(More)Advanced defense for IE

Bo Qu, Royce Lu & Ga1ois

Agenda

- whoami
- history|less
- ls ~/InternetExploder/js
- ls ~/InternetExploder/flash
- ls ~/InternetExploder/luchong
- ./exp
- man -h

About us

- IPS team of Palo Alto Networks(09:00-17:00)
- Researchers(19:00-22:00)
  - http://osvdb.org/affiliations/1148-palo-alto-networks
- White hats
  - 100+ CVEs from vendors
  - 0 bug sold to ZDI/3rd party
  - Exploit writer for defense in depth

History

- June patch
  - Isolated heap
  - Not a problem
- July patch
  - Deferred free
  - Not a silver bullet

History

- UAF is NOT everything
  - Type confusion
  - Overflow
  - Uninitialized memory
  - Other memory corruption…
- Defense on the heap, or deeper?

Project JS

- Exploit trends in IE browser
  - UAF and OBA(Out of Boundary Access)
  - Write primitive
  - Write what?
    - BSTR
    - *Array*
    - Element Attribute
    - Other

Project JS

- Why Array?
  - Simple: Few JS code
  - Powerful: From write one byte to read/write anywhere
  - Extensive: UAF and OBA, heap spray and heap layout, javascript and vbscript

Project JS

- Defense array heap spray and heap layout
  - Hook array allocate function
  - Loop Counts and array length

Project JS

- Defense array write primitive
  - Core idea: Precise “address + length/buffer” modification checking
  - Three types
    - Different allocate functions
    - Different get/set/length functions
  - Code overlapping problem of inline hook
  - Different functions between JIT and not

Project JS

- Defense array write primitive
  - Typed Array
  - Native Int Array with head and data together[not sparse]
  - Native Int Array with head and data alone[not sparse]

Project JS

- Limitations
  - Check most UAF/OBA exploit
    - except the one not using BSTR/Array/EA
    - except the one like cve-2013-2551
  - “BSTR, Element Attribute” to be continued…

Project Flash

- Why flash?
  - It is popular
  - It provides more than it should have
  - It used to be a blind point

Project Flash

- [Heap] Spray
  - Regular heap spray
  - Small chunk spray

Project Flash

- [Heap] Spray
  - Two-layer defense
  - Object allocation monitor
  - Memory usage monitor
  - Reduce false positives

Project Flash

- Vector
  - The root of all evils
  - Modification of length
  - Modification of buffer address
  - Full memory access is *bad*

Project Flash

- Vector

0x00	0x04	0x08	0x0c	0x10	0x14	0x18	0x1c	0x20	0x24
VT	0x4e	?	?	?	0x00	PTR	0x00	0x00	0x00

  - Write 0? 1-(58.3%)^n chance to exploit.

Project Flash

- Vector
  - Hooking read/write functions(6 places?)
  - Length checking
    - Single object checking
    - Multiple objects checking
  - Buffer checking
    - Mapping table
    - Buffer validation

Project Luchong

- Project Luchong(路冲)
  - The bad Fengshui
  - Destroy predictable heap layout
  - Transparent to user level

Project Luchong

- Why Luchong
  - Heap Fengshui is vital for exploitation
  - Heap is predictable
    - Continuous, linear increasing.
    - Alignment.
    - Other features for performance

Project Luchong

- Mechanism
  - Understand the accurate spay
    - Higher 20 bits
      - Guaranteed by repeatedly allocation
      - Optimized by linear increasing mechanism
    - Lower 12 bits
      - Guaranteed by alignment

Project Luchong

				0C0C0C0C					

paloalto NETWORKS

Project Luchong

- Mechanism(cont.)
  - Break linear increasing mechanism
    - Large sized chunk
    - Small sized chunk
    - Light-weight solution

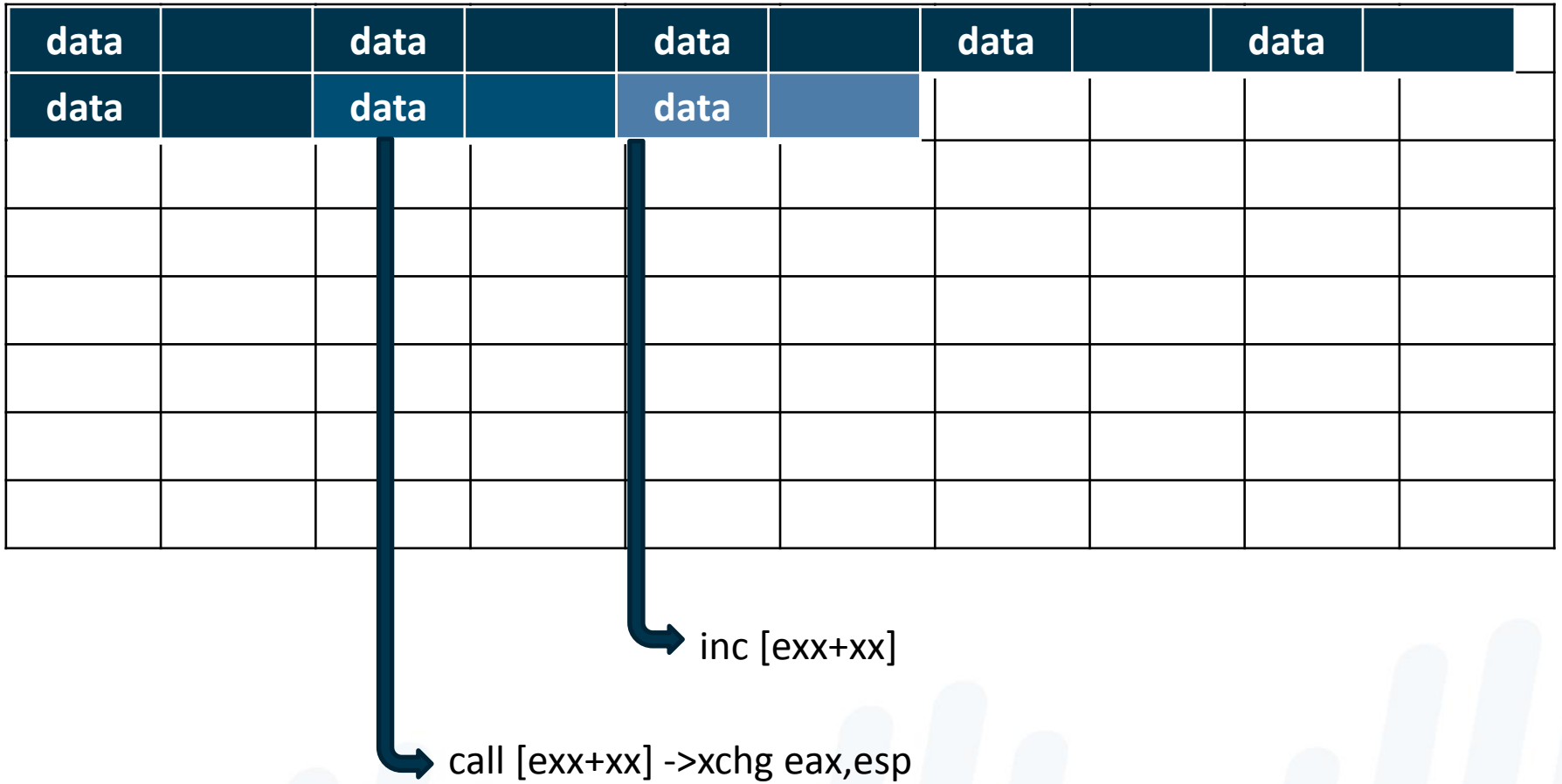
Project Luchong

- Mechanism(cont.)
  - Break the alignment
    - Large sized chunk (0x1000 alignment)
    - Small sized chunk (0x08 alignment)
    - Allocate more bytes than it requests

Project Luchong

- Mechanism(cont.)
  - Understand the exploits
    - Buggy object and exploit object are different ones.
    - Exploit object must be placed in certain position
      - UAF, same position
      - OBA, next to buggy object
      - Others
    - Size matters

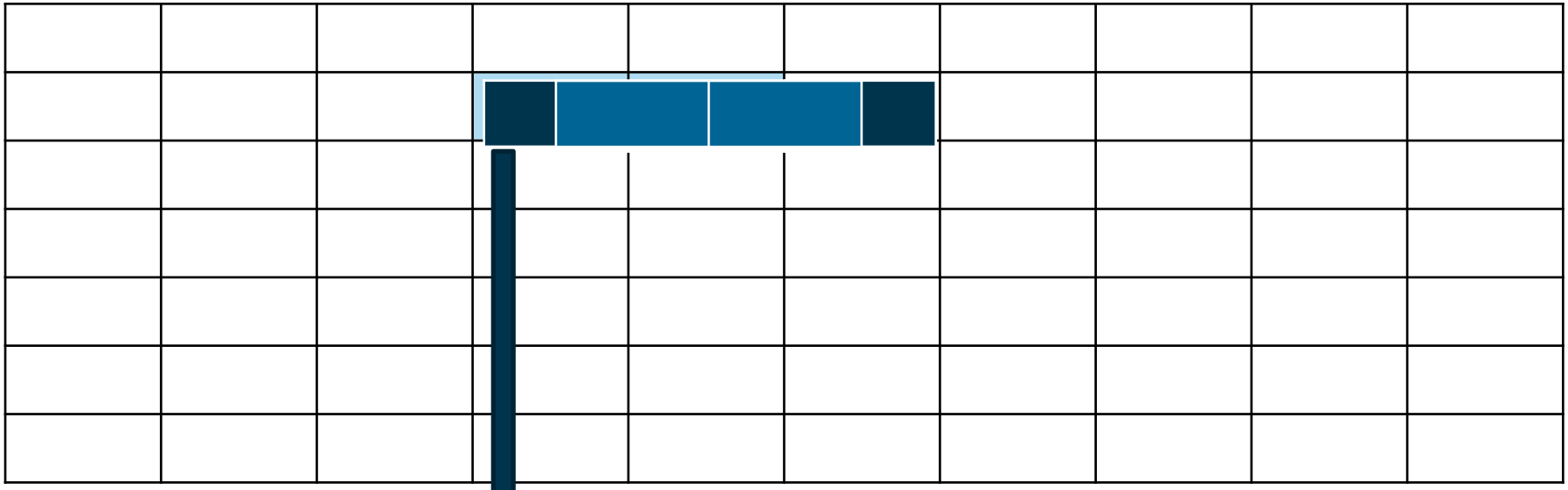
Project Luchong



Project Luchong

- Mechanism(cont.)
  - Break the heap fengshui
    - Focus on small chunk(<0x200 bytes)
    - Create more heaps
    - Randomize the memory layout
    - Randomize the actual size

Project Luchong



1, misaligned
2, inaccurate data control
3, failed exploitation

Project Luchong

- Everything else
  - Cookie for the heap chunk
    - Post exploitation checking
  - Chunk initialization
    - Deal with uninitialized cases
  - Timestamp for the free’d chunk
    - Enhanced deferred free

Project Luchong

- Limitation despite of 99% coverage
  - Trade off between performance and accuracy
  - Stack things
    - CVE-2014-2797, type confusion on the stack
  - Brute force
  - Logic bugs

Demo

- Demo

Q&A