MICHELANGELO

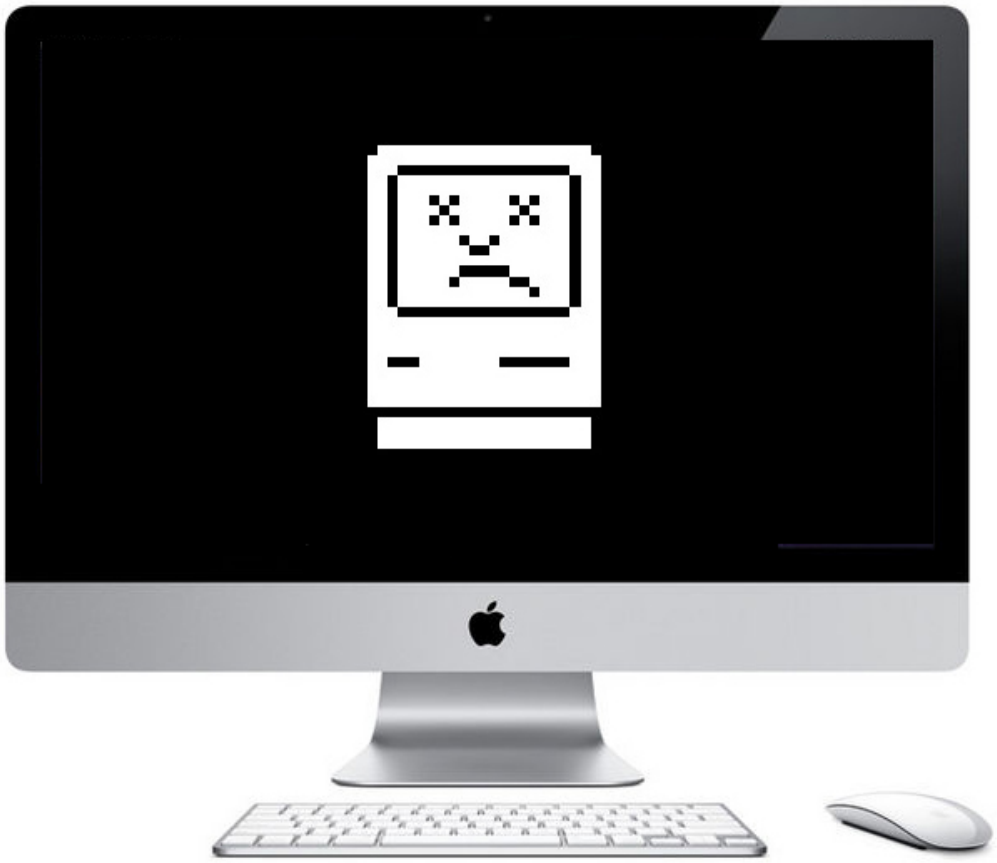
STONED

JERUSALEM

CIH

Melissa

DOWNAD Conficker



TREND MICRO

Mac Binary Analysis: A Sn3ak Peek

Christopher Daniel So
Threat Analyst

TREND MICRO
Securing Your Journey to the Cloud

AGENDA

- Mac Malware Incidents
- Mac Executable Files
- C Language
- Objective-C
- Q & A

AGENDA

- At the end of the presentation, we should be familiar on some techniques on how to do static analysis on Mac binaries

Scope and Limitations

- In this presentation, we will talk about:
  - Background info
  - x86 code
  - Calling convention for Intel Macs
- We will **not** talk about:
  - Analyzing PowerPC code
  - In-depth discussion on Cocoa
  - 64-bit Intel Macs
  - Scripting

WARNING!!!

This is a technical presentation.

Mac Malware Incidents

TREND MICRO

Mac Malware Incidents

OSX_MACKONTROL.A
(MaControl)

OSX_IMULER.B and
OSX_IMULER.C
(Imuler)

OSX_COINMINE.A
(DevilRobber)

Mac Executable Files

TREND MICRO

Concepts and Terms

- Static analysis
- Endianness
- Magic number
- Calling convention
- Object-oriented programming

Mach-O

- Can contain the following:
  - Executable code
  - Object code
  - Shared libraries
  - Core dumps
- Only supports 1 architecture
- Magic number is 0xFEEDFACE (32-bit) or 0xFEEDFACF (64-bit)
- Position-independent code

Universal Binary

- A bundle of Mach-O files for different platforms
- Magic number is 0xCAFEBABE (big endian)
- Used to ease the transition from PowerPC Macs to Intel Macs and to support both 32-bit and 64-bit Macs

C Language

C Language

- Mac OS X primarily uses the C calling convention
- Parameters are placed in the stack, right to left, and the caller cleans the stack after the subroutine returns
- Stack must be 16 byte aligned

C TO ASSEMBLY

Windows - Visual C++ 6.0

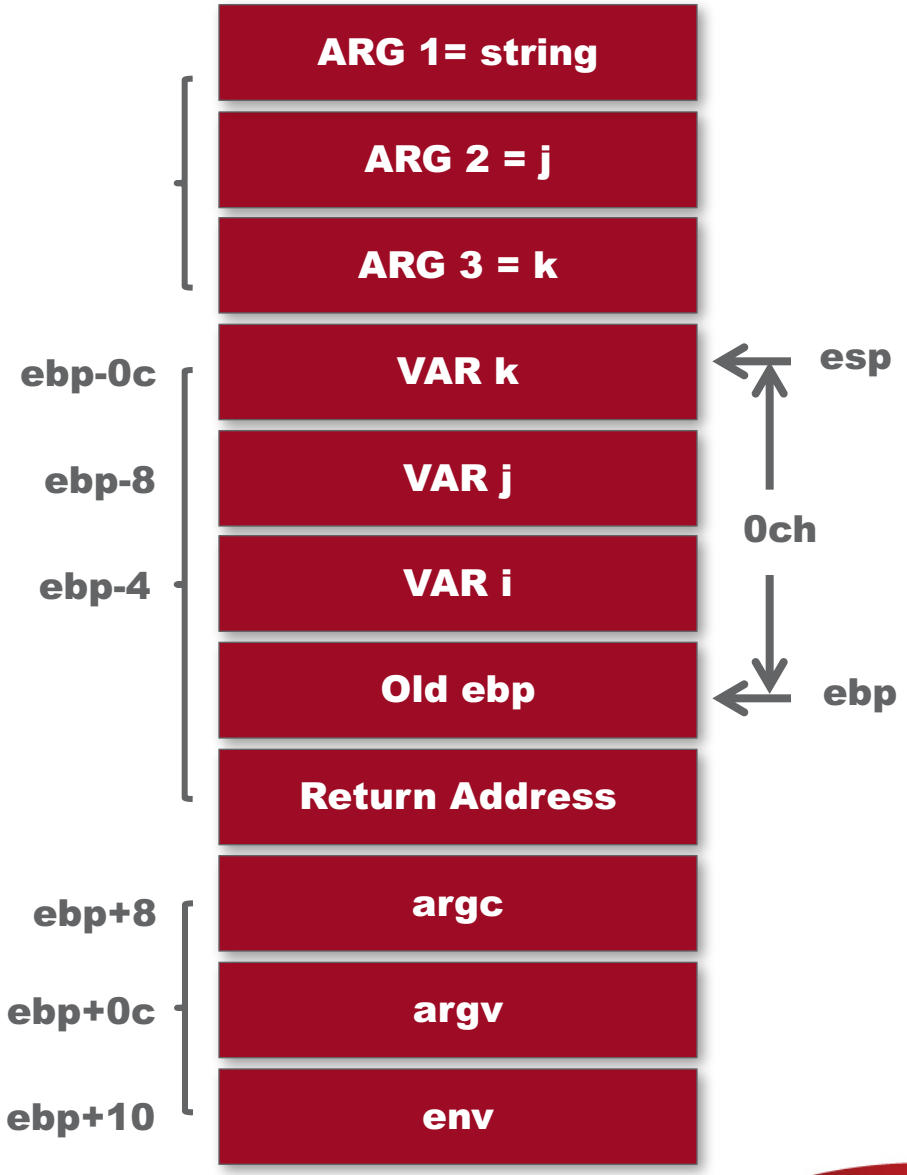
```c
#include <stdio.h>
int main(int argc, char argv[])
{
  int i, j, k;
  i = argc;
  j = i * 2;
  k = i + j;
  printf("Hello %d %d\n", j, k);
  return 0;
}
```

```
push ebp
mov  ebp, esp
sub  esp, 0ch
mov  eax, dword ptr [ebp+8]
mov  dword ptr [ebp-4], eax
mov  ecx, dword ptr [ebp-4]
shl  ecx, 1
mov  dword ptr [ebp-8], ecx
mov  edx, dword ptr [ebp-4]
add  edx, dword ptr [ebp-8]
mov  dword ptr [ebp-0ch], edx
mov  eax, dword ptr [ebp-0ch]
push eax
mov  ecx, dword ptr [ebp-8]
push ecx
push offset HelloStr
call  _printf
add  esp, 0ch
xor  eax, eax
mov  esp, ebp
pop  ebp
ret
```

C TO ASSEMBLY

Windows - Visual C++ 6.0

```c
#include <stdio.h>
int main(int argc, char argv[])
{
  int i, j, k;
  i = argc;
  j = i * 2;
  k = i + j;
  printf("Hello %d %d\n", j, k);
  return 0;
}
```



C TO ASSEMBLY

Mac OS X – gcc 4.2

```c
#include <stdio.h>
int main(int argc, char argv[])
{
  int i, j, k;
  i = argc;
  j = i * 2;
  k = i + j;
```

```
push   %ebp
mov    %esp,%ebp
push   %esi
sub    $0x34,%esp
call   0x1f1c <main+12>
pop    %eax
mov    0xc(%ebp),%ecx
mov    0x8(%ebp),%edx
mov    %edx,-0x8(%ebp)
mov    %ecx,-0xc(%ebp)
mov    -0x8(%ebp),%ecx
mov    %ecx,-0x18(%ebp)
mov    -0x18(%ebp),%ecx
imul   $0x2,%ecx,%ecx
mov    %ecx,-0x1c(%ebp)
mov    -0x18(%ebp),%ecx
mov    -0x1c(%ebp),%edx
add    %edx,%ecx
mov    %ecx,-0x20(%ebp)
```

C TO ASSEMBLY

Mac OS X – gcc 4.2

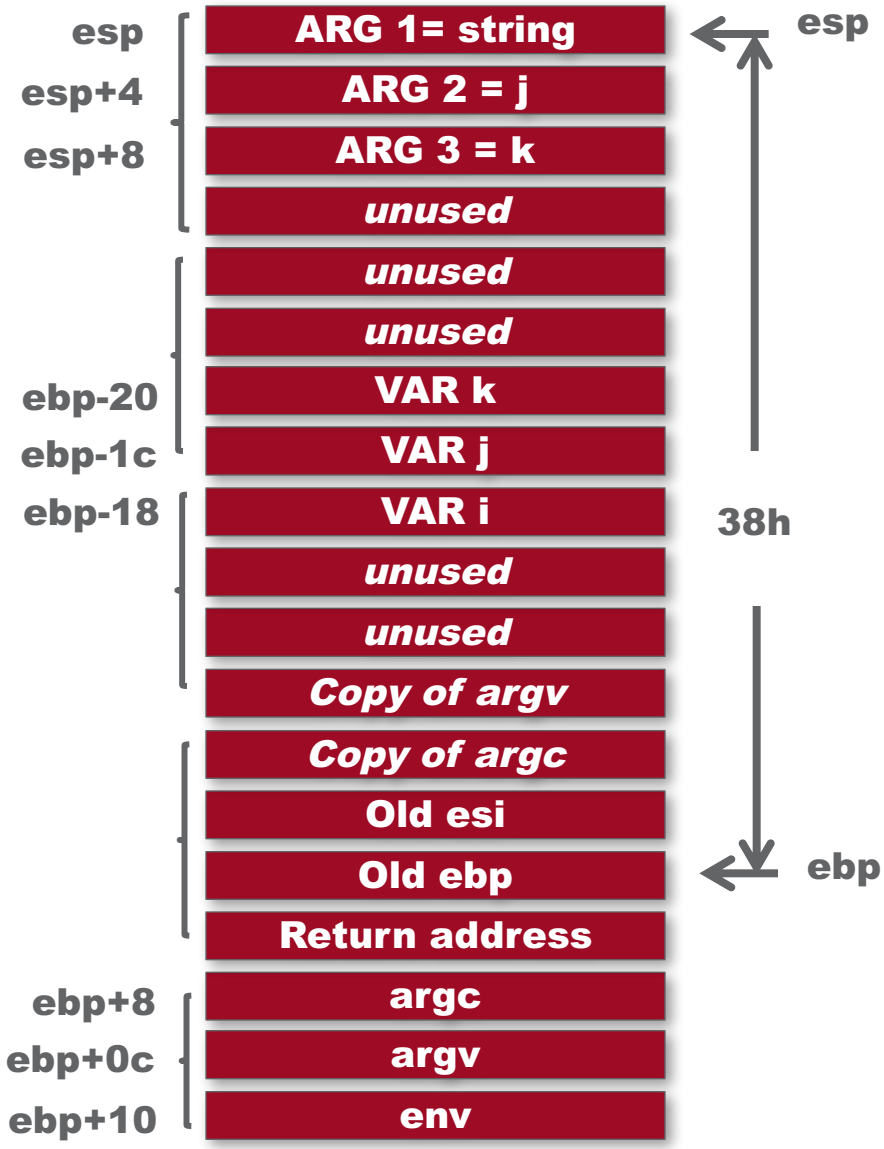
```c
#include <stdio.h>
int main(int argc, char argv[])
{
  int i, j, k;
  i = argc;
  j = i * 2;
  k = i + j;
  printf("Hello %d %d\n", j, k);
  return 0;
}
```

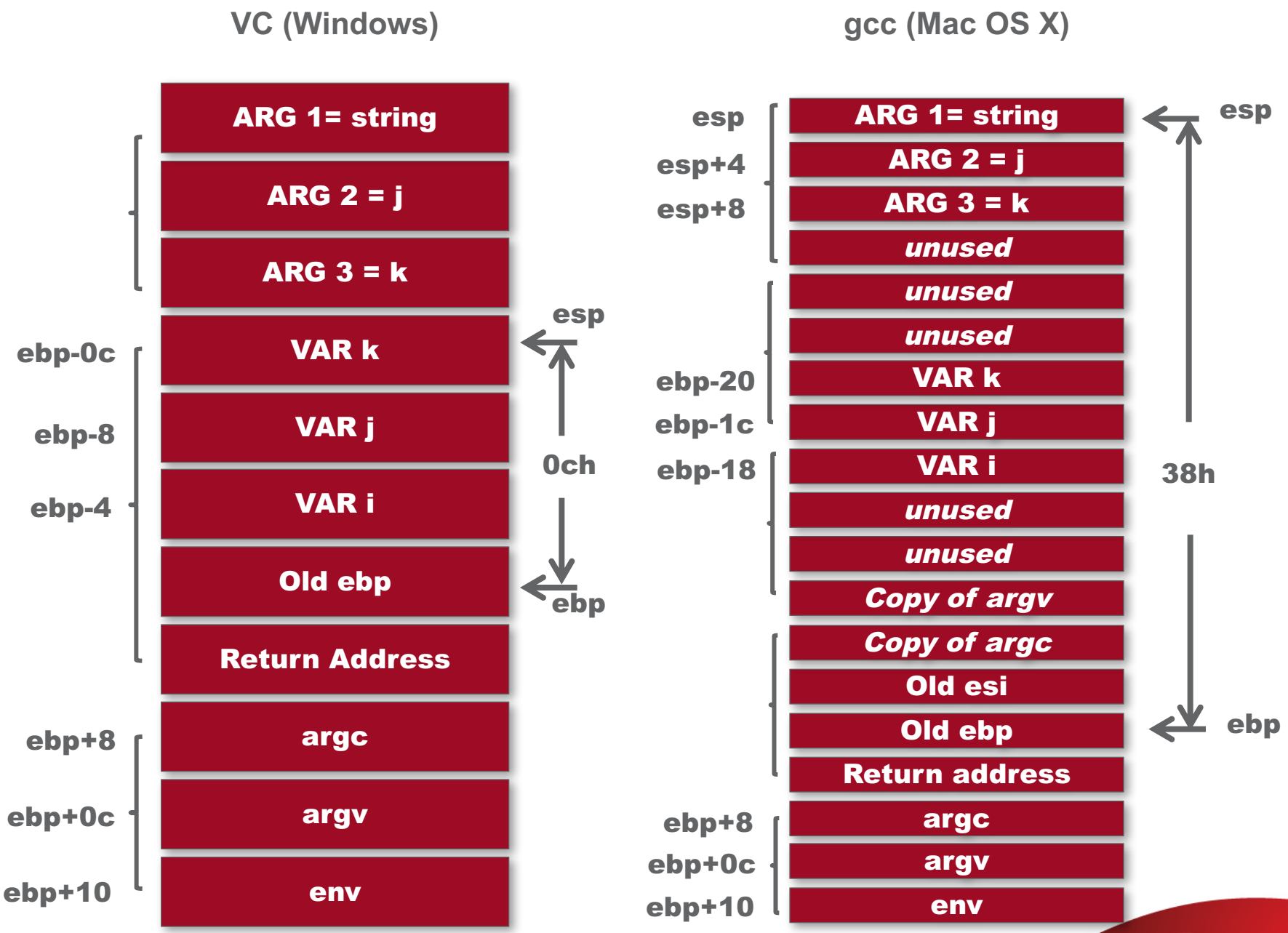
```
mov     -0x1c(%ebp),%ecx
mov     -0x20(%ebp),%edx
mov     %esp,%esi
mov     %edx,0x8(%esi)
mov     %ecx,0x4(%esi)
lea     0x84(%eax),%eax
mov     %eax,(%esi)
call    0x1f7a <dyld_stub_printf>
movl    $0x0,-0x14(%ebp)
mov     -0x14(%ebp),%eax
mov     %eax,-0x10(%ebp)
mov     -0x10(%ebp),%eax
add     $0x34,%esp
pop     %esi
pop     %ebp
ret
```

C TO ASSEMBLY

Mac OS X – gcc 4.2

```c
#include <stdio.h>
int main(int argc, char argv[])
{
  int i, j, k;
  i = argc;
  j = i * 2;
  k = i + j;
  printf("Hello %d %d\n", j, k);
  return 0;
}
```




VC (Windows)
ARG 1= string
ARG 2 = j
ARG 3 = k
ebp-0c
VAR k
ebp-8
VAR j
ebp-4
VAR i
Old ebp
Return Address
ebp+8
argc
ebp+0c
argv
ebp+10
env
esp
0ch
ebp
gcc (Mac OS X)
esp
ARG 1= string
esp+4
ARG 2 = j
esp+8
ARG 3 = k
unused
unused
unused
ebp-20
VAR k
ebp-1c
VAR j
ebp-18
VAR i
unused
unused
Copy of argv
Copy of argc
Old esi
Old ebp
Return address
ebp+8
argc
ebp+0c
argv
ebp+10
env
esp
38h
ebp

Points to Remember

- **[esp+*n*]**

  parameters to be passed

- **[ebp+*n*]**

  parameters passed

- **[ebp-*n*]**

  local variables

Objective-C

Let’s refresh your memory of C++!

- Classes
- Instance
- Properties
- Method
- Compile-time binding

Let's refresh your memory of C++!

```
/* Instantiate */
Door *FrontDoor = new Door();

/* Assign the properties */
FrontDoor->Color = Brown;
FrontDoor->Width = 120;
FrontDoor->Height = 360;

/* Call the methods */
FrontDoor->Open();
FrontDoor->Eat("Sandwich");
```

Compile-time error!

Methods:

- Open
- Close
- Paint
- Destroy

Objective-C

- Created by **Brad Cox** and **Tom Love** in 1983
- Strict superset of C
- Object syntax from Smalltalk

Dr. Brad Cox and Dr. Tom Love

Objective-C

- Used in the **Cocoa API**
  - Used in Mac OS X
  - Objective-C based
  - NeXTSTEP
  - Class named begin with the acronym ‘NS’

Cocoa API

Time for Objective-C

- Classes **(same)**
- Instance **(same)**
- Properties **(same)**
- Method
- Dynamic binding

Methods:

Open
Close
Paint
Destroy

Time for Objective-C

```
/* Instantiate */
Door *FrontDoor = [[Door alloc] init];

/* Assign the properties */
FrontDoor.Color = BROWN;
FrontDoor.Width = 120;
FrontDoor.Height = 360;
/* Property assignment
   translates to something like [FrontDoor
  setHeight:360] */

/* Call the methods */
[FrontDoor Open];
[FrontDoor Eat:"Sandwich"];
```

Methods:

- Open
- Close
- Paint
- Destroy

No compile-time error!

TREND MICRO

A comparison…

C++

```
obj->method(arg);
```

```
push arg
mov  ecx, obj
call Object::method
```

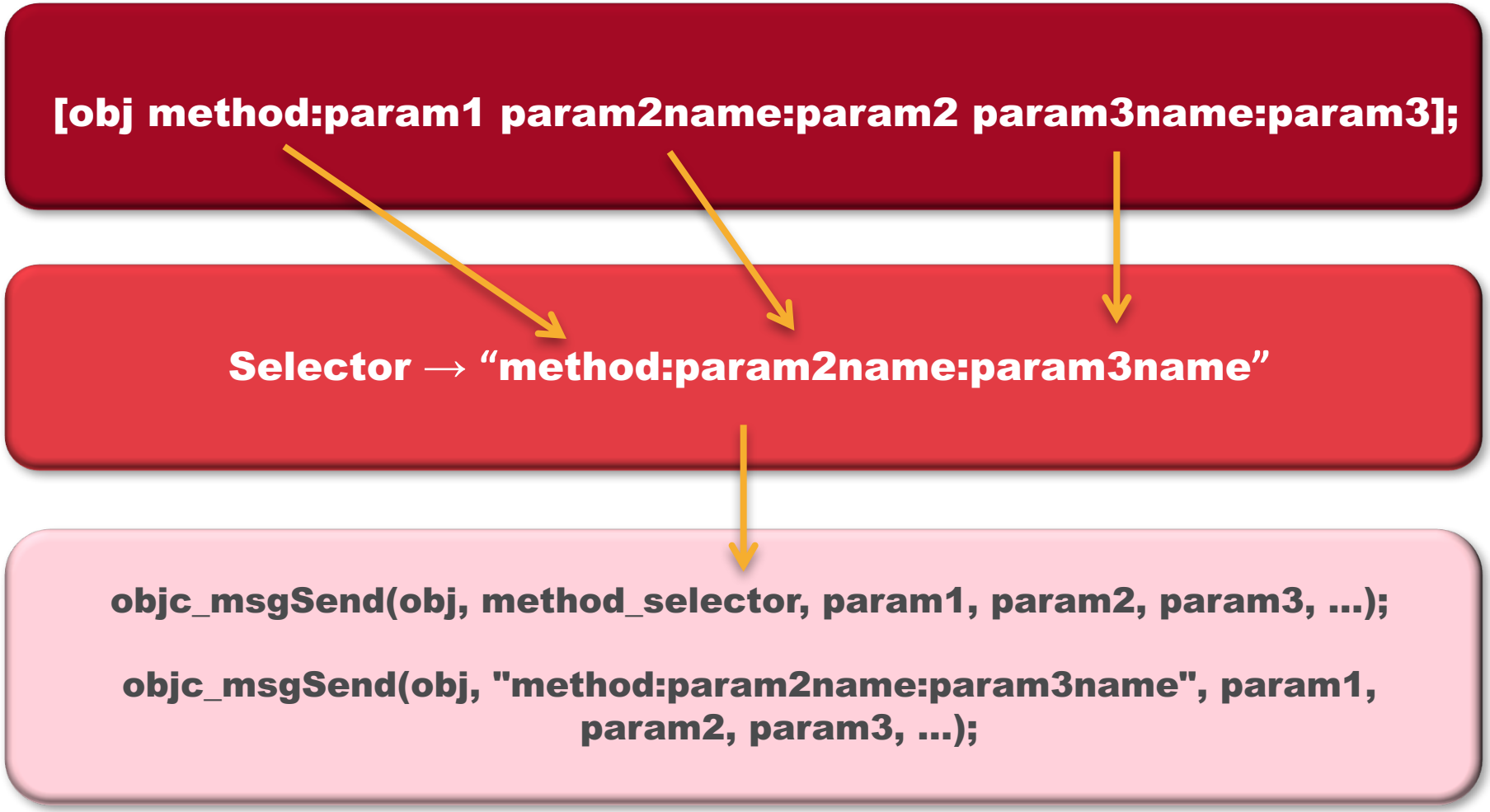
Objective-C

```
[obj method:arg];
```

```
objc_msgSend(obj,"method:",
       arg);
```

```
mov  [esp], obj
mov  [esp+4], strSelector
mov  [esp+8], arg
call _objc_msgSend
```

More on methods



Demo

Recap

TREND MICRO

Recap

- The Universal Binary magic number is 0xCAFEBABE
- Mach-O files have the magic number 0xFEEDFACE or 0xFEEDFACF
- Calling convention is C
- `[esp+n]`- parameters to be passed
- `[ebp+n]`- parameters passed
- `[ebp-n]`- local variables
- Message expressions are converted to objc_msgSend calls.
- You can tell doors to eat!

References

- Apple Developer Connection (Mac OS X)
  - https://developer.apple.com/technologies/mac/
- Mac OS X Developer Library
  - https://developer.apple.com/library/mac/navigation/

That's all folks!

Any questions?

TREND MICRO