

# Внешний аудит безопасности корпоративных сетей

## Лекция 9 Переполнения буфера III

Семён Новосёлов

2022



# Переполнения буфера

- Переполнение стека
- **Переполнение кучи**



# Пример уязвимости. EternalBlue

- **EternalBlue** — эксплойт к уязвимости в SMB (CVE-2017-0144), разработанный АНБ
- Выложен в сеть хакерской группой Shadow Brokers
- Были уязвимы все версии Windows (XP, 7, 8, 10)
- Эксплойты есть в составе Metasploit



# WannaCry

- EternalBlue  
Использовался  
программами-  
шифровальщиками  
WannaCry/NotPetya



# Связанные уязвимости и вариации

## EternalChampion

- CVE-2017-0146
- race condition
- банковский троян TrickBot

## EternalRomance

- CVE-2017-0143 + CVE-2017-0147
- вирус-шифровальщик BadRabbit

## EternalSynergy

- EternalChampion + EternalRomance

Oops! Your files have been encrypted.

If you see this text, your files are no longer accessible. You might have been looking for a way to recover your files. Don't waste your time. No one will be able to recover them without our decryption service.

We guarantee that you can recover all your files safely. All you need to do is submit the payment and get the decryption password.

Visit our web service at [caforssztxqzf2nm.onion](http://caforssztxqzf2nm.onion)

Your personal installation key#1:

ZMCOXBgX7oKoxrakfBMXAloe0t6McW7Wfx5I+rJJD8hzv6DPpYhNQNCioJW6GX3w  
y4wZX6UdirzbsD7sIeuKEndRDDeez+FLaoElfQxGsGQ2qUOC4Aaxd7KS8T301c0ig  
mc1A0Uy+r71X6QcIBZe3il7gqNTb1AyKqUK94dANmsI7hQcrC16q2WnxRjH4rF7e  
3sFVUaJW+imUby9m+LjnoMqb5zUJzU3y2sj7UCoj4bWTrM093a9pGuyh058vPY2I  
2LqEcudkJQFSjUmb8FN7E8pSyo20F4jZ5KRQMSESNRt6hBBxV0o3Geb15KBEjW1Y  
giKd0daIPSunWM0IJA5GkfccbgTVX77Rjg==

If you have already got the password, please enter it below.  
Password#1: \_

# Server Message Block (SMB)

- Протокол для удалённого доступа к файлам, принтерам и другим сетевым ресурсам
- Обнаружение компьютеров и устройств в сети работает в Windows через SMB, включён по-умолчанию
- Широко используется в составе ОС Windows
- Номера портов: 139/445

# Уязвимость CVE-2017-0144, MS-010

Уязвим код в функции `SrvOs2FeaListSizeToNt` драйвера `srvnet.sys`:

```
if (variableBuffer >= lastValidLocation ||  
    variableBuffer + fea->cbName + 1 + SmbGetUshort(&fea->cbValue)) > lastValidLocation)  
{  
    SmbPutUshort( &FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList) );  
    break;  
}
```

Функция преобразует данные о расширенных атрибутах файлов (список `FeaList`) из присланного пакета в структуру `_FILE_FULL_EA_INFORMATION`

# Расширенные файловые атрибуты

1. Флаги (сжатый, зашифрованный, скрытый, ... )
2. Пользовательские метаданные в формате ключ-значение

**Пример:** файловые потоки в NTFS

**Использование:** хранение контрольных сумм, отметки о проверке антивирусом



# Структуры для хранения файловых атрибутов

```
typedef struct _FEA {  
    BYTE fEA; // флаги  
    BYTE cbName; // длина ключа  
    USHORT cbValue; // длина значения  
    //...  
} FEA;
```

```
typedef struct _FEALIST {  
    ULONG cbList; // размер списка  
    FEA list[1];  
} FEALIST, *PFEALIST;
```

```
if (variableBuffer >= lastValidLocation ||  
    variableBuffer + fea->cbName + 1 + SmbGetUshort(&fea->cbValue) > lastValidLocation)  
{  
    SmbPutUshort(&FeaList->cbList, PTR_DIFF_SHORT(fea, FeaList));  
    break;  
}
```

```
typedef struct _FEALIST {  
    ULONG cbList; // размер списка  
    FEA list[1];  
} FEALIST, *PFEALIST;
```

```
typedef struct _FEA {  
    BYTE fEA; // флаги  
    BYTE cbName; // длина ключа  
    USHORT cbValue; // длина значения  
    //...  
} FEA;
```

**Целочисленное переполнение:** SmbPutUshort записывает UShort (2-х байтовый тип) в ULONG (4-байта). Если передать больше  $2^{16}-1$  атрибутов, то выделится слишком мало памяти для хранения списка и перезапишется память за списком

# Последствия переполнения

- Список атрибутов FeaList контролируется пользователем (передается при запросе)
- Можно перезаписать участок памяти своими данными
- Обработка производится в **srvnet.sys**  $\Rightarrow$  перезаписывается память ядра, код выполняется с максимальными привилегиями

# Выполнение кода по произвольному адресу

**Задача:** Перезаписать какой-либо указатель на функцию, лежащий за буфером.

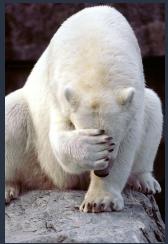
- Каждый раз при получении SMB-запроса драйвер **srvnet.sys** создаёт структуру **SRVNET\_BUFFER**
- Структура содержит указатель на функцию, которая выполняется после обработки SMB-запроса

```

struct SRVNET_BUFFER {
    // offset from POOLHDR: 0x10
    USHORT flag;
    char pad[2];
    char unknown0[12];
    // offset from SRVNET_POOLHDR: 0x20
    LIST_ENTRY list;
    // offset from SRVNET_POOLHDR: 0x30
    char *pNetBuffer;
    DWORD netbufSize; // size of netBuffer
    DWORD ioStatusInfo; // copy value of IRP.IOStatus.Information
    // offset from SRVNET_POOLHDR: 0x40
    MDL *pMd1; // at offset 0x70
    DWORD nByteProcessed;
    DWORD pad3;
    // offset from SRVNET_POOLHDR: 0x50
    DWORD nbssSize; // size of this smb packet (from user)
    DWORD pad4;
    QWORD pSrvNetWskStruct; // want to change to fake struct address
    // offset from SRVNET_POOLHDR: 0x60
    MDL *pMd12;
    QWORD unknown5;
    // offset from SRVNET_POOLHDR: 0x70
    // MDL md1; // for this srvnetBuffer (so its pointer is srvnetBuffer address)
    // MDL md12;
    // char transportHeader[0x50]; // 0x50 is TRANSPORT_HEADER_SIZE
    // char netBuffer[0];
};

```

Структура  
перезаписывается так,  
чтобы код корректно  
отработал до перехода  
на указатель на  
функцию



# Техника Grooming / Heap Spread

**Проблема:** нет никаких гарантий, что структура `SRVNET_BUFFER` лежит за списком `FeaList`

- Код `srvnet.sys` дополнительно содержит утечку памяти в обработке транзакций, позволяющую выделять большие участки памяти при соединении.
- Большие участки памяти выделяются в памяти последовательно.

## Решение проблемы с расположением SRVNET\_BUFFER:

1. Выделить несколько больших участков памяти (создав соединения)
2. Освободить участок (hole) в середине (закрыв соединение)
3. Следующее соединение с большой вероятностью будет выделять память в hole
  - поместить туда FeaList, перезаписав указатель на функцию в одном из SRVNET\_BUFFER
4. Записать шелкод в участки памяти MEM\_SRVNET\_BUFFER\_i
5. Закрывать соединения (для вызова функции)

MEM\_SRVNET\_BUFFER\_1

MEM\_SRVNET\_BUFFER\_1

MEM\_SRVNET\_BUFFER\_3

Hole (FeaList)

MEM\_SRVNET\_BUFFER\_4

MEM\_SRVNET\_BUFFER\_5

MEM\_SRVNET\_BUFFER\_6

## Дополнительно

- Чтобы посторонние соединения не мешали, первый участок в памяти выделить меньше размера списка FeaList и закрыть соединение перед созданием hole
- Тогда все соединения будут выделять память в этом участке, так как они требуют мало памяти

**Итог:** получаем выполнение кода по любому адресу.



# Последствия

В Windows 7/8/10:

- **ASLR**: некоторые адреса фиксированные (0xffffffffffffd00000), отталкиваемся от них
- **NX-бит**: можно отключить защиту, перезаписав участок памяти ядра отвечающий за права доступа к памяти (PTE)

Все защиты можно обойти и выполнить любой код.

# Литература и ссылки

- RiskSense. EternalBlue Exploit Analysis and Port to Microsoft Windows 10:  
[https://risksense.com/wp-content/uploads/2018/05/White-Paper\\_Eternal-Blue.pdf](https://risksense.com/wp-content/uploads/2018/05/White-Paper_Eternal-Blue.pdf)
- Д. Фостер, В. Лю. Разработка средств безопасности и эксплойтов. 2011
- Joly N. - Mitigating the unkn0wn when your SMB exploit fails  
[https://hitcon.org/2017/CMT/slide-files/d2\\_s2\\_r0.pdf](https://hitcon.org/2017/CMT/slide-files/d2_s2_r0.pdf)
- Microsoft Windows 7/2008 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)  
<https://www.exploit-db.com/exploits/42031>
- S. Le Berre - Bypassing kernel ASLR Target : Windows 10 (remote bypass)

