코드 난독화
(Code Obfuscation)

15장 정보보호이론
2015 spring

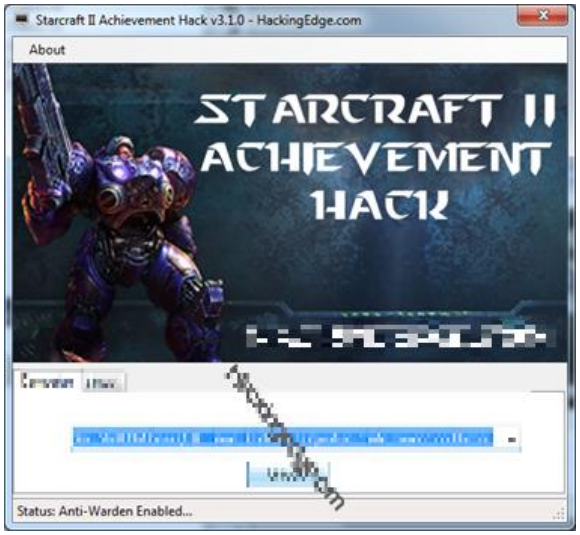
목 차

- 소프트웨어 보호의 필요성
- 코드 난독화의 기능
- 코드 난독화의 역기능
- 코드 난독화 및 역난독화 기술 연구의 현상황
- 결론
- 임베디드보안연구실에서는…

소프트웨어 보호의 필요성

소프트웨어 크랙

- 소프트웨어 지적재산권 침해 사례 증가
  - 자동화된 해킹 툴을 온라인 상에서 어렵지 않게 구할 수 있음



Reverse Engineer

소프트웨어 취약점 악용

- 코드 분석을 통하여 소프트웨어에 내재된 **취약점을 찾을 수 있음**
  - **악의적인 목적**으로 Exploit 할 경우 시스템 전체에 피해를 줄 수 있음
  - Ex) 제로데이 공격: 패치가 공개되지 않은 취약점을 이용하여 공격

소프트웨어 보호의 필요성

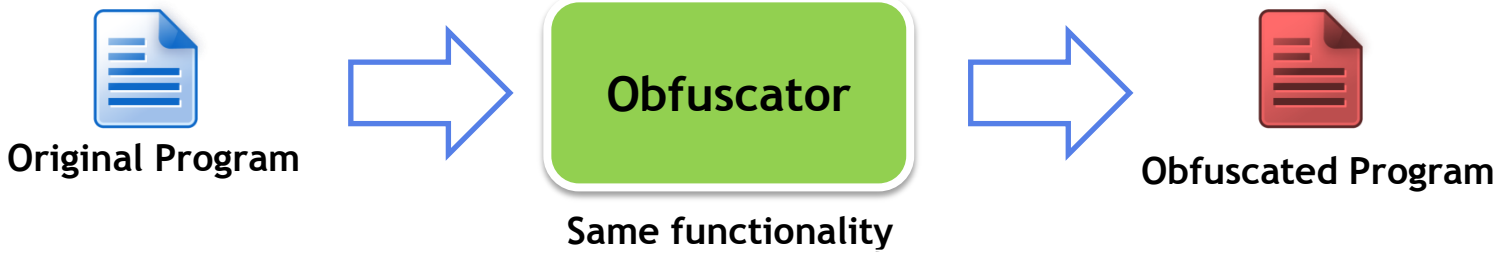
- 악의적인 목적의 지적재산권 침해나 취약점 악용을 방지하기 위함
  - 하드웨어 측면의 보호 방법
  - Ex) 동글(Dongle)
    - 동글이 장착된 컴퓨터에서만 소프트웨어 실행 가능
    - 라이선스 침해 및 무단 복사를 방지할 수 있음
    - 추가적인 비용(도구)이 발생하게 되며 분실 가능성이 있음

  - 소프트웨어 측면의 보호 방법
  - Ex) 소프트웨어 난독화, 암호화, 변조방지 등
    - 추가적인 비용은 발생하지 않음
    - 주요 알고리즘을 숨기거나 코드의 가독성을 떨어뜨리는 방법
    - 역공학에 대한 완벽한 대안은 될 수 없음 (많은 시간을 들이면 분석 가능)
    - 보호 대상 소프트웨어의 성능 저하가 발생할 수 있음

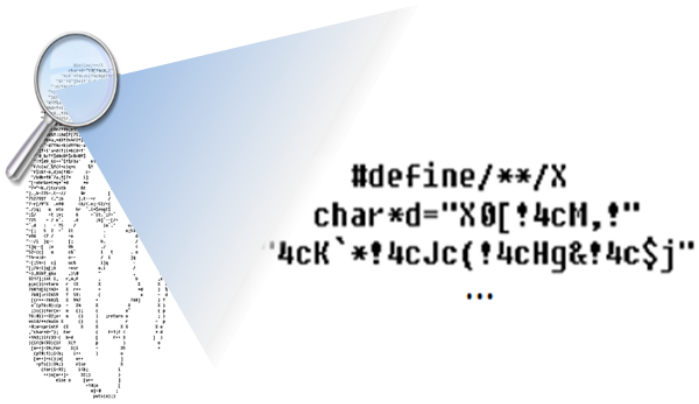
코드 난독화의 기능

난독화의 의미

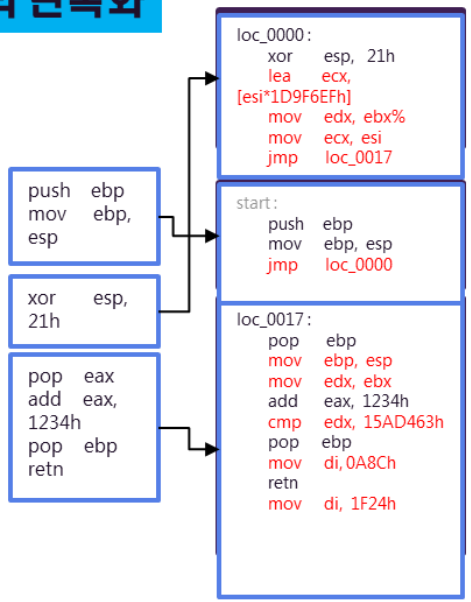
- 프로그램의 **분석을 어렵게** 하기 위해 자료구조, 제어흐름 등을 변환
  - ✘ 변환된 프로그램의 **기능성은 원본 프로그램과 동일하게 유지**



소스 코드 난독화

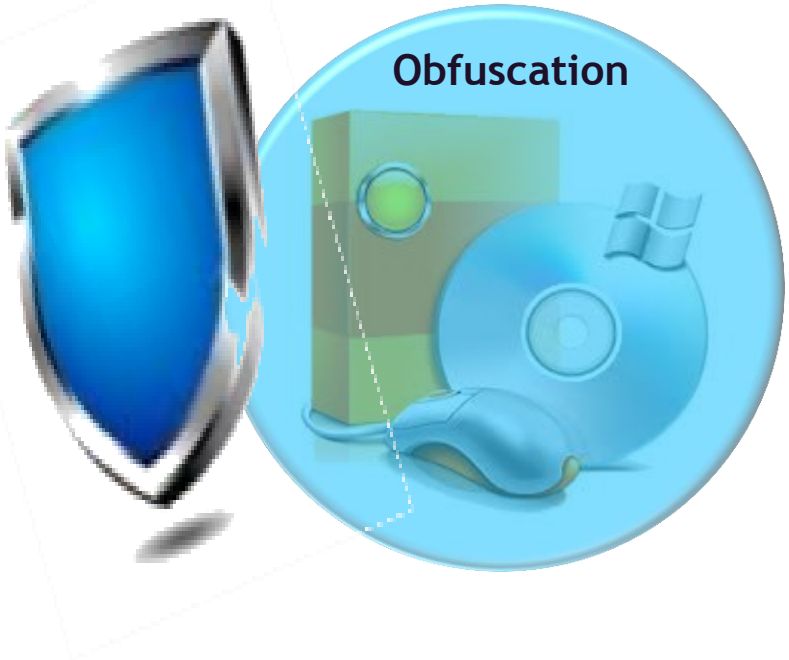


바이너리 난독화



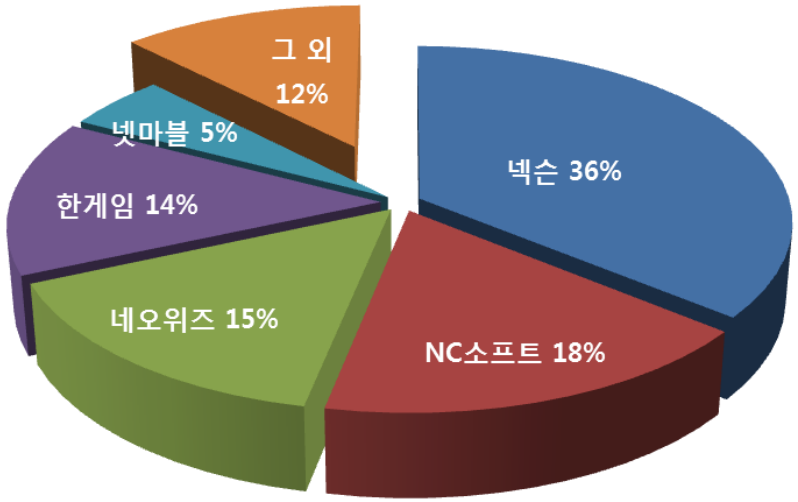
난독화의 필요성

- 소프트웨어 측면의 프로그램 보호 방법
  - 분석에 소요되는 시간을 증가시킴으로써 프로그램 보호
  - 실행 시 시간/메모리 오버헤드가 증가할 수 있음
  - 추가적인 비용을 필요로 하지 않음 (Ex: 보호용 하드웨어 구입)



국내 소프트웨어 보호 사례

- 상위 10개 국내 게임회사 및 다양한 분야에서 소프트웨어 보호 적용
  - ✘ 대부분 Themida라는 난독화 도구를 구입하여 이용



국내 게임회사는 Themida로
게임클라이언트 보호

<2012년 상위 10개 국내 게임회사의 매출 비율>
출처 : thisisgame.com, 2012

- 국내외 게임시장 및 다양한 분야에서 소프트웨어 보호를 위해 난독화 도구를 사용
  - 넥슨, NC소프트, 네오위즈, 한게임, 넷마블 등의 회사에서 서비스중인 게임,
    메신저 프로그램(카카오톡), 휴대폰 관리 프로그램(KIES, 삼성), 보안 프로그램(엑실드, 안랩) 등

난독화 기법

- 난독화 기법은 크게 4가지로 분류할 수 있음

난독화 기술

구획 난독화 (Layout obfuscation)	프로그램에는 영향을 주지 않는 i) 형식 변화, ii) 주석 제거, iii) 식별자 손상 방법 등을 이용하여 분석을 어렵도록 하는 방법
데이터 난독화 (Data obfuscation)	프로그램에서 처리하고 있는 변수(데이터)를 나누거나 합치는 방법을 사용하여 분석이 어렵도록 하는 방법
제어 난독화 (Control obfuscation)	코드의 제어 흐름을 어지럽게 함으로써 분석을 어렵도록 하는 방법
방지 난독화 (Preventive obfuscation)	역공학 도구로 사용될 수 있는 디버거, 메모리 덤프 도구 등을 무력화하기 위한 방법

난독화 기법의 예 [1/3]

- Identifier Obfuscation
  - 함수명 혹은 변수명을 의미 없는 문자(a,b,c…)로 치환하여 가독성 저하
- Comment Removal
  - 소스코드에 작성되어 있는 주석을 제거하여 가독성 저하

```
private void a(string A_0, string A_1)
{
    DirectoryInfo info = new DirectoryInfo(A_0);
    DirectoryInfo info2 = new DirectoryInfo(A_1);
    info2.Create();
    DirectoryInfo[] directories = info.GetDirectories();
    FileInfo[] files = info.GetFiles();
    for (int i = 0; i < files.Length; i++)
    {
        try
        {
            files[i].CopyTo(info2 + "/" + files[i].Name);
        }
        catch
        {
        }
    }
    for (int j = 0; j < directories.Length; j++)
    {
        this.a(directories[j].FullName, A_1 + "/" + directories[j].Name);
    }
}
```

Identifier Obfuscation

```
//=======================================================
//  THIS BUTTON GETS INFO[illegible]ROM A TEXT BOX
//=======================================================
private void btnStrings_Click(object sender, EventArgs e)
{
    string firstName;
    string messageText;

    messageText = "Your name is: ";

    //=======================================
    //  GET THE TEXT[illegible]F TEXT BOX
    //=======================================
    firstName = textBox1.Text;

    //MessageBox.Sh[illegible]geText + firstName);


}
```

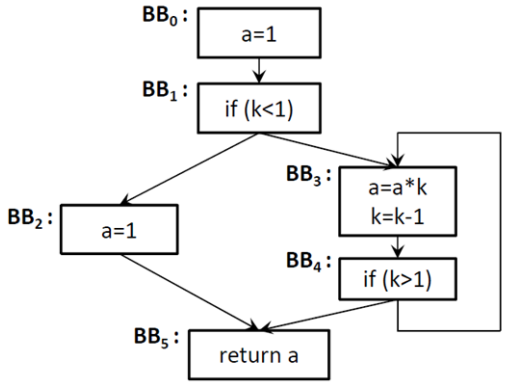
Comment Removal

난독화 기법의 예 [2/3]

Control Flow Flattening

- 제어흐름을 분석하기 어렵게 제어흐름 그래프를 평탄화시키는 기법

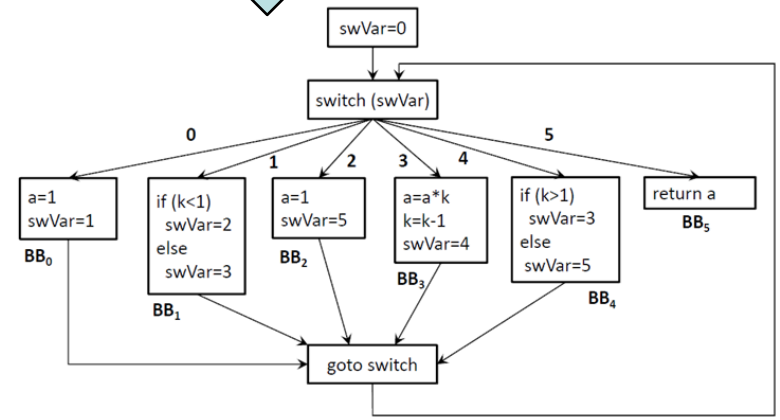
```
int fun(int k)
{
  int a=1;
  if (k<1)
    a=1;
  else
    do{
      a *= k--;
      }while (k>1);
  return a;
}
```



Control Flow Flattening 적용

<제어흐름 평탄화 기법이 적용되기 전 함수의 소스 코드와 제어흐름 그래프>

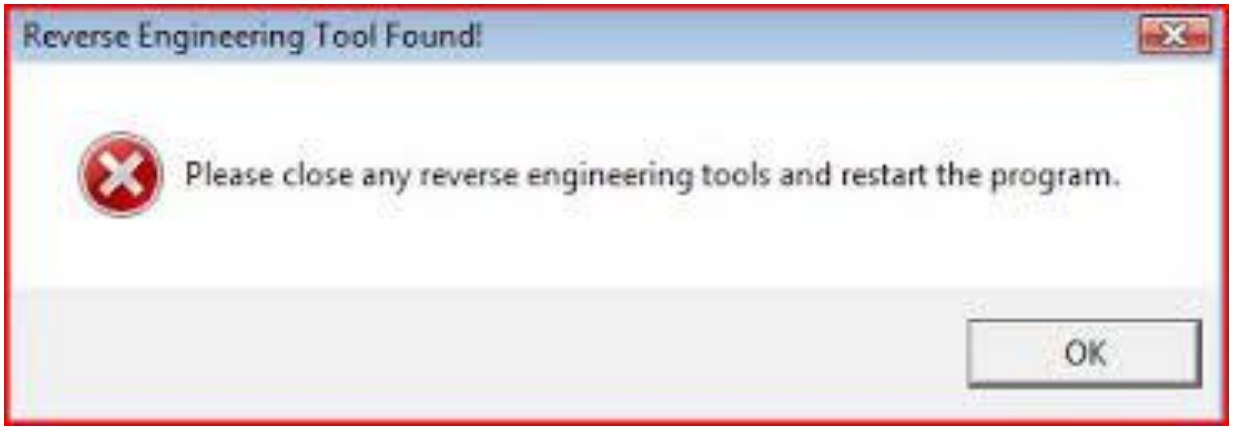
```
int fun(int k) {
  int swVar=0;
  for( ; ; )
    switch(swVar) {
      case 0 : a=1; swVar=1; break;
      case 1 : if (k<1) swVar=2; else swVar=3; break;
      case 2 : a=1; swVar=5; break;
      case 3 : a=a*k; k=k-1; swVar=4; break;
      case 4 : if (k>1) swVar=3; else swVar=5; break;
      case 5 : return a;
    }
}
```



<제어흐름 평탄화 기법이 적용된 후 함수의 소스 코드와 제어흐름 그래프>

난독화 기법의 예 [3/3]

- Anti Debugging
  - 프로그램을 동적 분석할 수 있는 디버거를 사용하지 못하게 하는 기법
  - Anti Debugging 기법이 적용된 프로그램을 그냥 실행할 경우엔 실행됨
    - 그러나 디버거와 함께 실행하면 올바르게 동작하지 않음



Anti Debugging 옵션이 적용된 프로그램을
디버거와 함께 실행한 경우

코드 난독화의 역기능

역난독화 연구의 필요성

- 난독화 기법의 역기능
  - 악성코드 제작자가 자신의 악성코드를 보호하기 위해 악용 가능(미탐)
  - 일반 프로그램이 난독화된 경우 안티바이러스에서 악성코드로 진단(오탐)

virustotal

SHA256: 14283b1093782caa4346a737a770ce2ba041bd558092ad9665f45867a5a2cec5

파일 이름: Obfuscated_Program.exe 난독화된 정상 프로그램

탐지 비율: 12 / 49

분석 날짜: 2014-03-12 01:49:47 UTC (1분 전)

난독화된 프로그램 내부 구조를 분석할 수 없음
→ 악성코드로 오탐

안티바이러스	결과
Ad-Aware	Gen:Trojan.Heur.RP.YuX@a8vZ9Qm
AntiVir	TR/Crypt.XPACK.Gen
BitDefender	Gen:Trojan.Heur.RP.YuX@a8vZ9Qm
Bkav	W32.HfsAutoB.83b2
Comodo	TrojWare.Win32.Agent.COC
Emsisoft	Gen:Trojan.Heur.RP.YuX@a8vZ9Qm (B)
F-Secure	Gen:Trojan.Heur.RP.YuX@a8vZ9Qm
GData	Gen:Trojan.Heur.RP.YuX@a8vZ9Qm
Malwarebytes	Trojan.Agent
McAfee-GW-Edition	Heuristic.LooksLike.Win32.Suspicious.J
MicroWorld-eScan	Gen:Trojan.Heur.RP.YuX@a8vZ9Qm

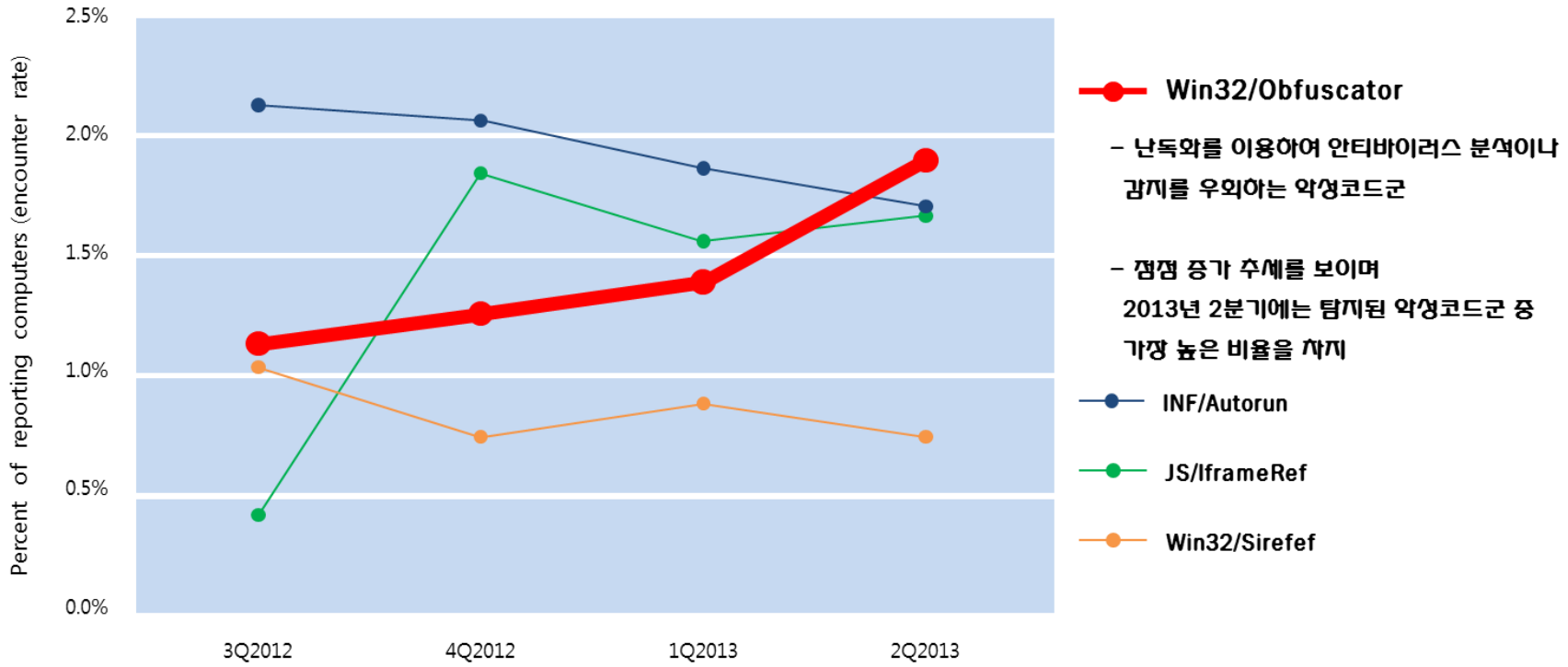
Obfuscation

난독화된 악성코드 내부 구조를 분석할 수 없음
→ 미탐

난독화된 악성코드 동향

- 난독화 기법을 이용하여 안티바이러스 탐지를 우회하는 악성코드군 증가

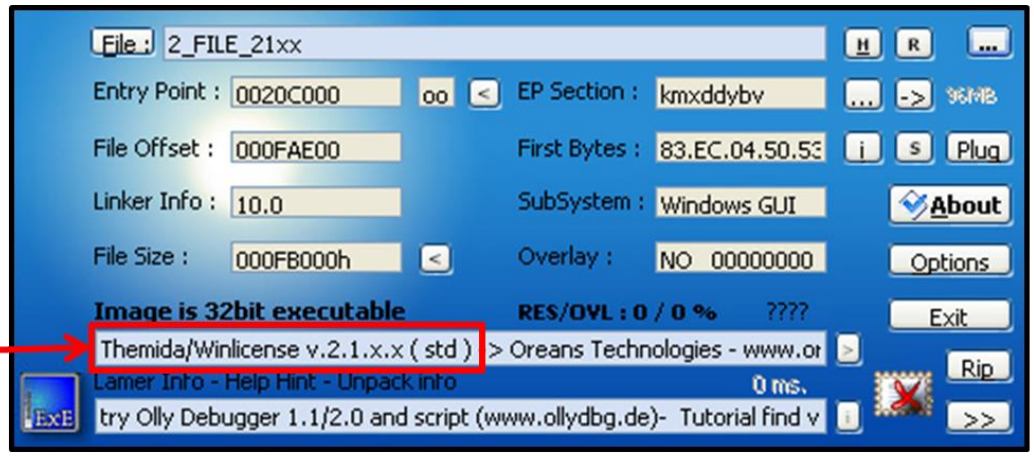
출처 : Microsoft Security Intelligence Report, Volume 15



난독화 기법 악용 사례

- 6.25 사이버테러
  - 사이버테러 분석 결과, 악성코드에 난독화 기법이 적용되어 분석지연
  - 악성코드가 난독화된 경우 이를 역난독화할 때 까지 분석이 불가능
    - 역난독화 기술 연구가 중요한 이유

Themida를 이용하여
난독화 기법 적용



<6.25 사이버테러에 사용된 악성코드>

- 2013년 6월 25일 청와대 및 주요 정부기관의 홈페이지를 대상으로 사이버테러 발생
- 공격자는 웹사이트 변조, DDoS, 신상정보 유출 등의 다양한 공격을 수행
- 이 공격을 수행한 악성코드는 상용 난독화 도구인 Themida에 의해 난독화되어 탐지 및 분석에 많은 시간이 소요되었음

난독화 기술과 역난독화 기술

- 난독화 기술과 역난독화 기술의 순기능과 역기능(요약)

순기능과 역기능

	난독화 기술	역난독화 기술
순기능	코드 난독화 기술은 적법한 소프트웨어 보호를 위한 도구로 사용됨	난독화 되어 있는 악성 코드의 분석을 통해 피해를 최소화할 수 있음
역기능	소프트웨어를 보호할 수 있는 기능을 악용하여 악성 코드의 분석을 어렵게 하는데 사용됨	적법한 소프트웨어의 지적 재산 보호를 위해 사용된 난독화를 무력화 시키는데 사용될 수 있음

난독화 코드 분석의 필요성

난독화 도구를 악용하여
공격 코드의 보호용도로 사용

난독화 도구 및 활용 용이
- 상용 난독화 도구들을 누구나 쉽게 구매하여 난독화 가능

난독화된 악성 코드 증가
- 최근 Themida, VMProtect와 같은 상용 난독화 도구를 적용한 악성코드가 급속히 증가 하고 있음

분석 시간 지연
- 난독화된 악성 코드를 기존 방식으로 상세 분석 할 경우 기존보다 수십~수백 배의 분석시간이 소요됨

코드 난독화 및 역난독화 기술 연구의 현상황

난독화 기술 연구의 현상황

- 난독화 도구는 국내에서도 개발이 되고 있음
  - 국내 난독화 도구의 보호 강도 검증 방안 미흡
  - 그렇기 때문에 아직 외산 기술/도구에 의존하고 있음(Ex: Themida)

- 난독화 도구의 보호 강도가 검증되지 못하는 이유
  - 난독화 기법의 보호 강도 측정 방안에 대한 연구가 미비하기 때문
  - 앞으로는 보호 강도의 정량적인 측정 방안 연구가 필수적임

Themida에서 제공하는 보호 옵션(일부)

역난독화 기술 연구의 현상황

- 역난독화 기술은 난독화 기술에 비하여 많이 연구되지 않음
  - 역난독화를 전문적으로 하는 도구는 상용화된 바 없음
  - 학술적인 연구 차원의 역난독화 기술 연구 결과는 발표되고 있음
- 자동화된 역난독화를 위해서는 해당 난독화 기법의 분석이 필요
  - 그러나 난독화 기법 분석 자체가 굉장히 어려운 일임
  - 자동화된 역난독화 기술이 연구되기 어려운 이유
- 난독화 도구는 상용화된 반면 역난독화 도구는 극히 제한적임
  - 한번에 하나의 기술(Ex: UPX Packing)만을 탐지하고 역난독화 가능
  - 그러나 난독화 도구는 한번에 여러 난독화 기법을 적용할 수 있게 함
  - 앞으로는 여러 기법에 대한 역난독화 방안 연구가 필수적임

결론

결론

- 소프트웨어 지적재산권 보호를 위하여 난독화 기법 연구
- 그러나 악성코드에 적용되어 악성코드 분석이 어려워짐
  - 역난독화 기법의 필요성 증가
- 난독화와 역난독화는 서로 칼과 방패와 같은 관계임
  - 현재까지는 난독화 기술 연구가 더 많이 진행되어옴
- 난독화와 역난독화 모두 국내에서 많은 연구가 필요하며…
  - 난독화 기술의 보호 강도 측정방안과
  - 여러 난독화 기법에 동시 대응할 수 있는 역난독화 기술의 연구가 필요

임베디드보안연구실에서는…

역난독화 기술 연구[PC]

- Themida에 의해 난독화된 프로그램의 자동 역난독화 방안 연구
  - 분석환경 구축 및 각 난독화 기법 특징 추출
  - 연구결과
    - 프로그램이 Themida에 의해서 난독화 되었는지 여부와
    - (Themida에 의해서 난독화 된 경우) 어떤 옵션이 적용되었는지 식별가능

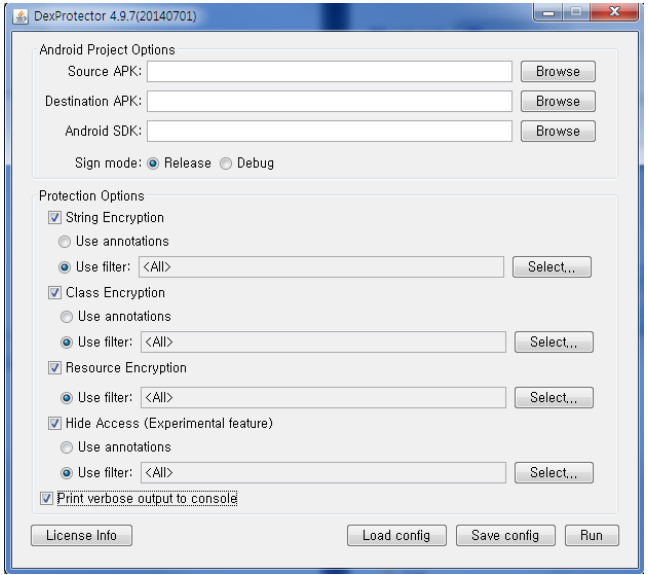
- 각 난독화 기법을 상세 분석 중이며 이의 최적화 방안 연구
  - 자동화된 역난독화 방안 연구 예정(각 옵션별 상세 분석 요구)

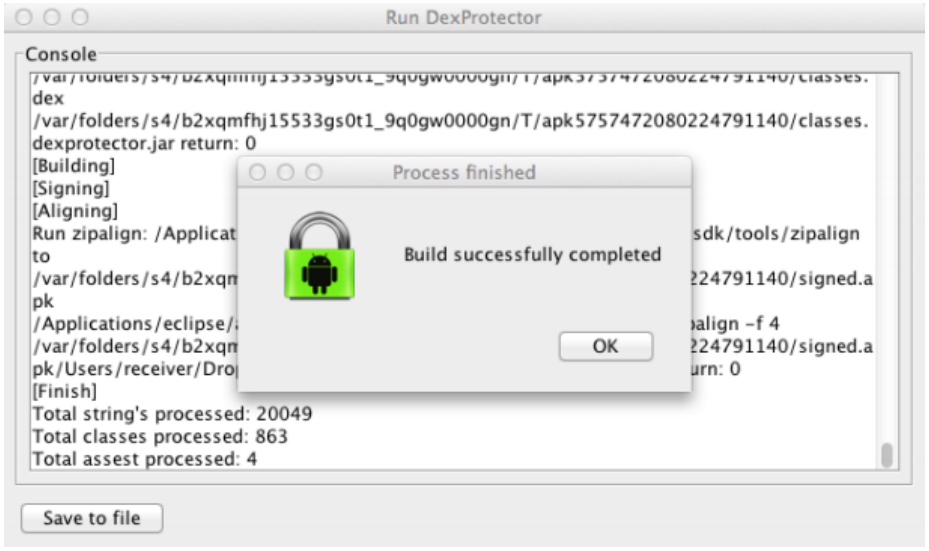
2012년 기준 국내 게임회사 중 매출 순위 상위 5개 회사가 Themida를 사용하여 게임 클라이언트 보호

역난독화 기술 연구[Mobile]

- DexProtector에 의해 난독화된 APK 파일 자동 역난독화 방안 연구
  - 분석환경 구축 및 각 난독화 기법 특징 추출
  - 연구결과
    - (DexProtector에 의해 난독화 된 경우) 어떤 옵션이 적용되었는지 식별가능
    - 난독화 기법 중 암호화 관련옵션의 복호화 루틴 분석 완료(역난독화 가능)

- DexProtector에 대응하는 자동화된 역난독화 도구 제작 예정





THANK YOU