How To Avoid Implement An Exploit Friendly JIT

Yunhai Zhang

twitter: @_f0rgetting_

weibo: @f0rgetting

Who am I

- Yunhai Zhang
- Researcher of NSFOCUS Security Team
- Focus on Exploit Detection and Prevention
- Winner of Mitigation Bypass Bounty: 2014-2016

Agenda

- Motivations
- Previous Work
- Issue 1: RWX Page
- Issue 2: Bad Neighbor
- Issue 3: Unsecure Default Behavior
- Conclusion

Motivations

- Why talk about JIT?
  - JIT is widely used in modern software
    - All major web browsers use JIT
  - What JIT do is just similar to what exploit need
    - Generate some code dynamically and execute it

Previous Work

- Interpreter Exploitation: Pointer Inference and JIT Spraying
  - Dion Blazakis, Aug 2010
- Attacking Client Side JIT Compilers
  - Chris Rohlf and Yan Ivnitskiy, Aug 2011
- Bypass DEP and CFG using JIT compiler in Chakra engine
  - Yang Yu AKA tombkeeper, Dec 2015

Issue 1: RWX Page

Issue 1: RWX Page

- RWX Page is a disaster for mitigation
- Avoid using RWX Page seems to be obvious
- Many popular software still use RWX Pages

Issue 1: RWX Page

DEMO

Issue 1: RWX Page

- Chrome
  - V8 JavaScript Engine
    - v8::internal::Code is allocated with PAGE_EXECUTE_READWRITE

Issue 1: RWX Page

Issue 1: RWX Page

- Opera
  - V8 JavaScript Engine

Opera 15	
Release date	2013-07-02
Rendering engine	Chromium 28
JavaScript engine	V8
Features	• New rendering engine based on Chromium/Blink • New user interface • Improved site compatibility • Silent auto updates • Combined address and search bar • Enhanced Speed Dial • Stash • Discover • Off-Road mode

DEMO

Issue 1: RWX Page

Issue 1: RWX Page

- Firefox
  - SpiderMonkey JavaScript Engine
    - RWX Pages are removed since version 46

Issue 1: RWX Page

- Firefox
  - Still use RWX Pages

DEMO

Issue 1: RWX Page

Issue 1: RWX Page

- IE & Edge
  - Chakra JavaScript Engine
    - No RWX Pages

Issue 1: RWX Page

- IE & Edge
  - There used to be a more general issue

DEMO

Issue 1: RWX Page

- What is this RWX Page?
  - ATL Thunk Pool

Issue 1: RWX Page

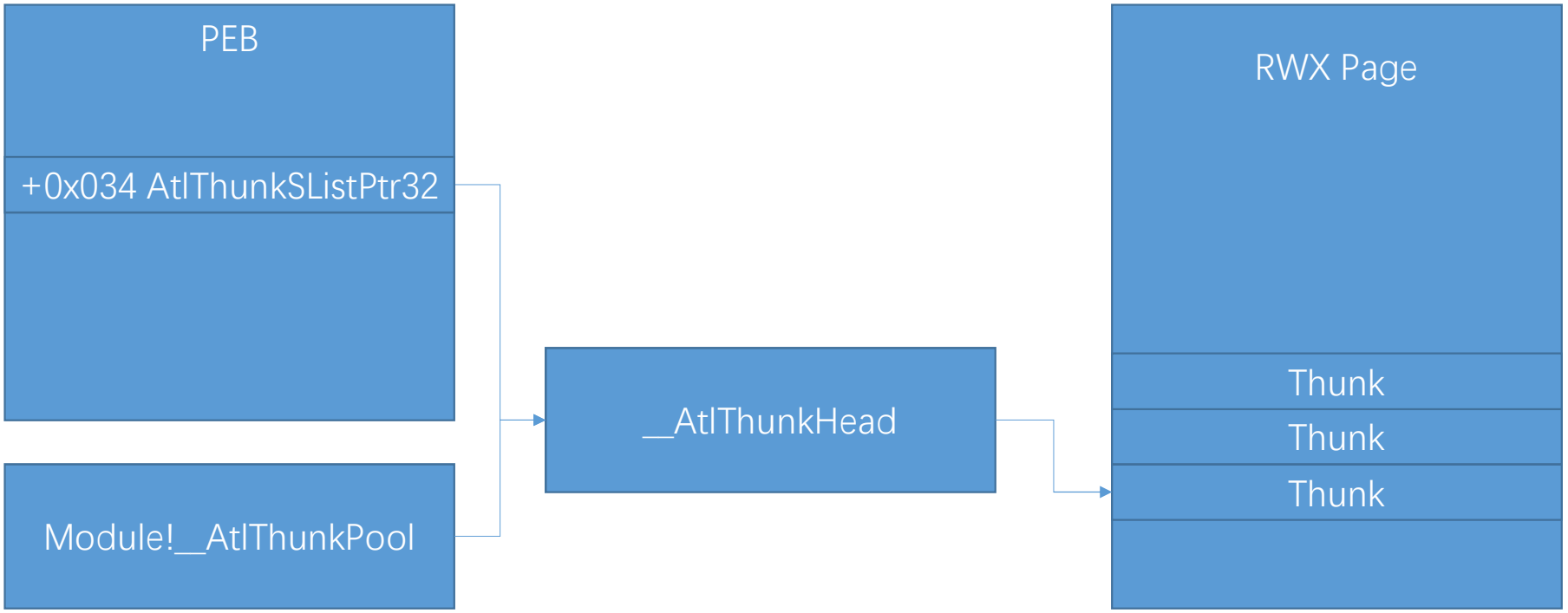
- How is ATL Thunk Pool allocated?
  - __AllocStdCallThunk_cmn

```
pool = VirtualAlloc(0, 0x1000u, 0x1000u, 0x40u);
thunk = pool;
if ( !pool )
  return 0;
next = *pool;
thunk2 = InterlockedPopEntrySList(__AtlThunkPool);
if ( thunk2 )
{
  VirtualFree(thunk, 0, 0x8000u);
  result = thunk2;
}
else
{
  end = thunk + 0xFF0;
  do
  {
    InterlockedPushEntrySList(__AtlThunkPool, thunk);
    thunk += 0x10;
  }
  while ( thunk < end );
  result = thunk;
}
```

PAGE_EXECUTE_READWRITE

Issue 1: RWX Page

- Where is ATL Thunk Pool?



Issue 1: RWX Page

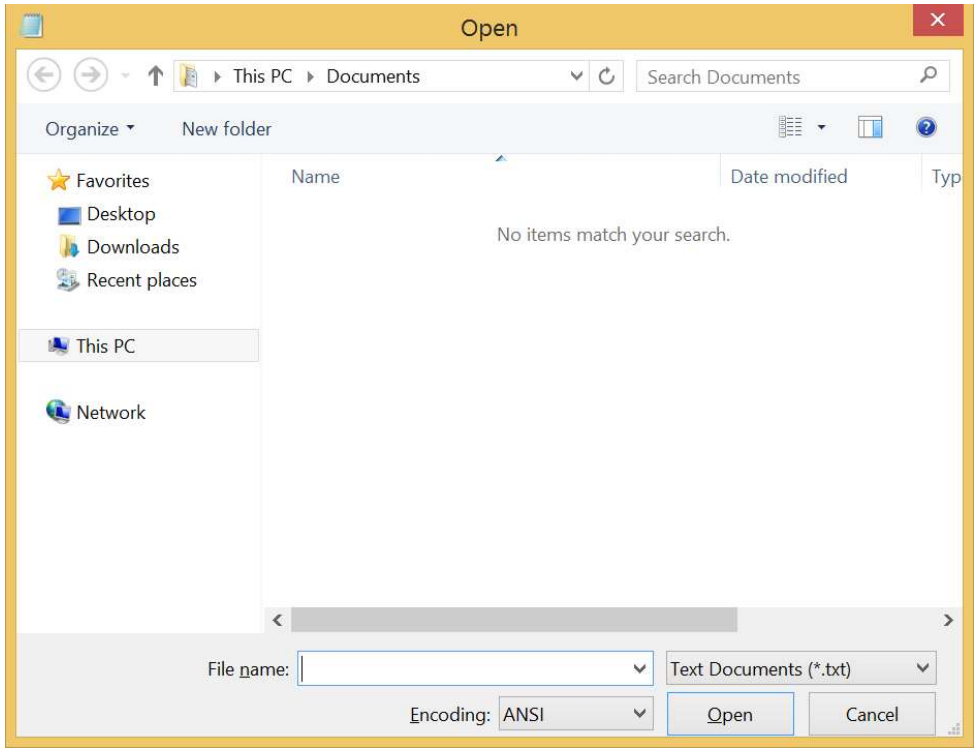
- Who use ATL Thunk Pool?

Issue 1: RWX Page

- Who use ATL Thunk Pool?

Issue 1: RWX Page

- Who use ATL Thunk Pool?



Issue 1: RWX Page

- Fix of the Issue
  - A new library atlthunk.dll is introduced
  - AtlThunk_AllocateData is called instead of __AllocStdCallThunk_cmn
    - No RWX Page anymore

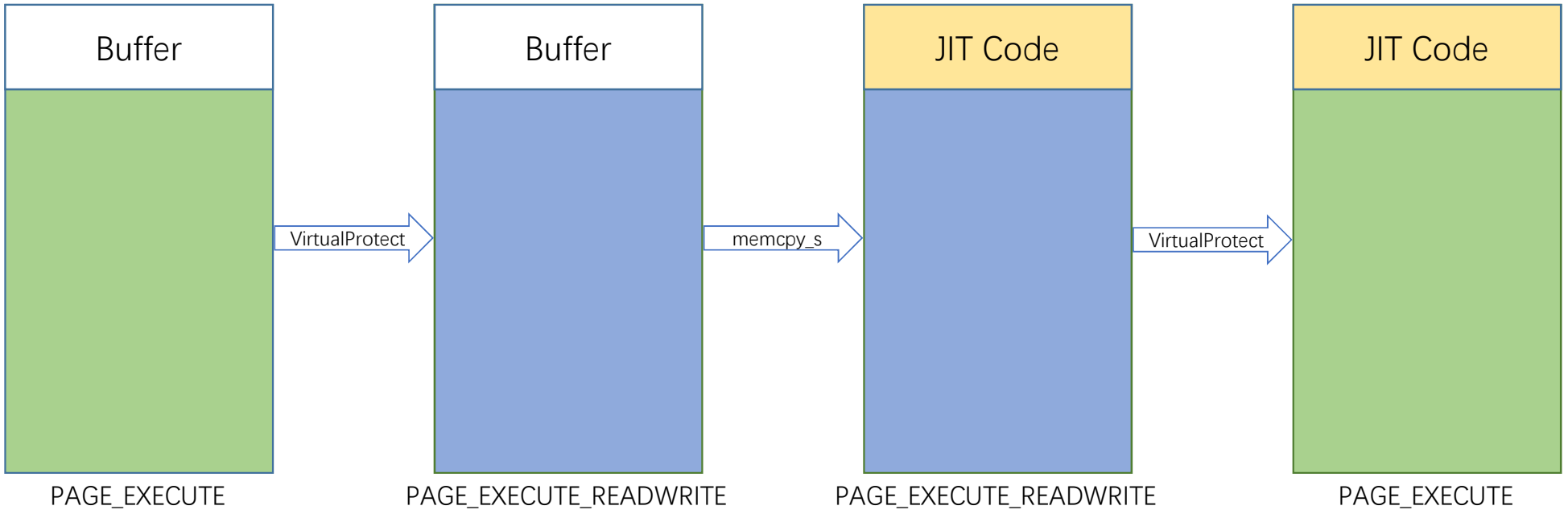
Issue 2: Bad Neighbor

Issue 2: Bad Neighbor

- Is RWX necessary?
- Not at the same time
  - When preparing shellcode RW is enough
  - When executing shellcode X is enough

Issue 2: Bad Neighbor

- How chakra manage JIT code



Issue 2: Bad Neighbor

- The granularity of VirtualProtect is Page

Parameters

lpAddress [in]

A pointer an address that describes the starting page of the region of pages whose access protection attributes are to be changed.

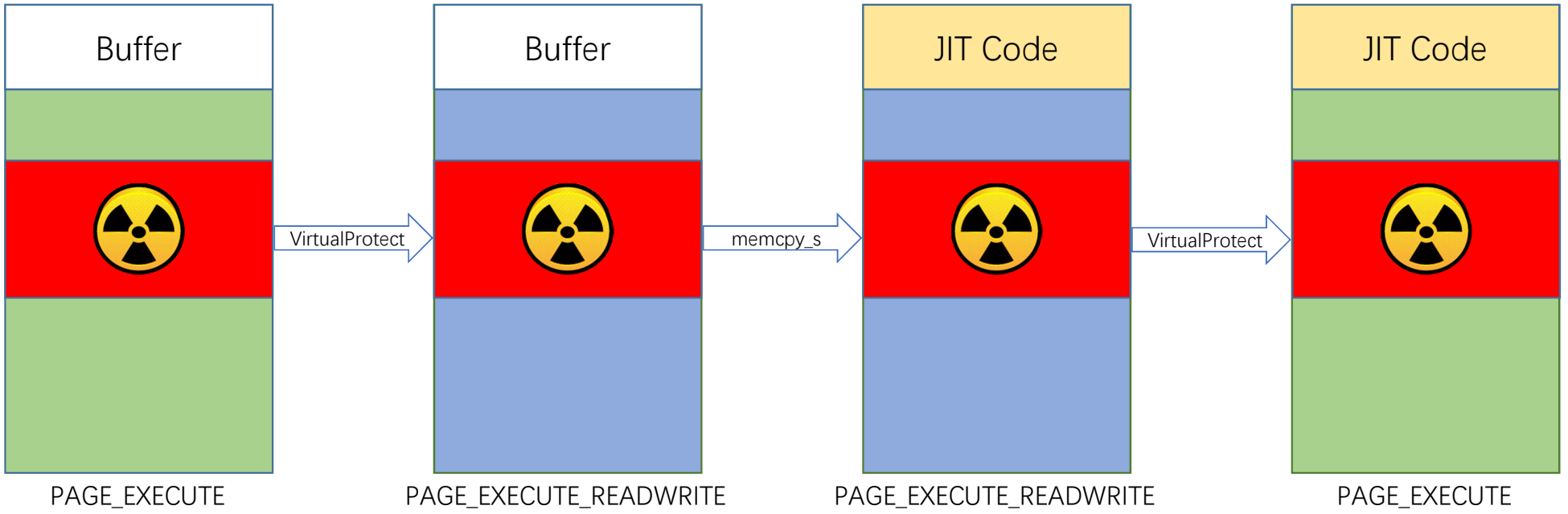
All pages in the specified region must be within the same reserved region allocated when calling the VirtualAlloc or VirtualAllocEx function using **MEM_RESERVE**. The pages cannot span adjacent reserved regions that were allocated by separate calls to **VirtualAlloc** or **VirtualAllocEx** using **MEM_RESERVE**.

dwSize [in]

The size of the region whose access protection attributes are to be changed, in bytes. The region of affected pages includes all pages containing one or more bytes in the range from the *lpAddress* parameter to `(lpAddress+dwSize)`. This means that a 2-byte range straddling a page boundary causes the protection attributes of both pages to be changed.

Issue 2: Bad Neighbor

- Bad neighbor



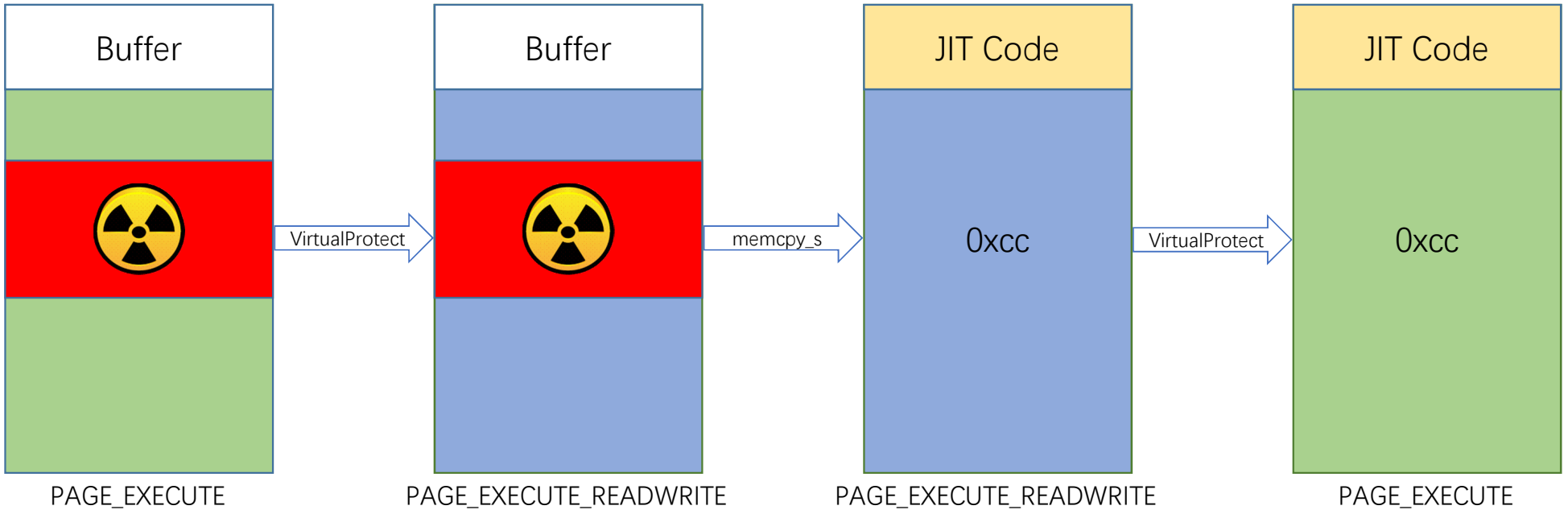
Issue 2: Bad Neighbor

- How Chakra allocate buffer for JIT
  - Allocate from CustomHeap
  - Try to find an existing Page in the buckets
    - PAGE_EXECUTE
  - Allocate a new Page if failed to get one
    - PAGE_READWRITE
  - Buffer is allocated from that Page

DEMO

Issue 2: Bad Neighbor

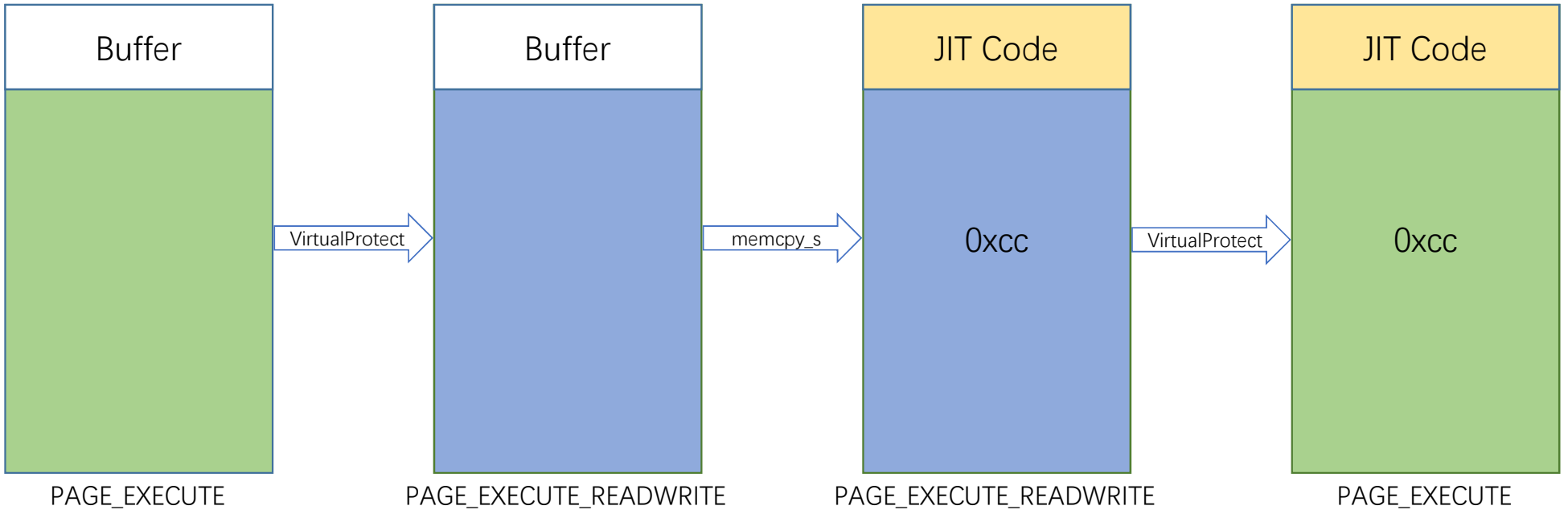
- Fix of the Issue



Issue 3: Unsecure Default Behavior

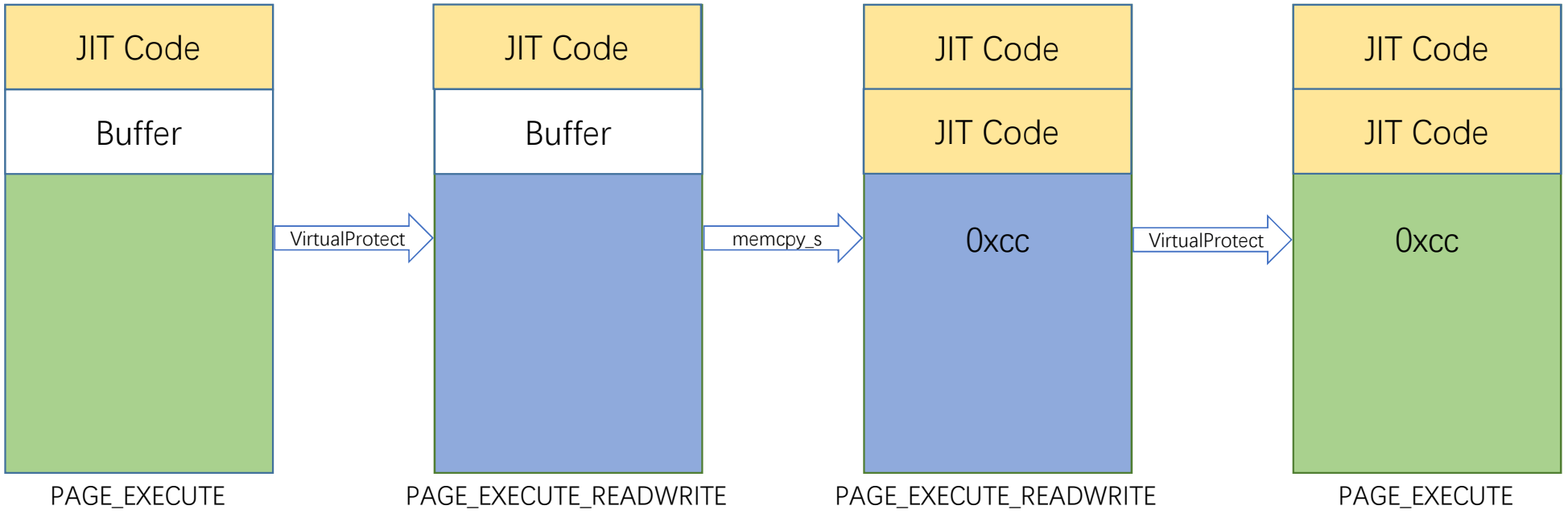
Issue 3: Unsecure Default Behavior

- Review previous issue – new page



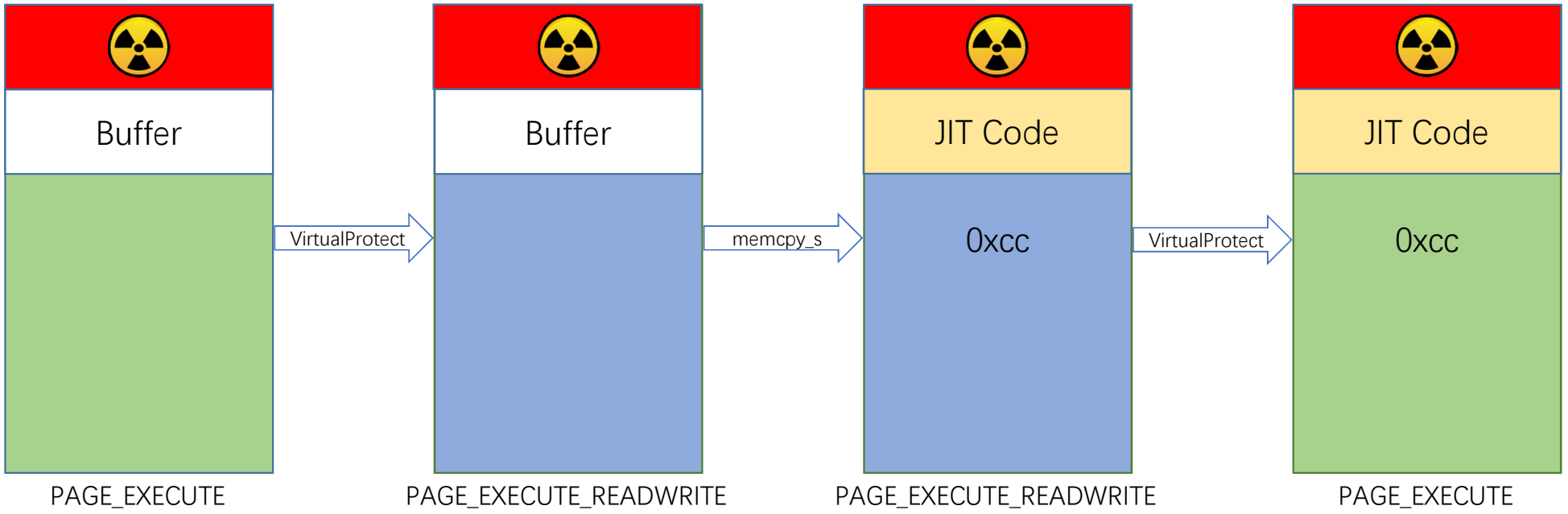
Issue 3: Unsecure Default Behavior

- Review previous issue – existing page



Issue 3: Unsecure Default Behavior

- Only remaining part are sanitized



Issue 3: Unsecure Default Behavior

- Problem
  - The existing Page is PAGE_EXECUTE
  - We can not write shellcode there
- Solution
  - CustomHeap search buckets for an existing Page
  - We can control which Page is used by inserting a fake one
    - PAGE_READWRITE

Issue 3: Unsecure Default Behavior

- Why CFG did not prevent those exploits?

Issue 3: Unsecure Default Behavior

- CFG is not enabled by default
  - VirtualAlloc will set all locations in the pages as valid
  - Unless PAGE_TARGETS_INVALID is explicitly set

PAGE_TARGETS_INVALID 0x40000000	Sets all locations in the pages as invalid targets for CFG. Used along with any execute page protection like **PAGE_EXECUTE**, **PAGE_EXECUTE_READ**, **PAGE_EXECUTE_READWRITE** and **PAGE_EXECUTE_WRITECOPY**. Any indirect call to locations in those pages will fail CFG checks and the process will be terminated. The default behavior for executable pages allocated is to be marked valid call targets for CFG. This flag is not supported by the **VirtualProtect** or CreateFileMapping functions.

Issue 3: Unsecure Default Behavior

- CFG is not enabled by default
  - VirtualProtect will update all locations in the pages as valid
  - Unless PAGE_TARGETS_NO_UPDATE is explicitly set

PAGE_TARGETS_NO_UPDATE 0x40000000	Pages in the region will not have their CFG information updated while the protection changes for VirtualProtect. For example, if the pages in the region was allocated using **PAGE_TARGETS_INVALID**, then the invalid information will be maintained while the page protection changes. This flag is only valid when the protection changes to an executable type like **PAGE_EXECUTE**, **PAGE_EXECUTE_READ**, **PAGE_EXECUTE_READWRITE** and **PAGE_EXECUTE_WRITECOPY**. The default behavior for **VirtualProtect** protection change to executable is to mark all locations as valid call targets for CFG.

Issue 3: Unsecure Default Behavior

- CFG is not enabled by default
  - Maybe compatibility reason
  - Otherwise SetProcessValidCallTargets must be called explicitly

SetProcessValidCallTargets function

[Some information relates to pre-released product which may be substantially modified before it's commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here.]

Provides CFG with a list of valid indirect call targets and specifies whether they should be marked valid or not. The valid call target information is provided as a list of offsets relative to a virtual memory range (start and size of the range). The call targets specified should be 16-byte aligned and in ascending order.

Issue 3: Unsecure Default Behavior

- Fix of the Issue
  - Enable CFG in JIT explicitly

Conclusion

- JIT implement is not secure enough currently
- Some guideline should be followed to implement JIT securely

Recommended Practice

- When allocating Buffer for JIT
  - Allocate a page with PAGE_READWRITE | PAGE_TARGETS_INVALID
  - Sanitize the whole page
  - Covert the Page to PAGE_EXECUTE | PAGE_TARGETS_NO_UPDATE
  - Add the Page to Buffer Manager

Recommended Practice

- When writing Code to Buffer
  - Get a Page from Buffer Manager
  - Verify that the Page is still PAGE_EXECUTE
  - Convert the Page to PAGE_EXECUTE_READWRITE | PAGE_TARGETS_NO_UPDATE
  - Copy the code to the Page
  - Sanitize unused part of that Page
  - Revert the Page to PAGE_EXECUTE | PAGE_TARGETS_NO_UPDATE
  - Call SetProcessValidCallTargets for the entrance

Call to Action

- Never use PAGE_EXECUTE_READWRITE
- Always sanitize unused memory in the same Page
- Always enable CFG explicitly

Questions?