HWP Shellcode Detector 제작 보고서

작성자 : 손영락

# 사용방법

**python HSD.py [sample.hwp]**

* **sample.hwp를 변환시킴**

**Enviroment : Windows / python3**

# **쉘코드 탐지 방법**

* **Heap Spray**
* **PEB (Process Enviroment Block)**
* **LDR\_DATA\_TABLE\_ENTRY**
* **TOPSTACK & TEB**
* **Window Function**
* **SHE (Structured Exception Handler)**

**위 6가지 방법을 통해 ShellCode를 탐지하고 삭제하는 방안을 세웠습니다.**

1. **Heap Spray**

**첫 번째로 hwp문서에 삽입되어 있는 쉘코드는 대부분 Heap Spray를 통해서 반복되는 바이트를 삽입 해놓았습니다. Entropy를 사용하여 Entropy가 0.2이하일 경우 heap spray가 사용되었다고 판단하였습니다(여러 셈플 결과). 그런 경우 bodytext의 정상적인 부분을 살리기 위하여 3000byte를 제외한 나머지 아랫 바이트들을 nop로 채웠습니다.**

* **그래도 파일이 손상되는 것을 막을 수 없었습니다.**
* **Hwp bodysection 의 정상적인 부분의 크기를 알 수 있는 방법을 찾아야하는데 찾지 못하였습니다.**

1. **PEB**

**대부분의 쉘코드는 dll의 주소를 사용하기 때문에 동적으로 할당되는 주소를 알려고 합니다.**

**mov $reg,DWORD PTR fs:0x30**

**mov $reg, fs:[$reg+30h]**

**위의 어셈코드와 같이 fs:0x30와 같은 구조를 가지고 있는 바이트를 nop로 변환시켰습니다.**

1. **LDR\_DATA\_TABLE\_ENTRY**

**로드된 모듈에 있는 정보를 포함하고 있는 구조체로 모듈의 주솟값을 확인 할 수 있습니다. 해서 해당 어셈의 구조를 가지고 있는 바이트를 nop 처리했습니다.**

1. **TOPSTACK & TEB**

**PEB에서 TEB와 동일 코드를 nop처리하기 때문에 중복되지만 조사 중 TEB의 topstack을 통해 공격하는 방법을 찾아, 해당 shellcode를 처리하기 위해서 mov esi, fs:[esi + 0x18] 구조를 가지는 shellcode를 nop처리 했습니다. Topstack의 경우 MZ일때까지 loop를 돌기 때문에 해당 바이트를 nop처리 했습니다.**

1. **SEH**

**SHE overwrite공격 기법을 막기 위해서 xor / mov esi, fs:[ecx] / not이 반복되는 구조를 찾아서 nop처리를 했습니다.**

1. **Window Function**

**Window shellcode에서 자주 사용된다고 생각하는 함수들의 바이트를 nop로 처리하였습니다. Hwp의 bodysection에서는 사용되지 않는 함수라고 판단했기 때문입니다.**

**Bindata파일은 확인하지 않으려고 했지만 해당 부분에 다른 (esp)와 같은 파일을 넣는 악성코드들이 있어 해당 부분의 추가하였습니다.**

**장점 :**

1. **빠르다는 것입니다. Entropy를 제외한 나머지 부분에서 정규식을 이용하여 해당 바이트의 구조를 가지는 것만 변환시키기 때문에 빠른 속도를 가지고 있습니다.**
2. **Entropy의 경우 shellcode를 가지고 있는 파일을 높은 확률로 잡습니다.**

**단점:**

1. **대부분이 정규식을 사용하기 때문에 해당 구조를 벗어나면 탐지하지 못합니다.**
2. **Entropy의 경우 어디까지가 정상 바이트인지 파악하지 못해 해당 파일의 Entropy가 낮을 경우 파일이 손상될 확률이 높습니다.**
3. **Decompress 후에 compress를 진행할때 바이트수가 많이 차이나는 경우가 있는데 해당 부분을 해결하지 못했습니다.**