs for downloading the payload and the "joined file" (i.e. the decoy image) in the new version of GuLoader are stored encrypted. GuLoader decrypts the URLs using the same key as used for decrypting the payload. After extracting the XOR keys, we can easily find and decrypt URLs in both samples.

```
CloudEyE
call    ab_load_C2_URL
db 55h                  ; ^3D= 'h'
db 0E7h ; з             ; ^93= 't'
db  5Bh ; [             ; ^2F= 't'
db  44h ; D             ; ^34= 'p'
db 0F3h                 ; ^C9= ':'
db 0E6h ; ж             ; ^C9= '/'
db  0Bh                 ; ^24= '/'
db 0E6h ; ж             ; ^D7= '1'
db 51h                  ; ^63= '2'
db  62h ; b             ; ^55= '7'
db 0C4h ; Д             ; ^EA= '.'
db 0D5h ; X             ; ^E5= '0'
db 0BCh                 ; ^92= '.'
db  1Dh                 ; ^2D= '0'
db  93h ; “             ; ^BD= '.'
db  17h                 ; ^26= '1'
db 0A2h                 ; ^8D= '/'
db  46h ; F             ; ^14= 'R'
db 0EBh ; л             ; ^A5= 'N'
db  7Ah ; z             ; ^17= 'm'
db  54h ; T             ; ^1E= 'J'
db  7Eh ; ~             ; ^31= 'O'
db 0AFh ; Ї             ; ^DA= 'u'
db    7                 ; ^36= '1'
db  80h ; Ъ             ; ^B7= '7'
db 0E3h ; г             ; ^D6= '5'
db 0F2h ; т             ; ^DC= '.'
db 0EAh ; к             ; ^88= 'b'
db 0CFh ; П             ; ^A6= 'i'
db 0C0h ; А             ; ^AE= 'n'
```

```
CloudEyE
call    ab_load_joined_URL
db 55h  ; ^3D= 'h'
db 0E7h ; ^93= 't'
db  5Bh ; ^2F= 't'
db  44h ; ^34= 'p'
db 0F3h ; ^C9= ':'
db 0E6h ; ^C9= '/'
db  0Bh ; ^24= '/'
db 0BAh ; ^D7= 'm'
db  1Ah ; ^63= 'y'
db  20h ; ^55= 'u'
db  98h ; ^EA= 'r'
db  89h ; ^E5= 'l'
db 0BDh ; ^92= '/'
db  40h ; ^2D= 'm'
db 0C4h ; ^BD= 'y'
db  40h ; ^26= 'f'
db 0E4h ; ^8D= 'i'
db  78h ; ^14= 'l'
db 0C0h ; ^A5= 'e'
db  39h ; ^17= '.'
db  7Ch ; ^1E= 'b'
db  58h ; ^31= 'i'
db 0B4h ; ^DA= 'n'
```

```
GuLoader
call    ab_load_C2_URL
db 0F2h                 ; ^9A= 'h'
db  5Ch ; \             ; ^28= 't'
db  57h ; W             ; ^23= 't'
db  0Dh                 ; ^7D= 'p'
db  14h                 ; ^67= 's'
db  33h ; 3             ; ^09= ':'
db 0EAh ; к             ; ^C5= '/'
db  0Bh                 ; ^24= '/'
db 0C5h ; E             ; ^A1= 'd'
db 0CCh ; M             ; ^BE= 'r'
db 0BEh ; s             ; ^D7= 'i'
db  17h                 ; ^61= 'v'
db  69h ; i             ; ^0C= 'e'
db  17h                 ; ^39= '.'
db 0BCh ; j             ; ^DB= 'g'
db 0C1h ; Б             ; ^AE= 'o'
db  8Ah ; Љ             ; ^E5= 'o'
db  2Eh ; .             ; ^49= 'g'
db  63h ; c             ; ^0F= 'l'
db 0A5h ; Ґ             ; ^C0= 'e'
db 0C0h ; A             ; ^EE= '.'
db  4Dh ; M             ; ^2E= 'c'
db 0DEh ; Ю             ; ^B1= 'o'
db    1                 ; ^6C= 'm'
db 0C3h ; Г             ; ^EC= '/'
db 0A9h ; ©             ; ^DC= 'u'
db  27h ; '             ; ^44= 'c'
db 0DBh ; Ы             ; ^E4= '?'
db 0F2h ; т             ; ^97= 'e'
db  9Ah ; Љ             ; ^E2= 'x'
```

```
GuLoader
call    ab_load_joined_URL
db 0F2h ; ^9A= 'h'
db  5Ch ; ^28= 't'
db  57h ; ^23= 't'
db  0Dh ; ^7D= 'p'
db  5Dh ; ^67= ':'
db  26h ; ^09= '/'
db 0EAh ; ^C5= '/'
db  49h ; ^24= 'm'
db 0D8h ; ^A1= 'y'
db 0CBh ; ^BE= 'u'
db 0A5h ; ^D7= 'r'
db  0Dh ; ^61= 'l'
db  23h ; ^0C= '/'
db  54h ; ^39= 'm'
db 0A2h ; ^DB= 'y'
db 0C8h ; ^AE= 'f'
db  8Ch ; ^E5= 'i'
db  25h ; ^49= 'l'
db  6Ah ; ^0F= 'e'
db 0EEh ; ^C0= '.'
db  8Ch ; ^EE= 'b'
db  47h ; ^2E= 'i'
db 0DFh ; ^B1= 'n'
```

Figure 18 – Comparison of CloudEyE and GuLoader encrypted URLs.

We can therefore conclude that the samples are almost identical and differ only generally due to applied code randomization techniques.

Identities behind CloudEyE

Let's refer to the contact emails posted by the user "**sonykuccio**" in the DarkEyE ads:

- xsebyx@hotmail.it (Sebyno)
- thedoktor2007@hotmail.it (EveryThing)

We looked for the emails and usernames in publically available leaked email databases and managed to find several entries related to "**sonykuccio**":

Figure 19 – Emails and usernames found in publically available databases.

Also, we surprisingly found a PDF containing a lot of real names and emails of Italian citizens, including the email "xsebyx@hotmail.it" and the corresponding name "Sebastiano Dragna":

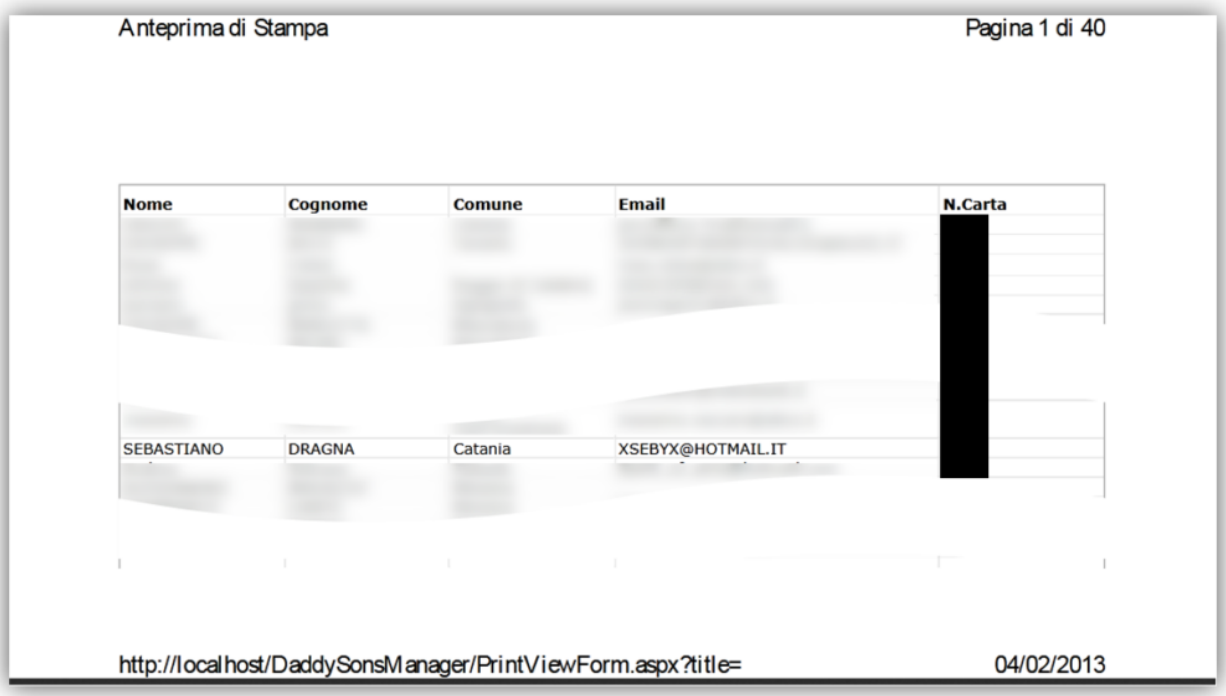


Figure 20 – A PDF with emails of Italian citizens.

Let's now refer to the Privacy Policy section on the website securitycode.eu. We see the same name! The owners of this business must sincerely believe in their own innocence if they dare to publish real names on the website:

Figure 21 – securitycode.eu privacy policy.

Therefore, "**sonykyccio**", "**xsebyx**", "**Sebyno**", "**decrypter@hotmail.it**", "**xsebyx@hotmail.it**", "**sonykuccio@gmail.com**" are avatars and emails of the same person: **Dragna Sebastiano Fabio**.

Unfortunately, we didn't manage to find any relation between another name published on the website (**Ivano Mancini**) and names used on popular hacker forums.

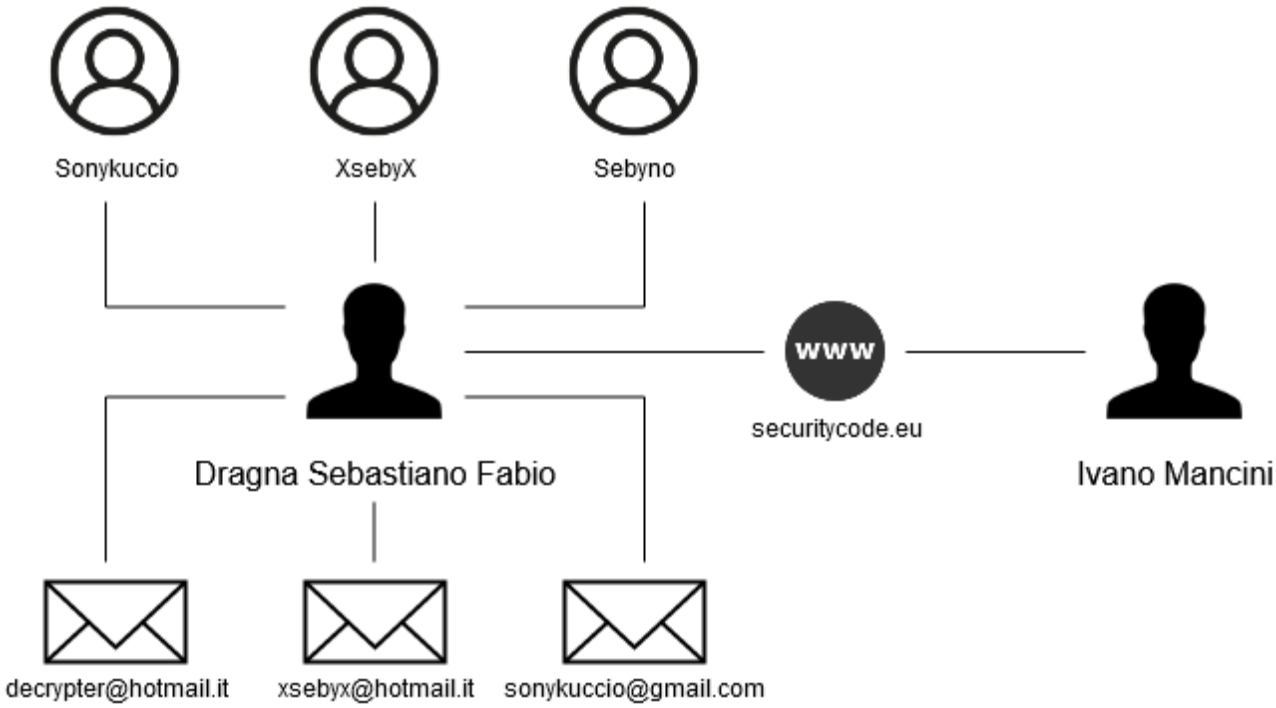


Figure 22 – Identities behind CloudEyE.

Sonykuccio is an old and established visitor to hacker forums. We saw that he started selling DarkEyE in the beginning of 2011. But even before creating DarkEyE protector, Sonykuccio was already providing services for protecting malware against anti-viruses

(FUD service) and a spreading service for malware:

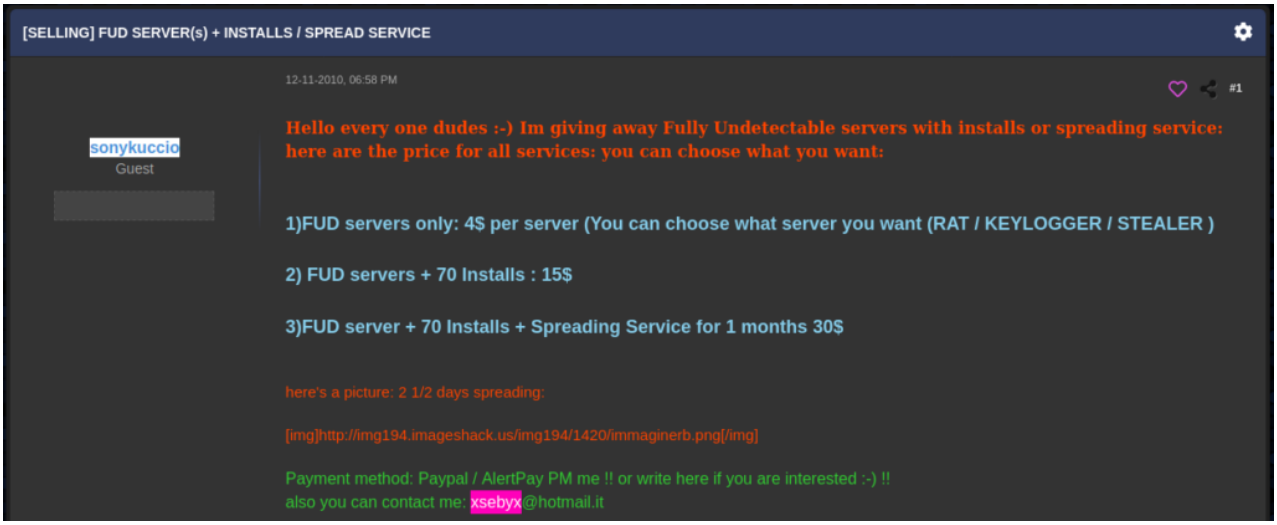


Figure 23 – Malicious services advertised by sonykuccio.

CloudEyE and Covid-19

As we said, we see hundreds of attacks every day in different campaigns. Some of the CloudEyE users have been cynically using the name "Coronavirus" as a way to deceive and mislead victims, using the fear and desire for information about the pandemic to infect people with malware.

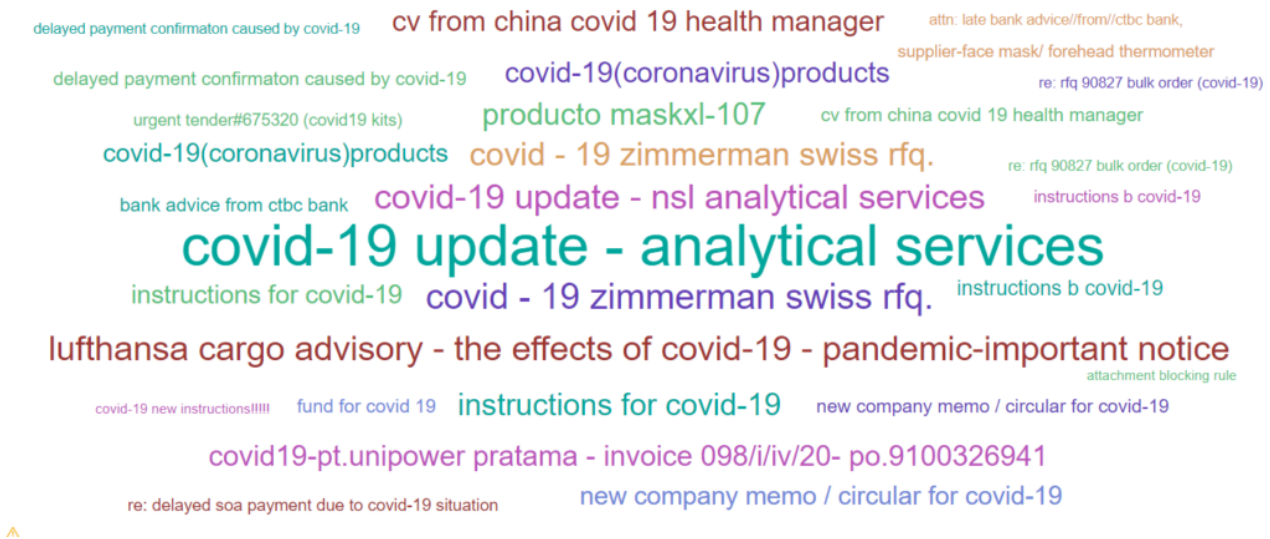


Figure 24 – CloudEyE and Coronavirus email subjects.

Revenue

The securitycode.eu website claims that their customer base numbers over 5,000. As they sell their basic package for $ 100 per month, this allows us to estimate their monthly income at $ 500,000.

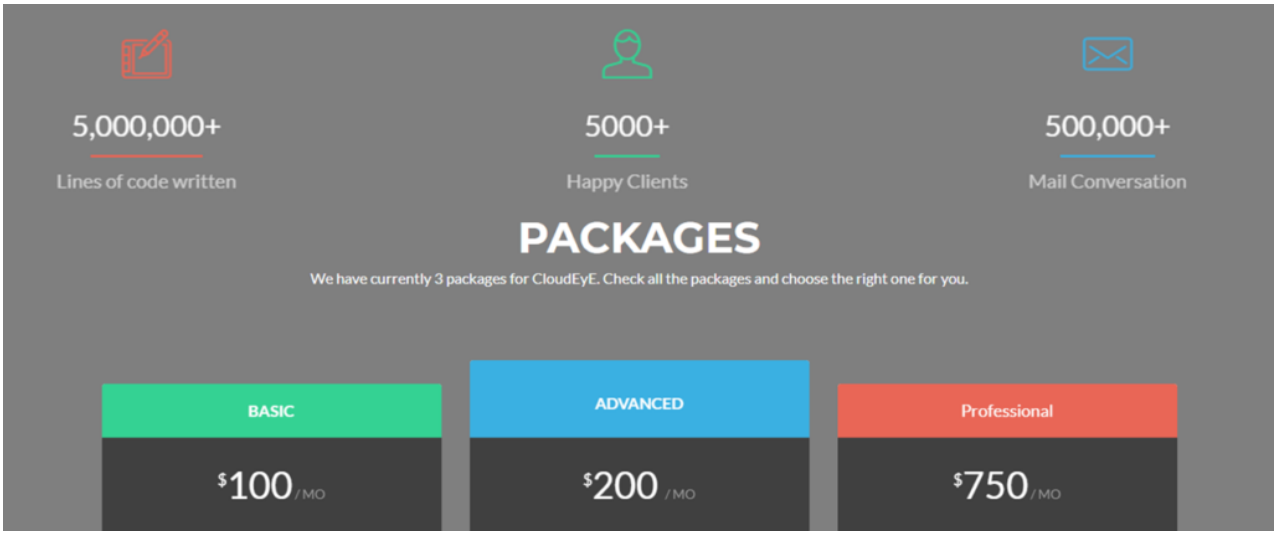


Figure 25 – CloudEyE pricing.

Conclusion

CloudEyE operations may look legal, but the service provided by CloudEyE has been a common denominator in thousands of attacks over the past year. Tutorials published on the CloudEyE website show how to store payloads on cloud drives such as Google Drive and OneDrive. Cloud drives usually perform anti-virus checking and technically don't allow the upload of malware. However, payload encryption implemented in CloudEyE helps to bypass this limitation. Code randomization, evasion techniques, and payload encryption used in CloudEyE protect malware from being detected by many of the existing security products on the market. Surprisingly, such a service is provided by a legally registered Italian company that operates a publically available website which has existed for more than four years.

Many of CloudEyE customers are threat actors with no deep technical knowledge, they are using publically available malware or leaked hacking tools for stealing passwords, credentials, private information, and gaining control of the victim's environment.

Appendix: Hashes of samples

Description	MD5
Researched GuLoader sample	3d1fd9bcef7cbe915bb49857461ad781
Encrypted GuLoader payload (Formbook)	95f29abac9c887639efc2d4e22b5350f
Formbook sample dropped by GuLoader	3b72bf861b5d2907b-b2d76d3d4d9d816
GuLoader Shellcode	0284062f9a7415e413ed319c13dc0988
CloudEyE Shellcode	5c4ed372836487562aa22ab9cd2798d9

Check Point Threat Emulation provides protection against this threat:

- *Dropper.Win.CloudEyE.A*
- *Dropper.Wins.CloudEyE.B*
- *Dropper.Win.CloudEyE.I*
- *Dropper.Win.CloudEyE.gl.J*
- *Dropper.Win.CloudEyE.gl.L*