Unpacking 알고리즘 특징 기반의 Packer 탐지 시그니처 생성 방안

신동휘 O, 임채태 정현철

한국인터넷진흥원

shindh@kisa.or.kr, chtim@kisa.or.kr, hcjung@kisa.or.kr

The packer detection signature generation based on unpacking algorithm characteristic

Donghwi Shin O, Chaetae Im, Hyuncheol Jeong

Korea Internet & Security Agency

요 약

악성코드의 기능들이 날로 정교해 지면서 악성 행위를 숨기거나 악성코드 분석이 어렵도록 만들기 위한 기법들이 적용되는 것을 쉽게 볼 수 있다. 이 중 악성코드 분석을 어렵게 만드는 대표적인 방식이 Packing이다. 그러므로 악성코드의 분석을 위해 Packing된 악성코드가 어떤 Packer로 Packing되어 있는지 확인할 필요가 있다. 그러나 현재 사용하는 대부분의 시그니처 기반 탐지 방식은 오탐율 및 미탐율이 높다. 본 논문에서는 Packer 탐지를 위한 새로운 시그니처 생성 방식을 제안하고 성능을 검증한다.

1. 서 론

악성코드의 수가 급격히 증가함에 따라 개인 PC, 서버, 네트워크에 큰 위협으로 부각되고 있다. 최근 악성코드는 자신의 행위가 노출되지 않고 오래 지속되는 것이 악성코드 제작자들에게 금전적인 이익을 가져다 주기 때문에 악성코드를 숨기거나 분석을 어렵게 하려고 한다. 이러한 현상은 최근 수집되는 악성코드의 92%는 Packing되어 있다는 간단한 통계만으로도 쉽게 확인할 수 있다. 이와 같이 최근 수집/발견되는 악성코드의 대부분이 Packing되어 있으므로 악성코드 분석을 위해 Packer의 탐지는 필수 요소이다.

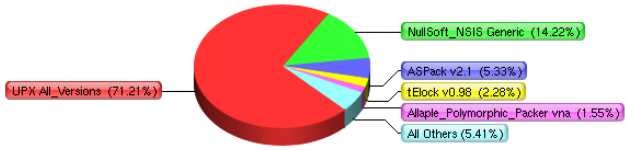

그림 1 최근 3개월 Packer 사용 현황(ShadowServer)[1]

그러나 현재 PEiD[2]의 경우 탐지율이 약 60%정도에 불과하며 SigBuster도 약 71%의 탐지율을 보이고 있다. 물론 사용되는 Packer의 종류가 위의 그림1과 같이 대부분 UPX와 같이 널리 알려진 Packer이지만 Packing 알고리즘은 나날이 발전하고 있으므로 Packer 탐지율을 높일 수 있는 새로운 방식이 필요하다. 그러므로 본 논문에서는 기존의 방식에서 벗어나 Packer들 마다 사용되는 Unpacking의 특징을 활용한 탐지 시그니처 생성 방법을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 Packer 탐지를 위한 Tool에 대한 소개 및 PEiD에서 사용되는 시그니처 생성 방법을 정리하고, 3장에서는 테스트를 위한 샘플을 대상으로 Packer 탐지 Tool의 성능을 검증하고, 4장에서는 제안하는 시그니처 생성 방식과 그 성능을 검증하며, 마지막으로 5장에서 결론을 맺는다.

2. Packer 탐지 Tool

본 장에서는 Packer 탐지에 사용되는 주요 Tool들과 이 Tool들 중, PEiD의 시그니처 생성 방법을 설명한다.

2.1 PEiD

PEiD는 PE 파일의 정보를 분석하기 위한 Tool로 Packer를 탐지하기 위해 시그니처 방식을 사용하고 있다. PEiD는 시그니처 생성 방식은 EP(EntryPoint)와 Section 정보를 기준으로 Hex Sequence를 시그니처로 사용한다.

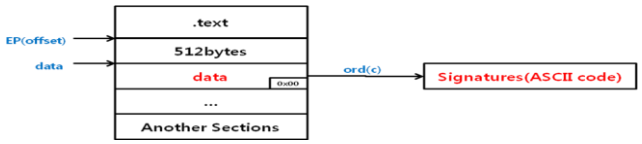

그림 2 PEiD EP 기반 시그니처 생성

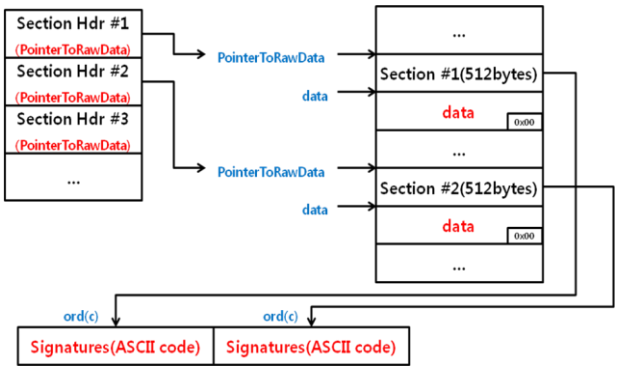

그림 3 PEiD Section 정보 기반 생성

2.2 Detect It Easy & Exeinfo PE

Detect It Easy(DIE)[3]는 러시아에서 PE파일 분석을 위해 만들어진 Tool이다. PE 파일 분석을 위해 다양한 정보를 확인할 수 있으며 Plug-in 기능을 제공하여 확장된 기능을 사용할 수 있다. 또한 Exeinfo PE[4]도 DIE와 유사하며 다양한 확장 기능을 제공한다.

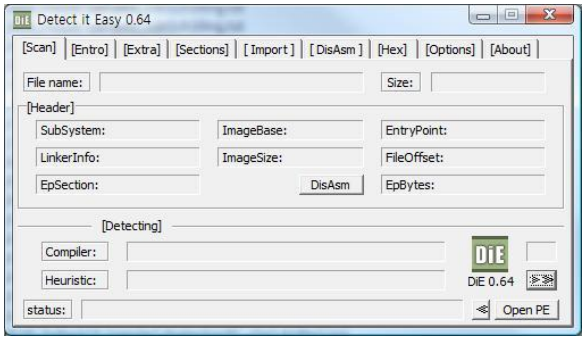

그림 4 Detect It Easy

2.3 Protection ID[5]

앞서 설명한 PEiD, DIE, Exeinfo PE는 PE 파일 분석을 목적으로 하는 Tool인 반면, Protection ID는 PE파일 분석보다는 실행되고 있는 시스템의 전반적인 정보를 분석하는 Tool이며 제공 기능들 중에 PE파일을 분석하는 기능이 포함되어 있는 것이다.

3. Packer 탐지 Tool 실험 결과

본 장에서는 Packer 탐지에 사용되는 주요 Tool들을 사용하여 다양한 Packer로 Packing된 실행파일을 대상으로 각각의 툴에 대한 Packer 탐지율, 오탐율, 미탐율을 측정한다. Packer 탐지율, 오탐율, 미탐율을 측정하기 위한 기준으로 정확한 Packer 종류와 버전 두 가지를 사용한다.

Packer 탐지 Tool의 성능측정을 위해 사용할 Packer 탐지 Tool은 PEiD, Detect It Easy, Exeinfope, Protection ID 모두 4가지 이다. 그리고 샘플은 Packing되지 않은 원본파일 3개를 임의의 Packer 11개로 Packing한 실행파일을 사용한다.

3.1 Pack 종류 탐지율(미탐율)

Packer 종류 탐지율은 4개의 Packer 탐지 Tool을 사용하여 11개의 Packer로 Packing한 샘플을 대상으로 Packer 종류를 식별하는 확률을 의미한다. Packer 종류 탐지율 실험 결과는 표1과 같다. 실험 결과를 보면 52%의 탐지율과 58%의 미탐율을 보이는 PEiD를 제외한 다른 탐지 Tool은 모두 Packer의 종류를 정확하게 식별할 수 있었다.

표 1 Packer 종류 탐지율(미탐율)

Packer	PEiD v0.95	DIE v0.64	Exeinfope v0.0.2.6	ProtectionID v0.6.3.5
UPX3.03	2(1)	3(0)	3(0)	3(0)
NsPack2.3	0(3)	3(0)	3(0)	3(0)
NsPack3.4	0(3)	3(0)	3(0)	3(0)
FSG1.33	3(0)	3(0)	3(0)	3(0)
FSG2.0	3(0)	3(0)	3(0)	3(0)
AsPack2.12	3(0)	3(0)	3(0)	3(0)
Themida2.0.3.0	0(3)	3(0)	3(0)	3(0)
Themida1.9.9.0	0(3)	3(0)	3(0)	3(0)
Armadillo 5.42	1(2)	3(0)	3(0)	3(0)
ASProtect SKE 2.1	3(0)	3(0)	3(0)	3(0)
ASProtect SKE 2.4	2(1)	3(0)	3(0)	3(0)
탐지율(미탐율)(%)	52(48)	100(0)	100(0)	100(0)

3.2 Pack 버전 탐지율(미탐율)

Packer 버전 탐지율은 4개의 Packer 탐지 Tool을 사용하여 11개의 Packer로 Packing한 샘플을 대상으로 Packer의 버전을 식별하는 확률을 의미한다. 실험결과는 표2와 같으며 Packer 종류 탐지율에 비해 Packer 버전 탐지율이 떨어지는 것을 확인할 수 있다.

표 2 Packer 버전 탐지율-오탐율(미탐율)

Packer	PEiD v0.95	DIE v0.64	Exeinfope v0.0.2.6	ProtectionID v0.6.3.5
UPX3.03	0-0(3)	3-0(0)	2-1(0)	3-0(0)
NsPack2.3	0-0(3)	0-3(0)	3-0(0)	0-3(0)
NsPack3.4	0-0(3)	0-3(0)	0-3(0)	3-0(0)
FSG1.33	3-0(0)	3-0(0)	3-0(0)	0-3(0)
FSG2.0	3-0(0)	3-0(0)	3-0(0)	0-3(0)
AsPack2.12	3-0(0)	3-0(0)	3-0(0)	3-0(0)
Themida2.0.3.0	0-0(3)	0-0(3)	0-3(0)	2-1(0)
Themida1.9.9.0	0-0(3)	0-0(3)	0-3(0)	0-3(0)

Armadillo 5.42	0-0(3)	0-0(3)	0-0(3)	0-3(0)
ASProtect SKE 2.1	3-0(0)	0-3(0)	0-3(0)	0-3(0)
ASProtect SKE 2.4	0-0(3)	0-0(3)	0-3(0)	0-3(0)
탐지율-오탐율 (미탐율)(%)	36-0 (64)	36-27 (37)	42-39 (19)	33-67 (0)

4. Unpacking 알고리즘 기반 탐지 알고리즘

4.1 Unpacking 알고리즘 기반 탐지 알고리즘

Packer의 기본적인 기능은 실행파일의 형태를 변형하기 위한 Packer 고유의 알고리즘을 활용하여 실행파일을 Packing 하는 것이다. 즉, Packer는 실행파일의 형태를 변형하기 위한 각자 고유 알고리즘을 갖고 있다.

예를 들어 Packer의 종류가 A이고 버전이 1.0인 Packer A v1.0라 하자. Packing되기 이전의 두 개의 실행파일을 각각 P, Q라 하고 실행파일 P와 Q를 Packer A v1.0로 Packing한 결과를 P'(Eq1), Q'(Eq2)라고 가정한다. P'을 실행하면 P' 에 저장된 Packer A v1.0 고유의 Unpacking 알고리즘을 통해 P의 코드 영역이 드러나고 해당 코드 영역을 실행한다. Q'의 경우에도 P'의 실행과정과 동일하게 Q'에 포함된 Packer A v1.0 고유의 Unpacking 알고리즘이 실행된 후, Q의 코드가 실행된다.

P'(with Packer A v1.0 Unpacking Algorithm)
= Packer A v1.0(P) – Eq1
Q'(with Packer A v1.0 Unpacking Algorithm)
= Packer A v1.0(Q) – Eq2

앞서 Packer들은 각각 고유의 알고리즘을 갖고 있다고 했으므로 Packing된 P'과 Q'에 저장되어 있는 Packer A v1.0의 Unpacking 알고리즘은 Packer A v1.0의 특징을 포함하고 있다. 그리고 P'과 Q'는 동일한 Packer A v1.0을 통해 Packing되었으므로 P'과 Q'에 포함된 Unpacking 알고리즘은 P와 Q에 종속적인 부분을 제외하고 상당히 유사한 속성을 갖는다.

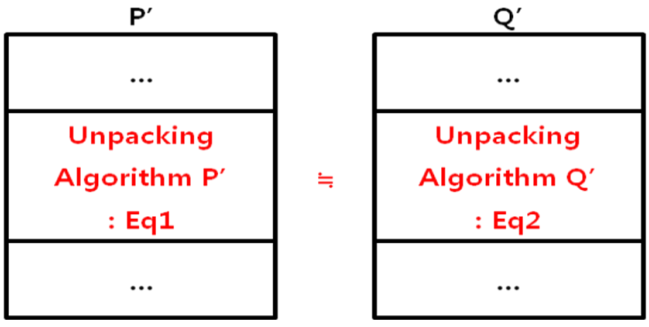

그림 5 P'과 Q'의 Unpacking Algorithm

결국 Packing된 P'과 Q'에 저장된 Packer A v1.0 Unpacking 알고리즘의 Operator Sequence는 거의 유사할 수 밖에 없다. 그러므로 동일한 Packer로 Packing된 실행파일에 포함된 Unpacking 알고리즘 영역은 유사한 Operator의 Sequence라는 특징을 활용하여(Eq3, Eq4) 이 부분을 Packer 탐지(Eq5)를 위한 시그니처로 활용한다.

Packer A v1.0 Signature(P') = OperatorSequence
(Packer A v1.0 Unpacking Algorithm) – Eq3
Packer A v1.0 Signature(Q') = OperatorSequence
(Packer A v1.0 Unpacking Algorithm) – Eq4
Packer A v1.0 Signature(P') ≒
Packer A v1.0 Signature(Q') – Eq5

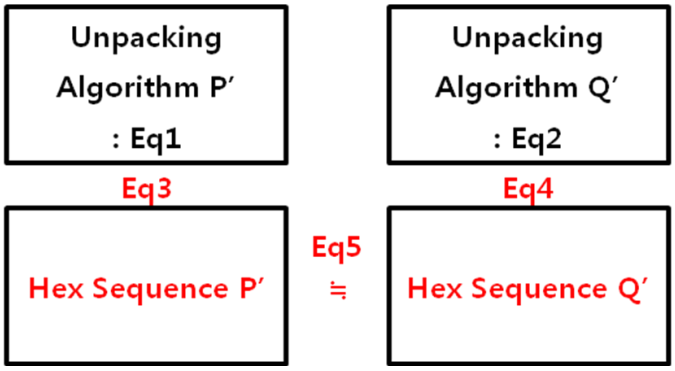

그림 6 Signature Generation & Detection

이제 P'과 Q'을 대상으로 Signature 생성 및 비교 방법을 설명한다. 일단 Signature 비교를 위해 기초 시그니처를 생성해야 한다. 여기서는 기초 시그니처 생성을 위해 P'으로부터 Packer A v1.0 Unpacking 알고리즘 영역의 Hex Sequence를 추출한다. 추출할 Hex Sequence는 그림7의 붉은 영역이다.

그림 7 Packer Signature Extraction Area

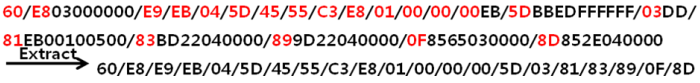

그림 8 Signature Extraction

위의 그림7에서 추출한 Hex Sequence에서

Operator의 역할을 하는 처음 2bytes를 추출하고 그림7과 같이 다시 Hex Sequence로 정리한다. 이 Hex Sequence를 Packer 탐지 시그니처로 활용한다. 즉, 시그니처 생성 방식은 다음과 같이 정의할 수 있다.

Signature = ∑_i{LEFT(Assemble Code, 2)_i}
LEFT(str, n) : Extract n bytes from a left side of str
i : number of unpacking algorithm line

4.2 Unpacking 알고리즘 기반 탐지 알고리즘 테스트

본 절에서는 4.1에서 정의한 시그니처 생성 방식을 적용하여 3.1, 3.2에서와 같이 Packer 종류와 버전을 탐지한다. 테스트 샘플은 3.1과 3.2과 동일하게 모두 11개의 Packer로 Packing된 3개의 샘플을 사용한다. 그리고 본 절에서는 Themida 2.0.3.0과 UPX 3.03을 대상으로 제안하는 시그니처 생성 및 탐지의 방식을 설명하고 11개의 Packer를 대상으로 테스트한 결과를 보인다.

첫째, UPX 3.03로 Packing된 샘플 3개를 대상으로 제안하는 알고리즘을 적용한다. 이전의 테스트에서 사용한 3개의 샘플을 각각 Ollydbg로 로딩하고 동일하게 EntryPoint부터 Unpacking 알고리즘의 끝까지 Hex Sequence를 추출한다. 그리고 추출된 Hex Sequence로부터 처음 2 Bytes를 재 추출 한다.

둘째, Themida 2.0.3.0로 Packing된 샘플 3개를 대상으로 UPX 3.03과 동일한 방식을 적용하여 Hex Sequence를 추출한다.

표 3 UPX 3.03 Hex Sequence

Sample1(303)	Sample2(303)	Sample3(303)
60	60	60
BE 00E04300	BE 00000101	BE 00204200
8DBE 0030FCFF	8DBE 0010FFFF	8DBE 00F0FDFF
57	57	57
83CD FF	83CD FF	EB 0B
EB 10	EB 10	90
90	90	8A06
90	90	46
90	90	8807
90	90	47
90	90	01DB
90	90	75 07
8A06	8A06	8B1E
46	46	83EE FC
8807	8807	11DB
47	47	72 ED
01DB	01DB	B8 01000000
75 07	75 07	01DB
8B1E	8B1E	75 07
83EE FC	83EE FC	8B1E
11DB	11DB	83EE FC
72 ED	72 ED	11DB
B8 01000000	B8 01000000	11C0
01DB	01DB	01DB

표 4 UPX 3.03 Hex Sequence(2bytes)

2bytes(Sample1)	2bytes(Sample2)	2bytes(Sample3)
60	60	60
BE	BE	BE
8D	8D	8D
57	57	57
83	83	EB
EB	EB	90
90	90	8A
90	90	46
90	90	88
90	90	47
90	90	01
90	90	75
8A	8A	8B
46	46	83
88	88	11
47	47	72
01	01	B8
75	75	01
8B	8B	75
83	83	8B
11	11	83
72	72	11
B8	B8	11
01	01	01

UPX 3.03로 Packing한 샘플을 대상으로 테스트한 결과 Unpacking 알고리즘으로부터 2 Bytes를 추출한 Hex Sequence 사이에 높은 유사도를 보이는 것을 확인할 수 있다. 테스트에 사용된 3개의 샘플 중, 1번과 2번 샘플의 경우 추출된 2Bytes Hex Sequence가 서로 100% 일치하는 것을 확인할 수 있으며 1번과 3번 Hex Sequence는 약 83% 일치하는 것을 확인할 수 있다.

표 5 Themida 2.0.3.0 Hex Sequence

Sample1(2030)	Sample2(2030)	Sample3(2030)
B8 00000000	B8 00000000	B8 00000000
60	60	60
0BC0	0BC0	0BC0
74 68	74 68	74 68
E8 00000000	E8 00000000	E8 00000000
58	58	58
05 53000000	05 53000000	05 53000000
8038 E9	8038 E9	8038 E9
75 13	75 13	75 13
61	61	61
EB 45	EB 45	EB 45
DB 2D 37 E0	DB 2D 37 00	DB 2D 37 50 43
FF	05	FF
FF	01	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
3D	FF	3D
40	FF	40
E8 00000000	3D	E8 00000000
58	40	58
25 00F0FFFF	E8 00000000	25 00F0FFFF

표 6 Themida 2.0.3.0 Hex Sequence(2bytes)

2bytes(Sample1)	2bytes(Sample2)	2bytes(Sample3)
B8	B8	B8
60	60	60
0B	0B	0B
74	74	74

E8	E8	E8
58	58	58
05	05	05
80	80	80
75	75	75
61	61	61
EB	EB	EB
DB	DB	DB
FF	05	FF
FF	01	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
3D	FF	3D
40	FF	40
E8	3D	E8
58	40	58
25	E8	25

Themida 2.0.3.0으로 Packing한 샘플을 대상으로 테스트한 결과를 UPX와 유사한 방식으로 테스트한 결과 2 Bytes를 추출한 Hex Sequence 사이에 역시 높은 유사도를 보이는 것을 확인할 수 있었다. 간단하게 백분율로 유사도를 계산하면 테스트에 사용된 3개의 샘플 중, 1번과 3번 샘플의 경우 추출된 2Bytes Hex Sequence가 서로 100% 일치하는 것을 확인할 수 있으며 1번과 3번 Hex Sequence는 약 93% 일치하는 것을 확인할 수 있다. 또한 동일한 Packer 중, 버전이 다른 경우(Themida2.0.3.0 & Themida 1.9.9.0)에도 서로 차이점을 찾아볼 수 있어 정확한 버전확인이 가능하다.

표7은 Packer 탐지 Tool의 성능 검증에 활용한 샘플을 통해 제안하는 알고리즘이 Packer의 이름 및 버전을 확인할 수 있는지 여부를 정리한 것이다. 표7에서 알 수 있듯이 대상이 되는 모든 Packer를 탐지할 수 있다.

표 7 Proposed Detection Signature Efficiency

Packer	Efficiency
UPX3.03	Enable(100%)
NsPack2.3	Enable(100%)
NsPack3.4	Enable(100%)
FSG1.33	Enable(100%)
FSG2.0	Enable(100%)
AsPack2.12	Enable(100%)
Themida2.0.3.0	Enable(100%)
Themida1.9.9.0	Enable(100%)
Armadillo 5.42	Enable(100%)
ASProtect 2.1	Enable(100%)
ASProtect 2.4	Enable(100%)

5. 결론 및 향후 연구 방향

본 논문에서는 기존 Packer 탐지 Tool의 성능을 검증하고 탐지율을 높이기 위한 시그니처 생성 방식을 제안했다. 그리고 11개의 Packer와 3개의 임의의 샘플을 대상으로 제안한 시그니처 생성 방식의 성능을 검증하였다. 검증 결과 UPX, FSG, NsPack, Themida를 비롯한 대부분의 Packer들을 정확하게 탐지할 수 있었으며 동일한 이름의 Packer라도 버전을 확인할 수 있는 요소들이 숨어있다는 것을 알 수 있었다. 물론 성능 검증 결과만으로 볼 때, 현재 가장 많이 사용된다고 알려져 있는 UPX를 대상으로 뛰어난 성능을 보이는 것을 확인할 수 있으나, 보다 범용적인 성능을 검증하기 위해 많은 수의 Packer와 다수의 샘플을 대상으로 테스트가 필요하다. 또한 Packer의 이름 및 버전을 정확하게 탐지하기 위한 알고리즘에 대한 Formal한 정의가 요구된다. 마지막으로 보다 원활한 검증을 위해 Hex Sequence를 추출하고 비교하는 일련의 과정을 자동화할 수 있는 Tool의 개발이 필요하다.

참고문헌

[1] ShadowServer, http:://www.shadowserver.org/
[2] PEiD, http://www.peid.info/
[3] Detect It Easy, http://hellspawn.nm.ru/
[4] Exeinfo PE, http://www.exeinfo.cjb.net/
[5] Protection ID, http://pid.gamecopyworld.com/
[6] UPX, http://upx.sourceforge.net/
[7] NsPack, http://download.cnet.com/Nspack/3000-2250_4-10403507.html
[8] FSG, http://www.woodmann.com
[9] AsPack, http://www.aspack.com/
[10] Themida, http://www.oreans.com/themida.php
[11] Armadillo, http://www.siliconrealms.com/
[12] ASProtect SKE, http://www.aspack.com/
[13] M. Zubair Shafiq, PE-Probe: Leveraging Packer Detection and Structural Information to Detection Malicious Portable Executables, VB2009, 29-33, 2009
[14] M. Zubair Shafiq, PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime, RAID2009, Volume 5758/2009, 121-141, 2009
[15] Gaith Taha, Counterattacking the packers, McAfee, 2007
[16] Executable compression, http://en.wikipedia.org/wiki/Executable_compression
[17] Lutz B□hne, Pandora's Bochs:Automatic Unpacking of Malware, 2008