MP5

Checkpoint 1

5.1 Reverse engineering

What is reverse engineering?

Reverse engineering is the process of of extracting knowledge or design information from a product. In MP5 context, the product is a binary kernel module.

Why reverse engineering?

- Malware analysis
    - W32.Stuxnet Dossier
- Bug fixing
    - Did Microsoft Just Manually Patch Their Equation Editor Executable? (CVE-2017-11882)
- Military or commercial espionage

How to reverse a binary program?

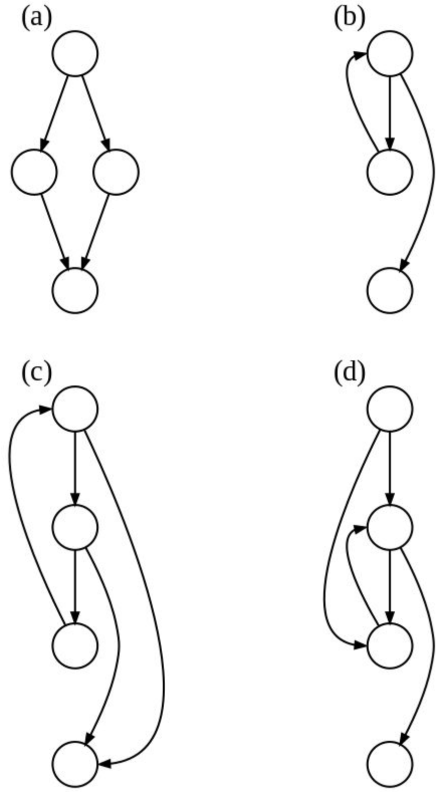
- Understand the operating system / network environment that the binary runs
- Binary analysis
  - Binary header, e.g., PE header (Win32), ELF header (Linux)
  - Symbol tables including sections, function names
  - Import tables and linked libraries
  - Strings and constants embedded in the file, e.g., “hard-coded” password or secret keys
- Disassembly
  - Read raw machine language such as assembly code
- Decompilation
  - Reconstruct the program at a higher level, e.g, near source code level
  - Control flow analysis
- Tools of the trade: https://github.com/fdivrp/awesome-reversing

Control Flow Graph

A directed graph that shows all paths that might be traversed through a program during its execution.

An important tool to understand program executions, especially jumps and loops

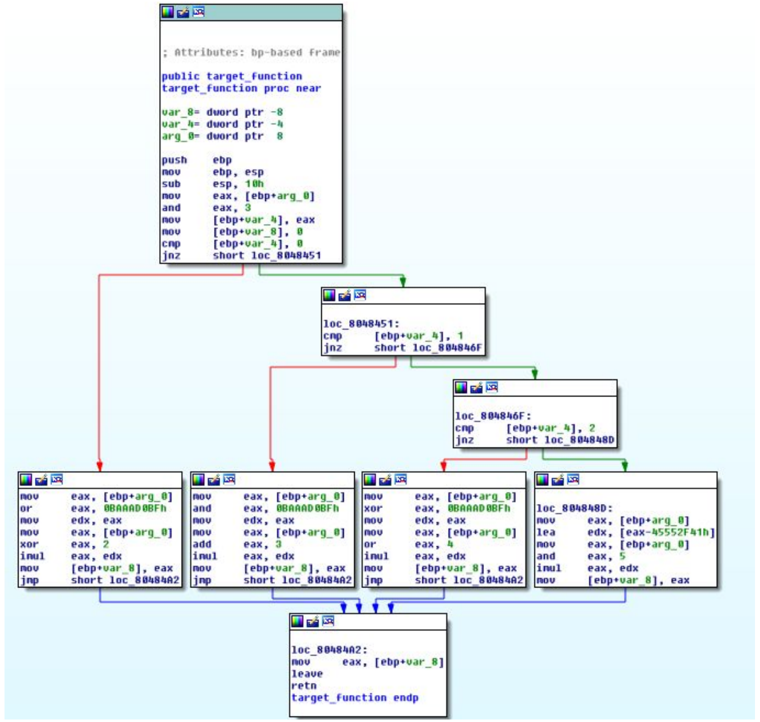
Tools are available to visualize CFG from assembly code, e.g., IDA, radare2, hopper



(a) an if-then-else
(b) a while loop
(c) a natural loop with two exits, e.g. while with an if...break in the middle; non-structured but reducible
(d) an irreducible CFG: a loop with two entry points, e.g. goto into a while or for loop

A control flow graph of an example function in IDA

```c
unsigned int target_function(unsigned int n)
{
  unsigned int mod = n % 4;
  unsigned int result = 0;

  if (mod == 0) result = (n | 0xBAAAD0BF) * (2 ^ n);

  else if (mod == 1) result = (n & 0xBAAAD0BF) * (3 + n);

  else if (mod == 2) result = (n ^ 0xBAAAD0BF) * (4 | n);

  else result = (n + 0xBAAAD0BF) * (5 & n);

  return result;
}
```



```
; Attributes: bp-based frame

public target_function
target_function proc near

var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, [ebp+arg_0]
and     eax, 3
mov     [ebp+var_4], eax
mov     [ebp+var_8], 0
cmp     [ebp+var_4], 0
jnz     short loc_8048451
```

```
loc_8048451:
cmp     [ebp+var_4], 1
jnz     short loc_804846F
```

```
loc_804846F:
cmp     [ebp+var_4], 2
jnz     short loc_804848D
```

```
mov     eax, [ebp+arg_0]
or      eax, 0BAAAD0BFh
mov     edx, eax
mov     eax, [ebp+arg_0]
xor     eax, 2
imul    eax, edx
mov     [ebp+var_8], eax
jmp     short loc_80484A2
```

```
mov     eax, [ebp+arg_0]
and     eax, 0BAAAD0BFh
mov     edx, eax
mov     eax, [ebp+arg_0]
add     eax, 3
imul    eax, edx
mov     [ebp+var_8], eax
jmp     short loc_80484A2
```

```
mov     eax, [ebp+arg_0]
xor     eax, 0BAAAD0BFh
mov     edx, eax
mov     eax, [ebp+arg_0]
or      eax, 4
imul    eax, edx
mov     [ebp+var_8], eax
jmp     short loc_80484A2
```

```
loc_804848D:
mov     eax, [ebp+arg_0]
lea     edx, [eax-45552F41h]
mov     eax, [ebp+arg_0]
and     eax, 5
imul    eax, edx
mov     [ebp+var_8], eax
```

```
loc_80484A2:
mov     eax, [ebp+var_8]
leave
retn
target_function endp
```

5.2 Object deserialization attack

Serialization is the process of translating an object on memory of a running program into a data structure that can be stored, e.g., on a file on disk.

The serialized object can be transmitted, e.g., via a network or read from disk, and later deserialized by another program to get a clone of the original object.

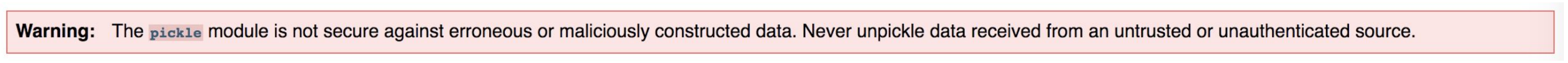
Malformed data or unexpected data could be used to abuse application logic, deny service, or execute arbitrary code, when deserialized.

Affected languages: C, C++, Java, Python, Ruby (and others)

Python pickle: a case study of object deserialization attack

Pickle is a format used by Python to serialize Python objects.

Python pickle is **insecure** by design.


Warning: The `pickle` module is not secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.

A pickle file is a sequence of instructions that will be executed by a Virtual Machine when deserializing an object.

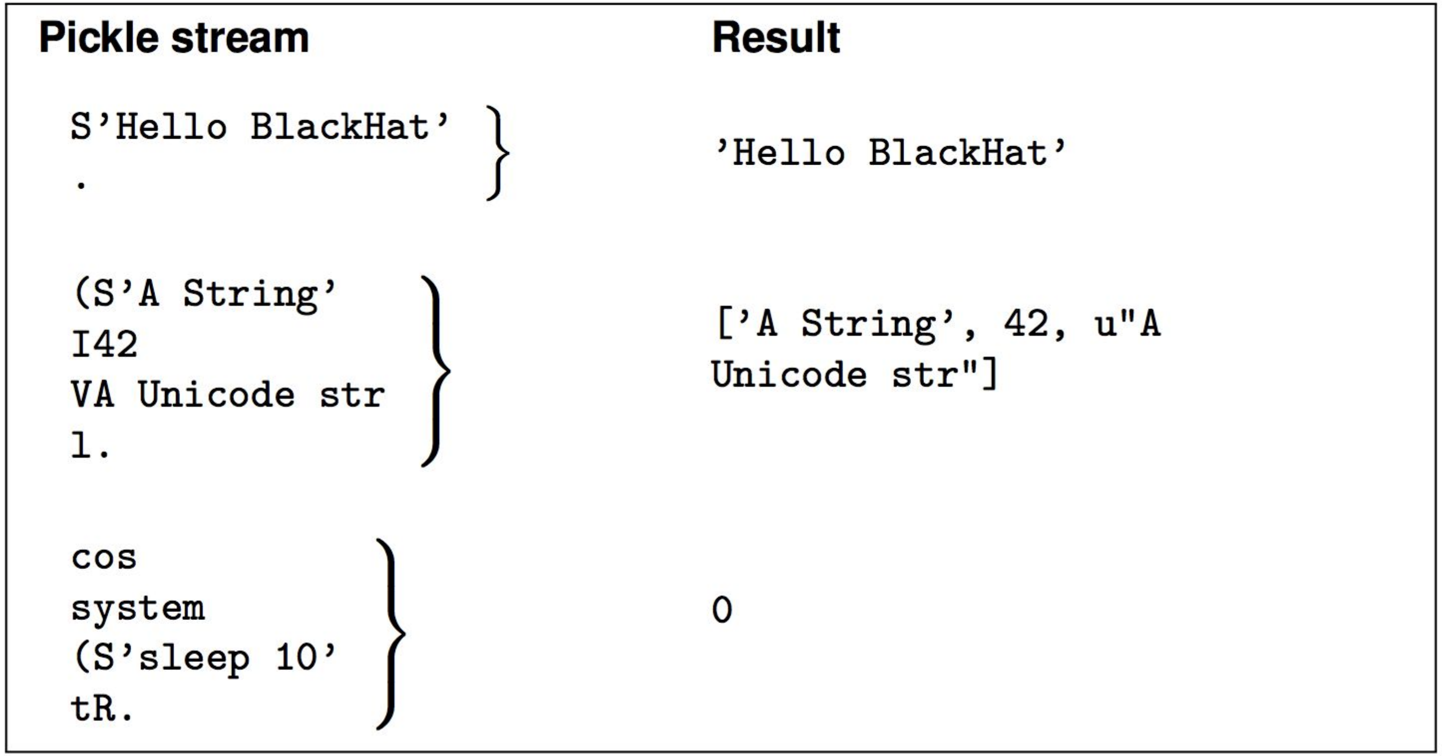
Instruction engine Reads and executes instructions from the pickle stream.

Stack Regular stack structure for scratch space, implemented as a Python list.

Memo Indexed array that functions as registers, implemented as a Python list. The convention ‘memo[i]’ is used to refer to memo slots.

https://media.blackhat.com/bh-us-11/Slaviero/BH_US_11_Slaviero_Sour_Pickles_WP.pdf

Example pickle files and output



Pickle stream	Result

`S'Hello BlackHat'`
`.`
	`'Hello BlackHat'`

`(S'A String'`
`I42`
`VA Unicode str`
`l.`
	`['A String', 42, u"A Unicode str"]`

`cos`
`system`
`(S'sleep 10'`
`tR.`
	`0`

An attacker sends a malicious pickle to a vulnerable server

malicious.pickle

```
@app.route("/upload")

def upload():

    try:

        for file in request.files:

            blob = request.files.get(str(file)).read()

            pickle.loads(blob)

            return "Pickle file loaded!"

    except Exception as ex:

        return "Error loading pickle!"
```

$\boxed{\text{Victim calls } \texttt{dumps()}} \Rightarrow \textit{attacker plays} \Rightarrow \boxed{\text{Victim calls } \texttt{loads()}}$

5.3 Double withdrawal attack exploiting race condition

Time of check to time of use (TOCTOU) is a class of software bugs caused by changes in a system between the checking of a condition (such as a security credential) and the use of the results of that check. The bugs were not obvious to software developers.

Victim	Attacker

`if (access("file", W_OK) != 0) {`
`    exit(1);`
`}`
 	
`fd = open("file", O_WRONLY);`
`// Actually writing over /etc/passwd`
`write(fd, buffer, sizeof(buffer));`
	`//`
	`//`
	`// After the access check`
	`symlink("/etc/passwd", "file");`
	`// Before the open, "file" points to the password database`
	`//`
	`//`

In this example, an attacker can exploit the race condition between the access and open to trick the setuid victim into overwriting an entry in the system password database. TOCTTOU races can be used for privilege escalation, to get administrative access to a machine.

Exploiting a TOCTTOU race condition

It requires precise timing to ensure that the attacker's operations interleave properly with the victim's. If TOCTOU time window is small, it is more difficult to exploit.

In our MP, the digital wallet first:

1. Check for your balance
2. Process the withdraw
3. Update the new balance

How can you increase the TOCTOU time windows and inject another withdraw in the step 2?

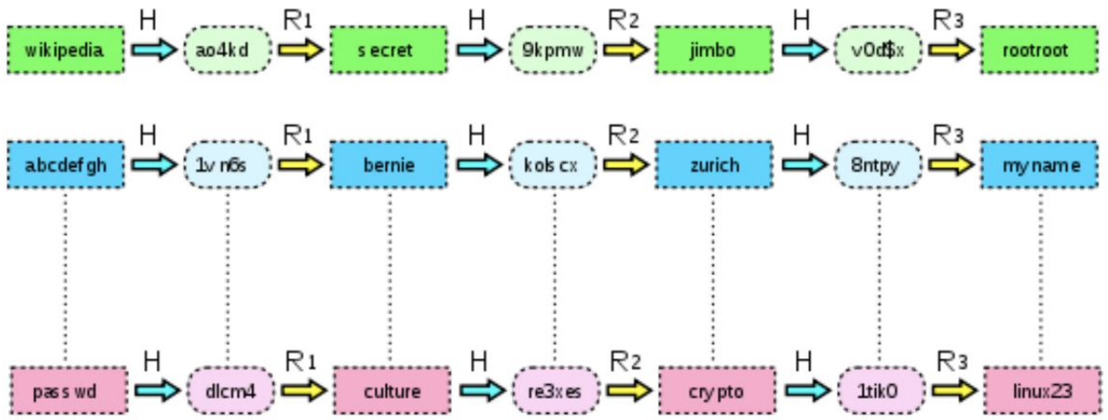
5.4 Passwords cracking

Brute force search through all possible passwords in order

- Use a dictionary
- Use a dictionary of common passwords
- Combine dictionary with common passwords and heuristics (e.g. p@$$w0rd and password123)
- Use statistical models of user passwords (e.g. Markov models)
- Easy to parallelize: hash password guess, compare to entire hash database
- Commonly done with arrays of GPUs

Rainbow Tables

- Many passwords are common
- Precompute them in a lookup table
- Time/space tradeoff



Quick Primer to Hashcat

Fastest password cracker
Can utilize GPU / CPU (legacy support)
Latest version 4.0.1 - available directly from website

Commands
-a - Attack mode (straight (0), combination (1), bruteforce (3), etc.)
-m - Type of hash to use (100 for SHA1)
-o - Specifies output file, results are also stored in hashcat.potfile
-r - Define a rule list to use
Charsets
?l - abcdefghijklmnopqrstuvwxyz
?u - ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d - 0123456789
?s - !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

Quick Primer to Hashcat

Example command:

```
    hashcat64 -m 100 -a 3 5.1.4.hashes ?a?a?a?a?a?a
```

What does the command above do?

More advanced methods discussed next week!