CVE-2017-8291

202112021 박채우

1. 포스트스크립트

- 고스트 스트립트 취약점을 이해하기 위해서는 포스트스크립트 프로그래밍 언어에 대해 간단히 이해할 필요가 있다.
- 포스트스크립트는 연산의 가장 기본 단위가 되는 객체를 대상으로 일정한 규칙에 따라 읽고 처리한다.

1. 포스트스크립트

- 포스트스크립트에서 가장 기본이 되는 개념인 객체는 코드 실행 과정에 따라 생성, 조작, 소멸 등 다양한 연산이 이루어진다.
- 포스트스크립트에서는 Simple과 Composite 2가지 유형으로 객체를 구분한다.

1. 포스트스크립트

SIMPLE OBJECTS	COMPOSITE OBJECTS
boolean	array
fontID	dictionary
integer	file
mark	gstate *(LanguageLevel 2)*
name	packedarray *(LanguageLevel 2)*
null	save
operator	string
real	

1. 포스트스크립트

- Simple 객체 : 하나의 값을 고정적으로 가지기 때문에 객체 본인과 본인이 표현하고자 하는 값은 분리될 수 없다.

Attribute	Types	00 00	Value

1. 포스트스크립트

- Composite 객체 : 객체 본인과 값이 연결이 되어있는 구조이기 때문에 객체가 참조하는 값은 연산에 따라 달라질 수 있다.
- 이러한 특징 때문에 copy 연산을 수행했을 때 Composite 객체는 값은 복사가 되지 않고 복사 대상의 값을 참조한다.

Attribute	Types	Size of Ref.	Pointer of Ref.

포스트 스크립트

타입	타입 값	타입	타입 값
No value	0x00	Integer	0x0B
Boolean	0x01	Mark	0x0C
Dictionary	0x02	Name	0x0D
File	0x03	Null	0x0E
Array	0x04	Operator	0x0F
Mixedarray	0x05	Real	0x10
Short array	0x06	Save	0x11
Unused array	0x07	String	0x12
Struct	0x08	Device	0x13
Astruct	0x09	Oparray	0x14
fondID	0x0A	Next index	0x15

1. 포스트 스크립트

- 예시 val false def;

Attribute	01	00 00	00 00 00 00

- 예시 num 123 def;

Attribute	0B	00 00	7B 00 00 00

- 예시 arr 16#100 array def;

Attribute	04	00 01	80 36 B9 00
Attribute	0E	CD CD	CD CD CD CD

1. 포스트 스크립트

- 즉시 실행 스택

40 60 add 2 div

- 지연 실행 스택

/average {add 2 div} def
40 60 average

2. CVE-2017-8291

2. CVE-2017-8291

- 1. 스택에 다수의 문자열을 추가한 후 여러 차례 eqproc 연산을 호출하여 스택포인터가 다시 감소하도록 유도
- 2. 스택포인터가 공격자가 의도한 고정된 위치에 도달하면 2개의 동일한 배열 객체를 스택에 추가 후 그 중 하나를 문자열 객체로 변경
- 3. 배열 객체는 주소에 해당하는 값에 직접 접근하여 값의 수정이 불가능하지만 문자열 객체는 수정이 가능함.
- 4. 문자열 객체를 이용해 주소에 쉘코드가 저장된 주소값을 추가하여 공격 진행

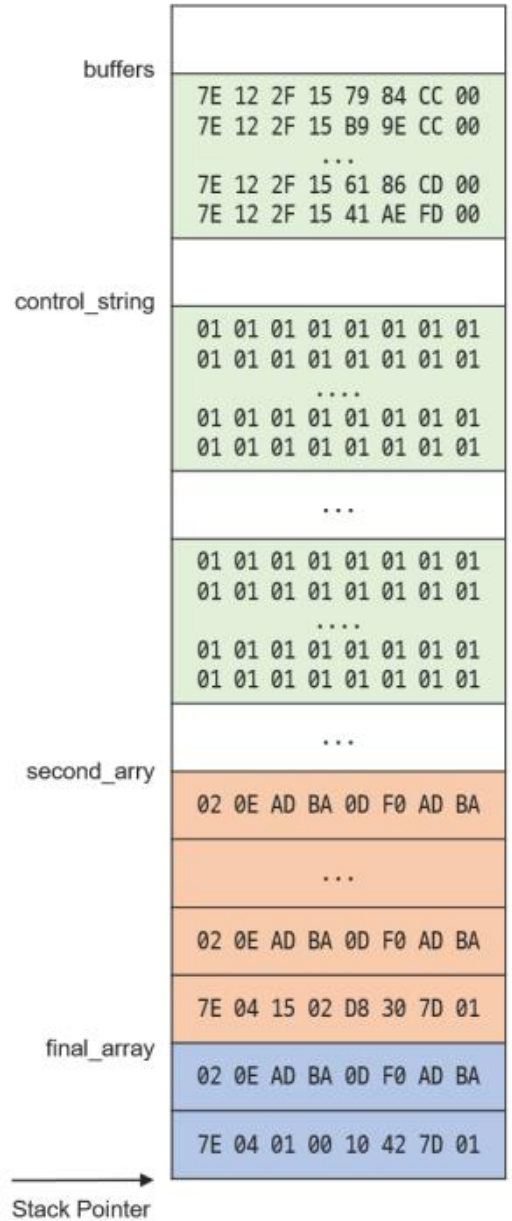
2. CVE-2017-8291

1. Spray
   스택 포인터를 높은 주소로 올리고 고정된 길이의 문자열을 여러 번 반복해서 스택에 Push 하는 단계

   0x31E 크기의 first_array 배열을 push 하고 그 후 0x215 크기의 second_array를 0x10회 push 한다.

   0x152F 길이의 문자열 객체 string의 내용을 모두 1로 초기화 후 해당 문자열을 참조하는 buffers 배열을 선언한다.

   그 후 second_array와 0x01 크기의 배열인 final_array 마지막으로 push


buffers
7E 12 2F 15 79 84 CC 00
7E 12 2F 15 B9 9E CC 00
...
7E 12 2F 15 61 86 CD 00
7E 12 2F 15 41 AE FD 00
control_string
01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01
....
01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01
...
01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01
....
01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01
...
second_arry
02 0E AD BA 0D F0 AD BA
...
02 0E AD BA 0D F0 AD BA
7E 04 15 02 D8 30 7D 01
final_array
02 0E AD BA 0D F0 AD BA
7E 04 01 00 10 42 7D 01
Stack Pointer

2. CVE-2017-8291

2. Overwrite

현재 위치해있는 스택 포인터의 위치를 eqproc 연산자를 이용해 다시 감소시키는 단계

String 배열의 시작 위치를 기준으로 0x150F 번째 문자를 overwrite_pos 으로 지정 후 스택 포인터의 감소는 해당 포지션의 문자가 0 으로 바뀔 때까지 진행한다.

2. CVE-2017-8291

3. Type confusion

   Element를 0xFFFF개 가지는 배열 leaked_array(1) 와 정수 0, 다시 leaked_array(2)를 스택에 push 한다.

   그 후 overwrite_pos 를 기준으로 0x18~0x1B 위치에 값 0x7E 0x12 0x00 0x80 값을 저장한다.

   해당 위치는 두번째로 push 된 leaked_array(1)의 위치이며 해당 값을 저장함으로써 leaked_array(2)은 0x12(string)객체로 변경됐고 0x7E(쓰기, 읽기, 실행 권한)을 0x8000 길이 만큼 보유하고 있는 것으로 변경됨

2. CVE-2017-8291

3. Type confusion

```
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00 01 01 01 00 00 01 01 7E 04 FF FF 60 00 37 00
02 0B 18 03 00 00 00 00 7E 04 FF FF 60 00 37 00
```

=> 7E 12 00 80 60 00 37 00

2. CVE-2017-8291

3. Type confusion
   그 후 권한이 미치는 영역을 늘리기 위해 문자열 객체인 leaked_array(2)를 이용해 0x7FFF8000 까지 읽기, 쓰기, 실행하는 권한을 가지게 된다.

2. CVE-2017-8291

4. Chain
   이전까지의 과정은 모두 임의의 메모리 영역에 권한을 부여하여 쉘 코드가 정상적으로 실행되도록 하는 준비단계

   leaked_array(2)의 Element에 다음과 같은 정보를 put 한다.

   1. shellcode가 저장된 주소(shell_addr)
   2. 0x100길이의 문자열(stub_addr) (해당 문자열의 +0x30 위치에 shell_addr과 가젯, api 등의 정보를 담고 있는 shell_stub 생성)
   3. 스트림 상태 구조체 주소 (해당 주소의 +0xB0 위치에 stub_addr) 저장

2. CVE-2017-8291

5. Shellcodel Execute

leaked_array 1 get closefile

Quit

Closefile 명령을 실행할 때 sclose 함수가 호출 되는데 해당 함수는 스트림 상태 구조체를 참조한다.

즉 스트림 상태 구조체의 +0xB0 부분에 있는 stub_addr에 접근하고 연쇄적으로 shell_stub가 실행 => shell_addr 접근 => shell code 실행의 단계로 최종적으로 권한을 가진 shellcode가 실행된다.