Verschleiernde Transformationen von Programmen

Analyse und Systematisierung von Techniken zur Verschleierung von Programmmerkmalen in Malware

Michael Rex

Informationssysteme und Sicherheit
Lehrstuhl VI
Fakultät für Informatik
TU Dortmund

SPRING 2011
6. SIDAR Graduierten-Workshop über Reaktive Sicherheit

21. März 2011

Gliederung

Gliederung

Motivation

Malware

*mal*icious soft*ware*: jede Art von Software mit bösartigen Absichten oder Effekten

G Data Malware-Report 2. Halbjahr 2010:
1076236 neue Schädlinge aus 2608 Familien

- viele unterschiedlich aussehende Binaries
- weit weniger tatsächlich unterschiedliche Malware-Exemplare

Wodurch unterscheiden sich die Varianten?

Gliederung

Was sind Merkmale?

Merkmale

bestimmte (statische oder dynamische) Eigenschaften eines Programms

- Zeichenketten im Binary
- Strukturen des Programmcodes
- Laufzeitverhalten
- ...

Merkmalsausprägung

konkrete Instanz eines Merkmals

Kategorien

Merkmale eingeteilt in 4 Kategorien:

- Erscheinungsbild
- Kontrollflussgraph
- Verhalten
- Funktionalität

Gliederung

Was sind Transformationen?

Transformationen

Techniken, um durch Veränderungen von Merkmalsausprägungen unterschiedliche Samples eines Programms zu erzeugen

- **verhaltenserhaltend**:
  verändert lediglich das Erscheinungsbild (Bytefolgen im Binary) des Programms, ohne die tatsächlich ausgeführten Aktionen zu verändern
- **funktionalitätserhaltend**:
  verändert neben dem Erscheinungsbild auch das Verhalten (Systemaufrufe) des Programms, wobei das neue Verhalten dieselben Auswirkungen hat wie das ursprüngliche

Anwendungen von Transformationen

- Komprimierung
- Schutz vor Reverse Engineering
- Lizenzmanagement
- ...

aber auch:

- verstecken vor Anti-Virus-Software
  (verschleiern von Merkmalen)
- komplizierteres Reverse Engineering der Malware
- kompliziertere Signaturerstellung
- ...

Anwendungen von Transformationen

- Komprimierung
- Schutz vor Reverse Engineering
- Lizenzmanagement
- ...

aber auch:

- verstecken vor Anti-Virus-Software
  (verschleiern von Merkmalen)
- komplizierteres Reverse Engineering der Malware
- kompliziertere Signaturerstellung
- ...

Grundlegende Transformationstechniken

- Komprimierung
- Verschlüsselung
- metamorphe Codeumformung
- Virtualisierung

Komprimierung

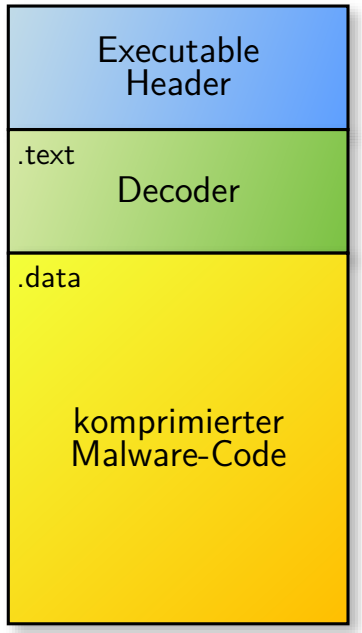
Funktionsprinzip:

- Originalcode wird komprimiert
- Decoder wird am Programmanfang eingefügt

Zur Laufzeit:

- Decoder entpackt komprimierten Code im Speicher
- übergibt Kontrolle an das wiederhergestellte Programm

ein Originalprogramm $\Rightarrow$
genau *eine* komprimierte Variante



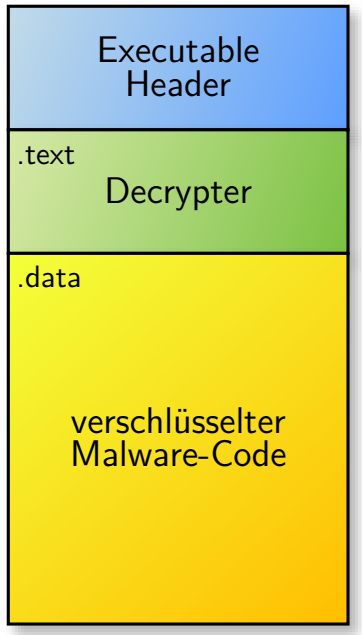
Verschlüsselung

arbeitet nach ähnlichem Prinzip:

- Originalcode wird verschlüsselt
  (durch Wechsel des Schlüssels ergeben sich unterschiedliche Samples)
- Decrypter entschlüsselt Code zur Laufzeit

Unterscheidung anhand der Anzahl der verwendeten Decrypter:

- *einfache Verschlüsselung*: ein einziger Decrypter (evtl. mit wechselndem Schlüssel)
- *oligomorphe Verschlüsselung*: ein paar wenige, unterschiedliche Decrypter
- *polymorphe Verschlüsselung*: praktisch unendlich viele Decrypter (metamorpher Decrypter)

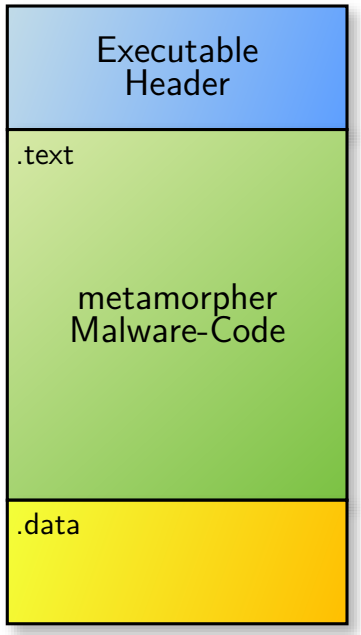


metamorphe Codeumformung

- kein Decrypter, Code wird nicht verschlüsselt
- Änderung direkt an den Opcodes des Programms

Techniken

- **Instruktionsersetzung**:
  Instruktionen durch semantisch gleiche ersetzen
- **Registerumbenennung**:
  die von einem Befehl verwendeten Register wechseln
- **Einfügung bedingungsloser Sprünge**:
  Codeblöcke umsortieren, mit Sprungbefehlen die richtige Reihenfolge bei der Ausführung herstellen
- **Einfügung überflüssigen Codes**:
  Code einfügen, der nie ausgeführt wird
- ...

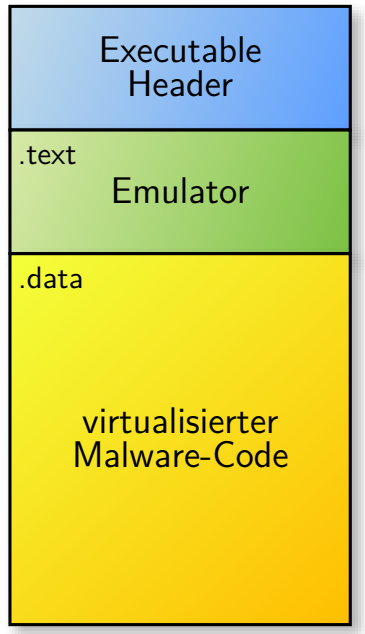


Virtualisierung

- Originalcode wird in Code für virtuelles Instruction Set übersetzt
- übersetzter Code wird von in Software implementierter, virtueller CPU ausgeführt

Vorteile der Virtualisierung

- Instruction Set je nach Bedarf zu entwerfen
- hohe Redundanz bei den Opcodes möglich
  $\Rightarrow$ viele Gelegenheiten für z. B. Instruction Substitution
- den Analysten unbekanntes Instruction Set
  $\Rightarrow$ schwierig Programm zu verstehen



Beeinflusste Merkmale

Merkmalskategorien

- Erscheinungsbild
- Kontrollflussgraph
- Verhalten
- Funktionalität

Transformation	E	C	V	F
Komprimierung	✓	–	–	–
Verschlüsselung	✓	–	–	–
metamorphe Codeumformung	✓	✓	✓	–
Virtualisierung	✓	✓	✓	–

Beeinflusste Merkmale bei der metamorphen Codeumformung

Technik	E	C	V	F
Instruktionsersetzung	✓	–	✓	–
Instruktionsfolgenersetzung	✓	–	✓	–
Instruktionsneuordnung	✓	–	✓	–
Ausdrucksneuordnung	✓	–	✓	–
Registerumbenennung	✓	–	✓	–
Datenumsortierung	✓	–	✓	–
Einfügung bedingungsloser Sprünge	✓	–	✓	–
Einfügung überflüssigen Codes	✓	–	✓	–
Codeerzeugung zur Laufzeit	✓	–	✓	–
Eingliedern und Ausgliedern	✓	✓	✓	–
Einfügung pseudo-bedingter Sprünge	✓	✓	✓	–
Pseudoverzweigung	✓	✓	✓	–

Gliederung

Hintergrund

Test von Anti-Virus-Software mit vielen unterschiedlichen Samples, um Aussagen über die Resistenz der Software gegenüber Transformationen zu treffen

Untersuchte Fragen

1. Inwiefern beeinflussen Transformationen die Erkennungsleistung von Anti-Virus-Software? Lassen sich syntaktische Merkmale wie Zeichenketten in einem Malware-Binary leicht verschleiern und die statische Analyse von Programmen dadurch aushebeln?

2. Wie wirken sich Transformationen auf das Verhalten von Programmen aus? Inwiefern sind Merkmale des Verhaltens resistent gegen Transformationen? Welche Transformationen können das Verhalten beeinflussen?

Untersuchte Fragen

1. Inwiefern beeinflussen Transformationen die Erkennungsleistung von Anti-Virus-Software? Lassen sich syntaktische Merkmale wie Zeichenketten in einem Malware-Binary leicht verschleiern und die statische Analyse von Programmen dadurch aushebeln?

2. Wie wirken sich Transformationen auf das Verhalten von Programmen aus? Inwiefern sind Merkmale des Verhaltens resistent gegen Transformationen? Welche Transformationen können das Verhalten beeinflussen?

Versuchsaufbau 1/2

Frage 1

Inwiefern beeinflussen Transformationen die Erkennungsleistung von Anti-Virus-Software? Lassen sich syntaktische Merkmale wie Zeichenketten in einem Malware-Binary leicht verschleiern und die statische Analyse von Programmen dadurch aushebeln?

- mehrere Varianten bekannter Malware erstellen:
  - 5 Packer erzeugen von
  - 5 Malware-Samples jeweils
  - 5 transformierte Varianten
- überprüfen ob Varianten noch von Anti-Virus-Software erkannt werden (VirusTotal)

Versuchsaufbau 2/2

Frage 2

Wie wirken sich Transformationen auf das Verhalten von Programmen aus? Inwiefern sind Merkmale des Verhaltens resistent gegen Transformationen? Welche Transformationen können das Verhalten beeinflussen?

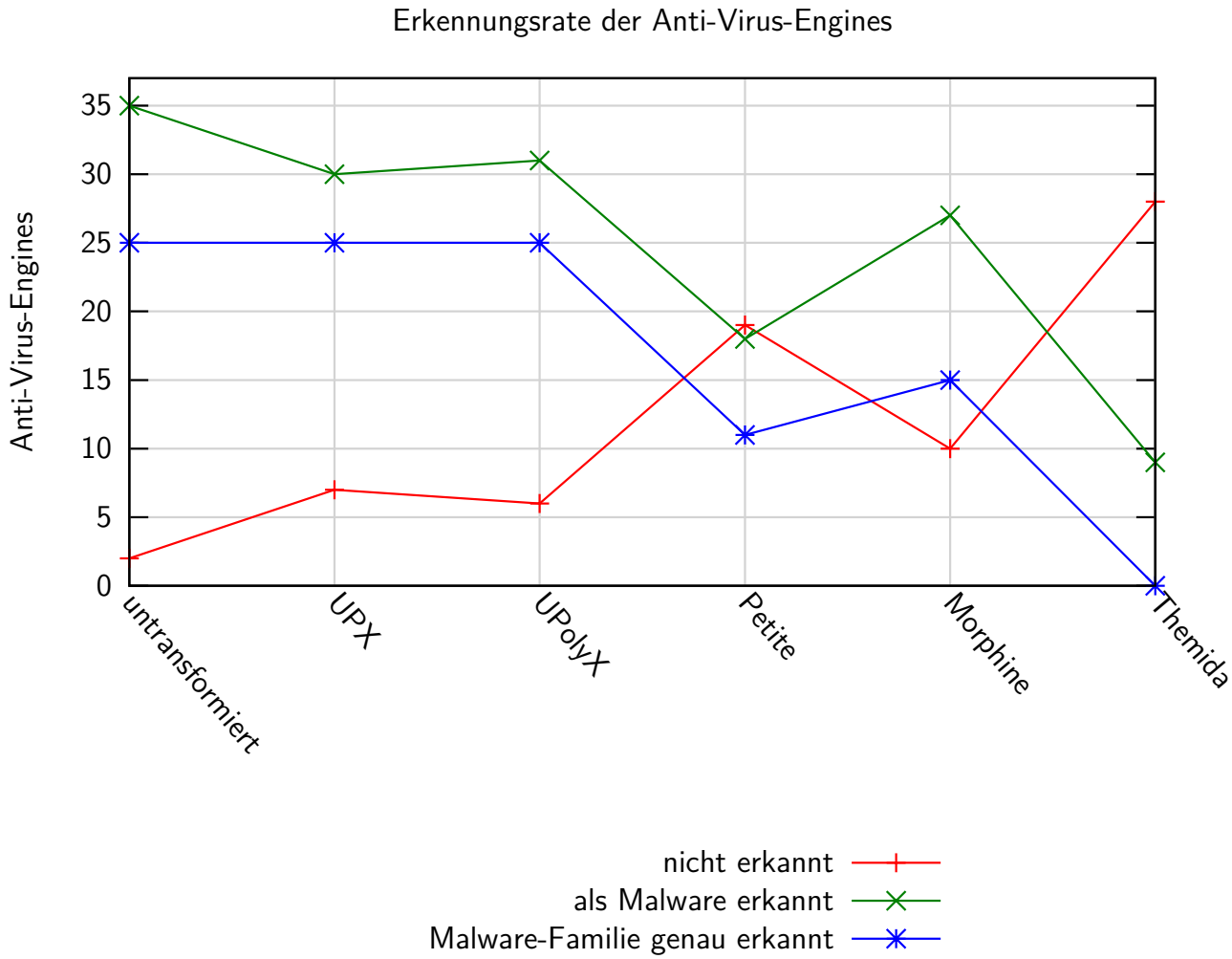
- eigene Test-„Malware“ mit komplett bekanntem Verhalten
  - 5 Packer erzeugen davon
  - 5 transformierte Varianten
- Verhaltensberichte auf Unterschiede zum Verhalten des Ursprungs-Programms analysieren

verwendete Packer

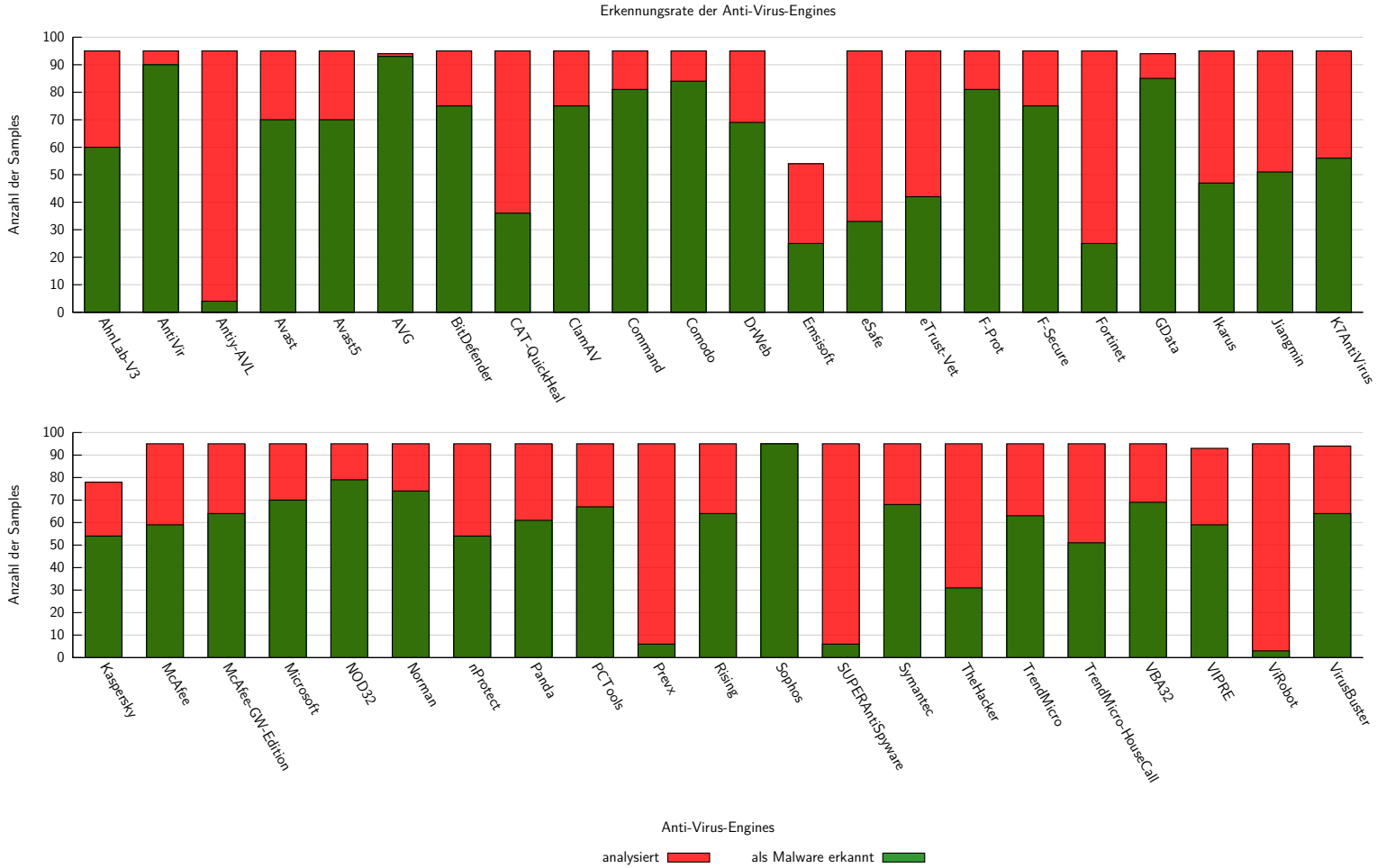
UPX	Komprimierung
UPolyX	polymorphe Verschlüsselung
Petite	Komprimierung
Morphine	polymorphe Verschlüsselung
Themida	Komprimierung, Verschlüsselung, metamorphe Codeumformung, Virtualisierung

Ergebnisse

Beispiel für *Net-Worm.Win32.Kido.ih*



Ergebnisse


Erkennungsrate der Anti-Virus-Engines
Anzahl der Samples
0
10
20
30
40
50
60
70
80
90
100
AhnLab-V3
AntiVir
Antiy-AVL
Avast
Avast5
AVG
BitDefender
CAT-QuickHeal
ClamAV
Command
Comodo
DrWeb
Emsisoft
eSafe
eTrust-Vet
F-Prot
F-Secure
Fortinet
GData
Ikarus
Jiangmin
K7AntiVirus
Kaspersky
McAfee
McAfee-GW-Edition
Microsoft
NOD32
Norman
nProtect
Panda
PCTools
Prevx
Rising
Sophos
SUPERAntiSpyware
Symantec
TheHacker
TrendMicro
TrendMicro-HouseCall
VBA32
VIPRE
ViRobot
VirusBuster
Anti-Virus-Engines
analysiert
als Malware erkannt

Ergebnisse

Verhalten

- untersucht auf Systemaufruf-Ebene
- keine nennenswerten Unterschiede feststellbar
  $\Rightarrow$ bei Komprimierung und Verschlüsselung zu erwarten
  $\Rightarrow$ bei Virtualisierung durch Einschränkungen von Themida zu erklären

Fragen?