Entropy 를 이용한 upx packing detection 에 대한 연구 보고서

일자 : 2018.11.30

이름 : 서창범

학번 : 2014004648

소속 : 컴퓨터공학부

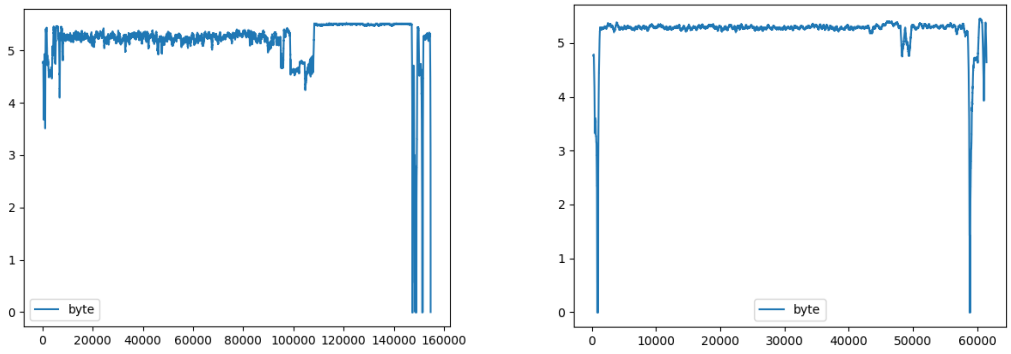
서론

패킹은 실행파일의 압축 기법으로서, 압축 데이터와 압축 해제 코드를 실행파일에 포함하여, PE load 이후 메모리상에 압축 데이터를 해제하여 보다 작은 실행파일 용량으로 원본 실행파일과 다름없이 프로그램을 실행하게 하는 기법입니다. 원본 PE 파일에 포함된 코드들을 압축하여 압축 데이터 형태로 갖고 있기 때문에, 패킹된 바이너리는 정적분석이 어려워지게 됩니다. 이러한 점에서 악의적인 공격자가 악성코드를 퍼뜨릴 때, 다양한 패커를 사용하여 자신이 만든 악성코드를 분석가로 하여금 분석하기 어렵게 만듭니다. 그래서 악성코드 분석에 있어서 악성코드의 패킹 여부를 파악하는 것은 중요하며, 이번 연구과제를 통해 entropy 계산을 이용한 upx 패킹 탐지에 관한 연구를 하였습니다.

upx 패킹 탐지를 확인하기 위해 대상 파일은 appcmd.exe 로 지정하였으며, 패킹 여부에 따른 entropy 경향성을 파악하기 위해 windows 10 운영체제의 system32 디렉터리에 있는 실행파일들을 entropy 를 계산하여 군집 분석을 하였습니다. appcmd.exe 대상 파일 또한 같은 방법으로 entropy 를 계산하여 시각적으로 어느 군집에 속해 있는지 그래프 상에 표시하여 entropy 를 이용한 명백한 upx packing 탐지 가능성을 보였고, 두 군집을 분명히 구분할 수 있는 mahalanobis distance 값을 찾았습니다.

본론

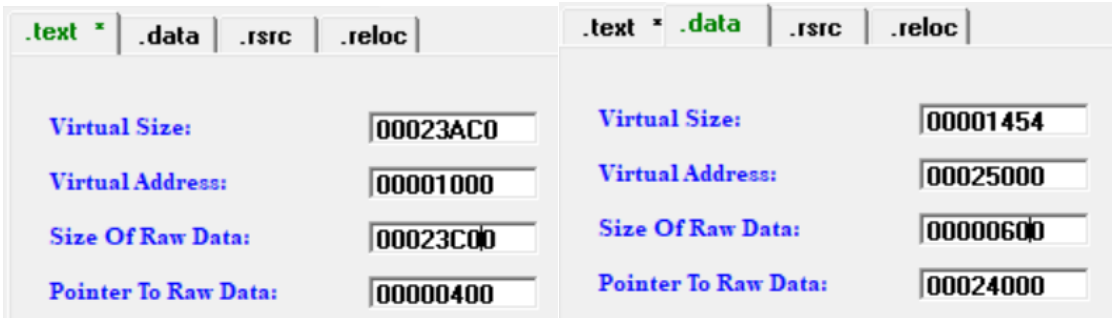
일단 appcmd.exe 파일에 대한 upx 패킹여부에 따른 entropy 그래프를 보고 비교분석을 해보았습니다. entropy 계산할 때 window size 는 256 으로 계산하였고, 이는 한 바이트당 정보의 개수가 0x00 부터 0xFF 까지 256 개임을 고려한 수치입니다.



<upx 패킹 여부에 따른 file offset 에 대한 entropy 그래프 (좌 : 원본, 우 : upx) >

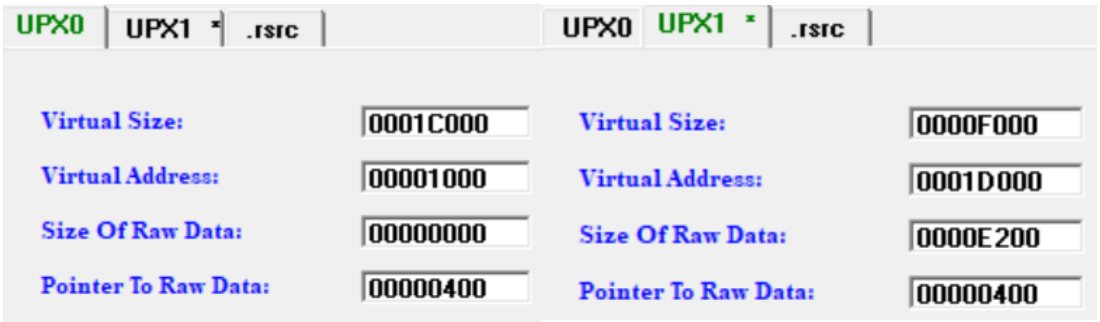
그래프 상에서의 모양 변화는 뚜렷하지만, entropy 그래프 자체만으로는 upx 패킹 여부의 명확한 근거를 찾지 못하였습니다. 하지만 특정 구간에 대해 entropy 의 분포가 좀 더 균일하고 규칙적인 분포를 나타냄을 알 수 있습니다.

해당 특정 구간은 PE 구조상의 첫번째와 두번째의 section 데이터이며, 해당 내용은 appcmd.exe 의 PE 구조를 분석하여 알 수 있었습니다. 아래는 각 섹션 헤더에 대한 PEViewer 로 본 header 정보입니다.



<appcmd.exe(원본)의 PE section header 정보 >

147456



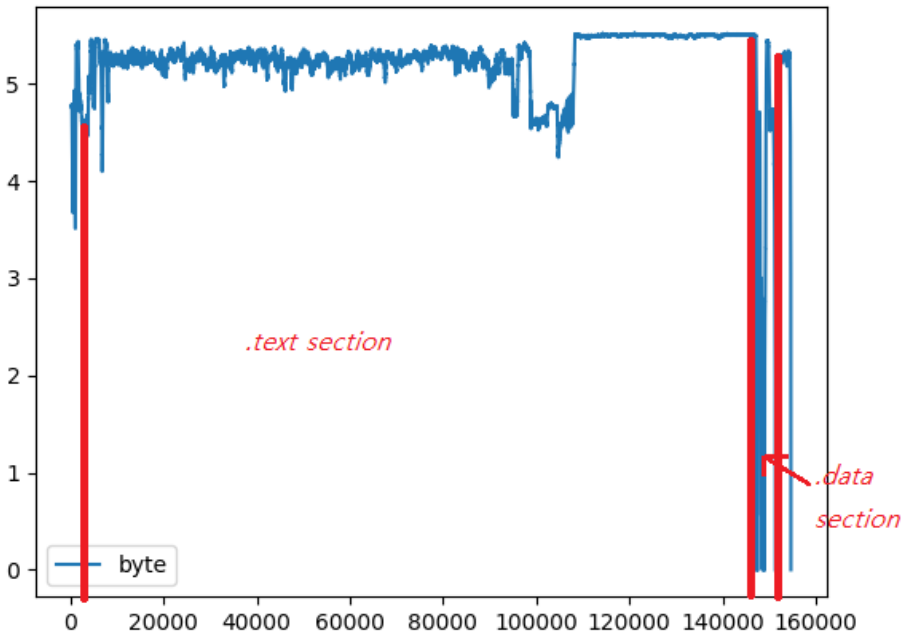
<appcmd_upx.exe(upx packed)의 PE section header 정보>

각 파일의 첫번째, 두번째 섹션 데이터가 있는 구간을 십진수로 표현하고, entropy 를 계산할 file offset 구간을 계산하면 아래의 표가 나옵니다.

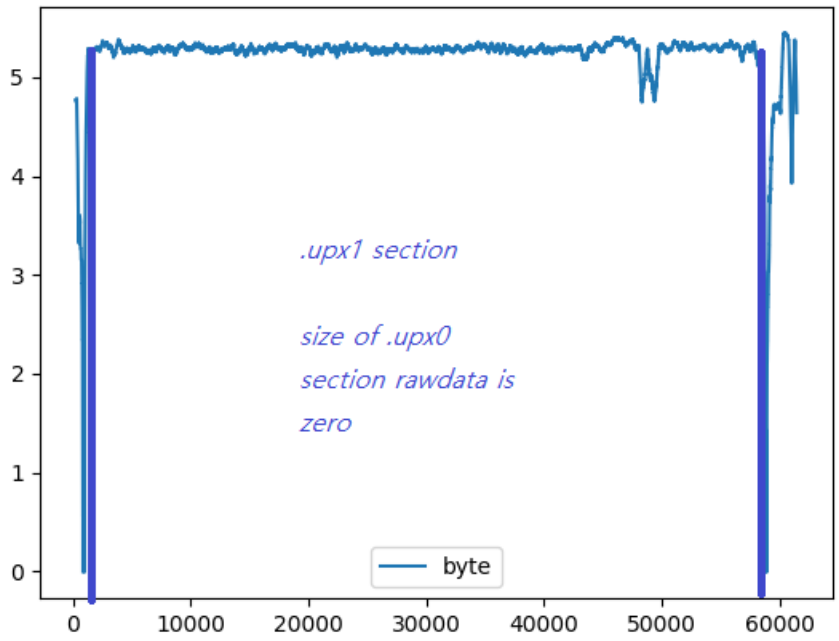
내용 / 파일	첫번째 섹션 시작주소	첫번째 섹션 사이즈	두번째 섹션 시작주소	두번째 섹션 사이즈	구간 시작	구간 끝
appcmd.exe	1024	146432	147456	1536	1024	148992
appcmd_upx.exe	1024	0	1024	57856	1024	58880

<appcmd.exe 의 첫번째, 두번째 섹션 raw data 구간>

위 표에서 구한 구간을 그래프에 다시 표시하면 아래와 같습니다.

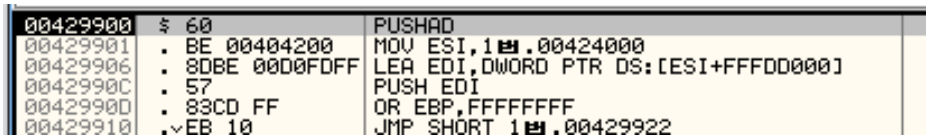


<appcmd.exe 의 첫번째, 두번째 섹션 raw data 구간>

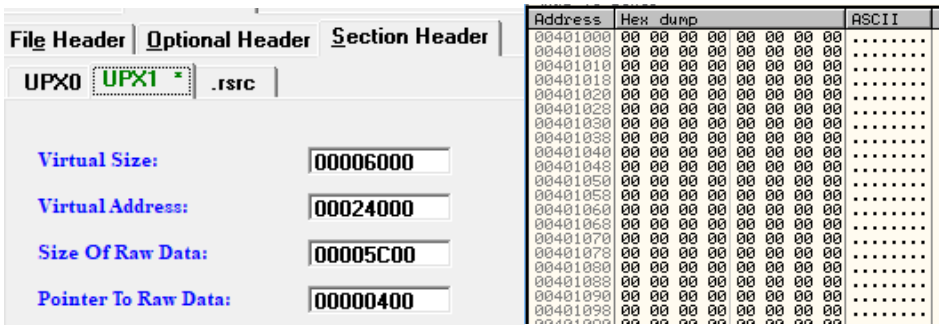


<appcmd.exe 의 첫번째, 두번째 섹션 raw data 구간>

위의 그래프에서 표시한 구간을 보면, 첫번째 섹션과 두번째 섹션의 엔트로피 분포가 패킹이 되었을 경우 상대적으로 균일하고 규칙적인 것을 알 수 있는데, 실제로 upx 패커는 원본의 코드와 데이터에 대한 압축 데이터를 .upx1 섹션에 압축을 하여 저장하기 때문에 압축 데이터의 엔트로피 특성이 나타난 것으로 볼 수 있습니다. appcmd.exe 에 대하여는 환경상 실행이 되지 않아 동적분석을 못하였지만, 다른 실행파일로 분석해본 결과 아래 사진의 두번째 instruction 에서 ESI 에 0x00424000 을 setting 하는데, 해당 주소는 upx1 섹션의 시작주소이었고, 반복적인 루프를 통해 해당 주소에서 압축을 해제한 다음, destination 주소(upx0 섹션 시작주소)에 복사하여 코드와 데이터를 복구하는 것을 알 수 있었습니다.

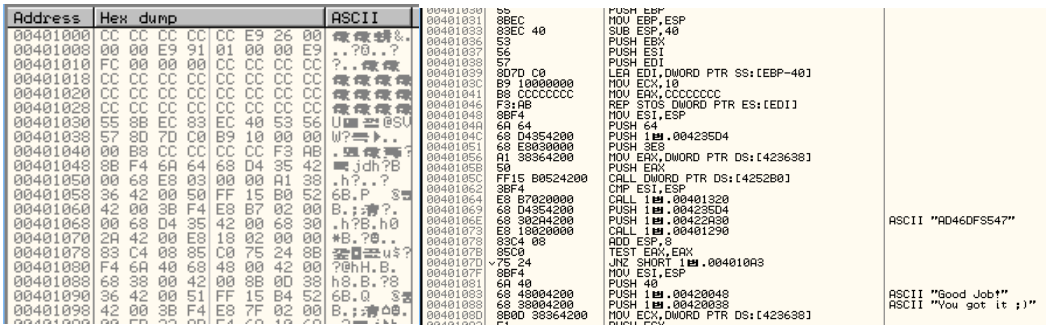


<upx 패킹이 된 바이너리의 entry point. 0x424000 은 .upx1 의 시작주소다.>



<좌측 : upx1 섹션의 헤더정보. image base(0x00400000) + virtural address(0x24000) 계산을 통해 실제 섹션이 load 되는 가상 메모리 주소를 알 수 있다.>

<우측 : upx0 섹션의 코드와 데이터가 복구되기 전 모습>

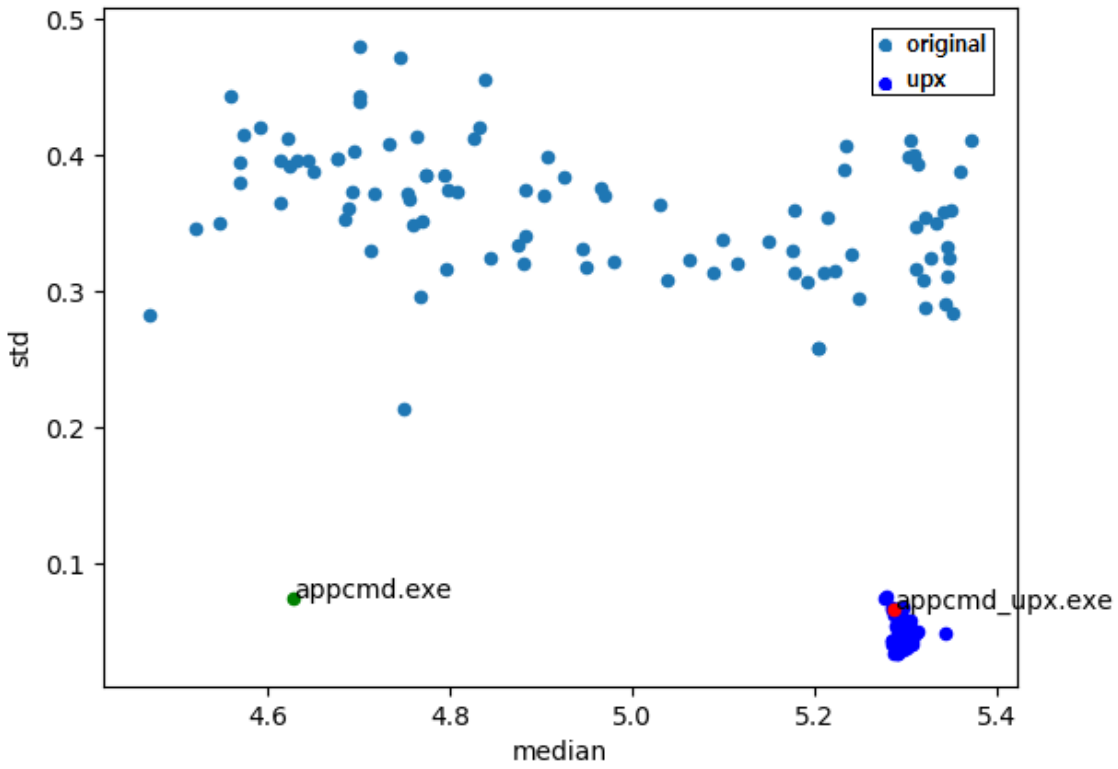


<upx0 섹션의 코드와 데이터 복구 후 모습. 실제로 disassemble 하면 instruction 이 잘 해석된다.>

이러한 압축 데이터의 엔트로피 특성을 upx 패킹 여부 탐지에 이용하고자 실행파일의 첫번째, 두번째 섹션 데이터에 대해 entropy 를 계산하고 해당 분포의 표준편차, 중간값을 계산하였습니다. 표준편차는 분포의 산포도를 나타내는 값으로서, 균일하고 규칙적으로 나오는 패킹된 실행파일의 entropy 분포에서 표준편차가 더 낮게 나올 경향이 있을 것으로 기대할 수 있고, 중간값은 패킹된 실행파일의 첫번째, 두번째 섹션의 대부분을 차지하는 압축데이터의 엔트로피 분포의 특징을 대표할 수 있는 값으로 볼 수 있습니다.

entropy 의 분포의 특징을 더 잘 나타내도록 계산하기 위해 패킹된 실행파일의 entropy 분포에 노이즈로 작용할 수 있는 첫번째 섹션과 두번째 섹션의 사이에 NULL padding 을 고려하여 entropy 를 계산하려는 256 개의 데이터 중 0 이 50% 이상 존재할 시 entropy 분포에서 제외하였습니다. 위의 방식으로 windows 10 system32 디렉터리에 있는 실행파일들 중 appcmd.exe 와 크기가 비슷한(100kb~200kb) 실행파일들에 대해 entropy 표준편차, 중간값을 계산하여 분포를 나타내고, 패킹 여부를 판단하고자 하는 appcmd.exe, appcmd_upx.exe 파일에 대해서도 같은 계산을 거쳐 scatter 그래프로 분포를 나타내어 보았습니다.

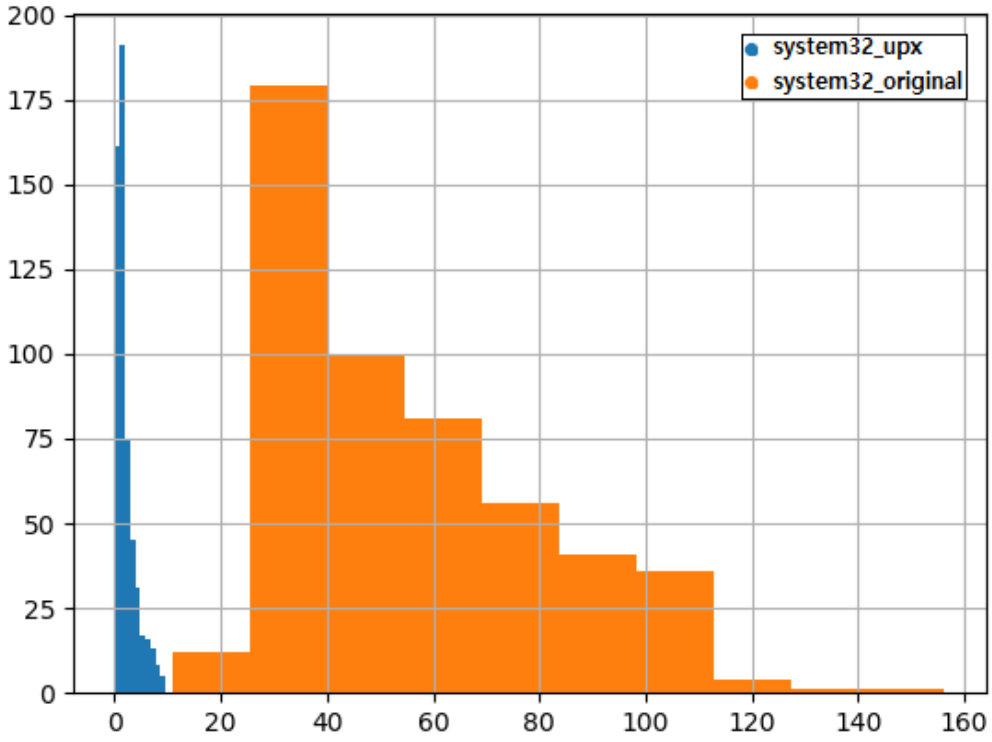
아래는 해당 scatter 그래프입니다.



<windows 10 system32 디렉터리의 100KB~200KB 파일들의 entropy 표준편차, 중간값 분포>

upx 패킹된 appcmd_upx.exe 의 경우 upx 분포에 분명히 포함되어 있는 것을 확인할 수 있고, 패킹이 되지 않은 경우 upx 군집으로부터 거리가 확연히 먼 것을 확인할 수 있었습니다. 실제로 upx 군집으로부터 각 파일의 mahalanobis distance(평균으로부터 표준편차 거리) 를 구하면 appcmd_upx.exe 는 2.549, appcmd.exe 는 88.239 로 수치적으로도 분명히 구분됨을 알 수 있습니다.

upx 군집으로부터 mahalanobis distance 를 system32 디렉터리의 모든 실행파일의 원본과 upx 패킹을 한 두 그룹에 대해 구하여 히스토그램을 그리면 아래와 같습니다.



<windows 10 system32 디렉터리의 모든 실행 파일들의 mahalanobis distance 히스토그램>

히스토그램을 보면 system32 의 upx 그룹과 원본 그룹이 분명히 구분됨을 알 수 있습니다. 두 그룹을 false positive 와 false negative 없이 구분할 수 있는 mahalanobis distance 값은 9.6 과 10.8 사이에 있음을 실험을 통해 알아냈고(각각 upx 그룹 내의 최댓값, original 그룹의 최솟값), 해당 기준으로 upx 패킹 탐지 효용성을 보였습니다.

결론 및 요약

이 연구에서 entropy 분포의 특성을 구분해 실행파일의 upx packing 여부를 판단 가능 여부를 보았고, 충분히 entropy 분포의 특성으로 upx 패킹 여부를 판단할 수 있음을 보였습니다. entropy 특성을 대표하는 값으로 중간값과 표준편차를 사용하였으며, 실제로 upx 패커의 동작을 동적분석하여 upx1 섹션에 압축데이터가 있음을 알고 이것을 고려하여 entropy 계산 시 분포의 특성에 노이즈를 없애고자 코드와 데이터가 있는 첫번째, 두번째 섹션에 대해서만 계산을 수행하였고, 섹션 사이의 null padding 을 배제하여 entropy 를 계산하였습니다. 이러한 계산 방법으로 window 10 system32 디렉터리의 appcmd.exe 와 용량이 비슷한 실행파일들에 대해 entropy 분포를 나타내었고 시각적으로 분명히 appcmd.exe 의 upx packing 여부를 판단할 수 있었습니다. 더 나아가 앞서 구한 upx 군집에서의 mahalanobis distance 를 system32 디렉터리의 모든 실행파일에 대해 원본과 upx packing 그룹으로 나눠 계산하고, 두 그룹을 분명히 나눌 수 있는 mahalanobis distance 가 존재함을 보였습니다.

참고 문헌

Using Entropy Analysis to Find Encrypted and Packed Malware, Robert Lyda (2007, IEEE)