

# Reverse Engineering 101

**Dozent:** Prof. Dr. Michael Eichberg

**Kontakt:** michael.eichberg@dhw-mannheim.de

**Version:** 1.0



1

---

**Folien:** **HTML:** <https://delors.github.io/sec-reversing101/folien.de.rst.html>

**PDF:** <https://delors.github.io/sec-reversing101/folien.de.rst.html.pdf>

**Fehler auf Folien melden:**

<https://github.com/Delors/delors.github.io/issues>

# Vorerfahrungen?

- Wer hat schon einmal Software or Hardware Reverse Engineering betrieben?
- Wer kennt Java Bytecode?
- Wer hat Erfahrung mit Python?

# Reverse Engineering

Reverse Engineering ist die Analyse von Systemen mit dem Ziel, ihren Aufbau und ihre Funktionsweise zu verstehen.

Typische Anwendungsfälle:

- die Rekonstruktion (von Teilen) des Quellcodes von Programmen, die nur als Binärabbild vorliegen.
- die Analyse von Kommunikationsprotokollen proprietärer Software

---

Vom Reverse Engineering ist das **Reengineering** zu unterscheiden. Im Fall von letzteren geht es „nur“ darum die Funktionalität eines bestehenden Systems mit neuen Techniken wiederherzustellen.

# Zweck von Reverse Engineering

- Herstellung von Interoperabilität
- Untersuchung auf Schwachstellen
- Untersuchung auf Copyrightverletzungen
- Untersuchung auf Backdoors
- Analyse von Viren, Würmern etc.
- Umgehung von ungerechtfertigten(?) Schutzmaßnahmen (z.B. bei Malware)

# Reverse Engineering - grundlegende Schritte

1. Informationsgewinnung zur Gewinnung aller relevanten Informationen über das Produkt
2. Modellierung mit dem Ziel der (Wieder-)Gewinnung eines (abstrakten) Modells der relevanten Funktionalität.
3. Überprüfung (审查 *review*) des Modells auf seine Richtigkeit und Vollständigkeit.

# Informationsgewinnung - Beispiel

Gegeben sei eine App zum Ver- und Entschlüsseln von Dateien sowie ein paar verschlüsselte Dateien. Mögliche erste Schritte vor der Analyse von Binärkode:

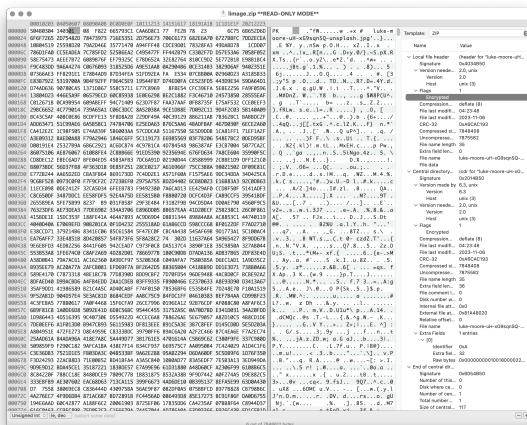
- Die ausführbare Datei ggf. mit **file** überprüfen (z.B. wie kompiliert und für welches Betriebssystem und Architektur)

Beispiel:

```
$ file /usr/bin/openssl
/usr/bin/openssl: Mach-O universal binary with 2 archi...
/usr/bin/openssl (for architecture x86_64): Mach-O 64-bit
/usr/bin/openssl (for architecture arm64e): Mach-O 64-bit
```

1

- Die Dateien mit einem (guten) Hexeditor auf Auffälligkeiten untersuchen.



2

Die Datei auf bekannte Viren und Malware überprüfen.

3

- Eine Datei mit einem bekannten Inhalt verschlüsseln und danach vergleichen.

Ist die Datei gleich groß?

Falls ja, dann werden keine Metainformationen gespeichert und das Passwort kann (ggf.) nicht (leicht) verifiziert werden.

- Eine Datei mit verschiedenen Passworten verschlüsseln.

Sind die Dateien gleich?

Falls ja, dann wäre die Verschlüsselung komplett nutzlos und es gilt nur noch den konstanten Schlüssel zu finden.

Gibt es Gemeinsamkeiten?

Falls ja, dann wäre es möglich, dass das Passwort (gehasht) in der Datei gespeichert wird.

5

- Eine Datei mit einem wohldefinierten Muster verschlüsseln, um ggf. den "Mode of Operation" (insbesondere ECB) zu identifizieren.

6

- Mehrere verschiedene Dateien mit dem gleichen Passwort verschlüsseln

Gibt es Gemeinsamkeiten?

Falls ja, dann wäre es möglich, dass die entsprechenden Teile direkt vom Passwort abgeleitet werden/damit verschlüsselt werden.

7

- ...

8

6

# Rechtliche Aspekte des Reverse Engineering

- **unterschiedliche nationale Gesetzgebung**
- Rechtslage in Deutschland hat sich mehrfach geändert
- Umgehung von Kopierschutzmechanismen ist im Allgemeinen verboten
- Lizenz verbietet das Reverse Engineering häufig

## Warnung

Bevor Sie Reverse Engineering von Systemen betreiben, erkundigen sie sich erst über mögliche rechtliche Konsequenzen.

# 1. SOFTWARE REVERSE ENGINEERING

Prof. Dr. Michael Eichberg

# Ansätze

**statische Analyse:** Studieren des Programms ohne es auszuführen; typischerweise mittels eines Disassemblers oder eines Decompilers.

**dynamische Analyse:**

Ausführen des Programms; typischerweise unter Verwendung eines Debuggers oder eines instrumentations Frameworks (z.B. [Frida](#)).

**hybride Analyse:** Kombination aus statischer und dynamischer Analyse.

Ansätze wie [Unicorn](#), welches auf [QEmu](#) aufbaut, erlaubt zum Beispiel die Ausführung von (Teilen von) Binärcode auf einer anderen Architektur als der des Hosts.

Ein Beispiel wäre die Ausführung einer Methode, die im Code verschlüsselte hinterlegte Strings entschlüsselt ( *deobfuscation*), um die Analyse zu vereinfachen.

# Disassembler

Überführt (maschinenlesbaren) Binärkode in Assemblercode

Beispiel:

- objdump -d
- gdb
- radare
- javap (für Java)

## Hinweis

Für einfache Programme ist es häufig möglich direkt den gesamten Assemblercode mittels der entsprechenden Werkzeuge zu erhalten. Im Falle komplexer Binärdateien (z.B. im ELF (Linux) und PE (Windows) Format) gilt dies nicht und erfordert ggf. manuelle Unterstützung zum Beispiel durch das Markieren von Methodenanfängen.

Im Fall von Java `.class` ist die Disassembly immer möglich.

# Decompiler

Überführt (maschinenlesbarem) Binärkode bestmöglich in Hochsprache (meist C oder Java). Eine *kleine* Auswahl von verfügbaren Werkzeugen:

- Hex-Rays IDAPro (kommerziell)
- **Ghidra** (unterstützt fast jede Platform; die Ergebnisse sind sehr unterschiedlich)
- JadX (Androids **.dex** Format)
- CFR (Java **.class** Dateien)
- IntelliJ
- [decompiler.com](http://decompiler.com)

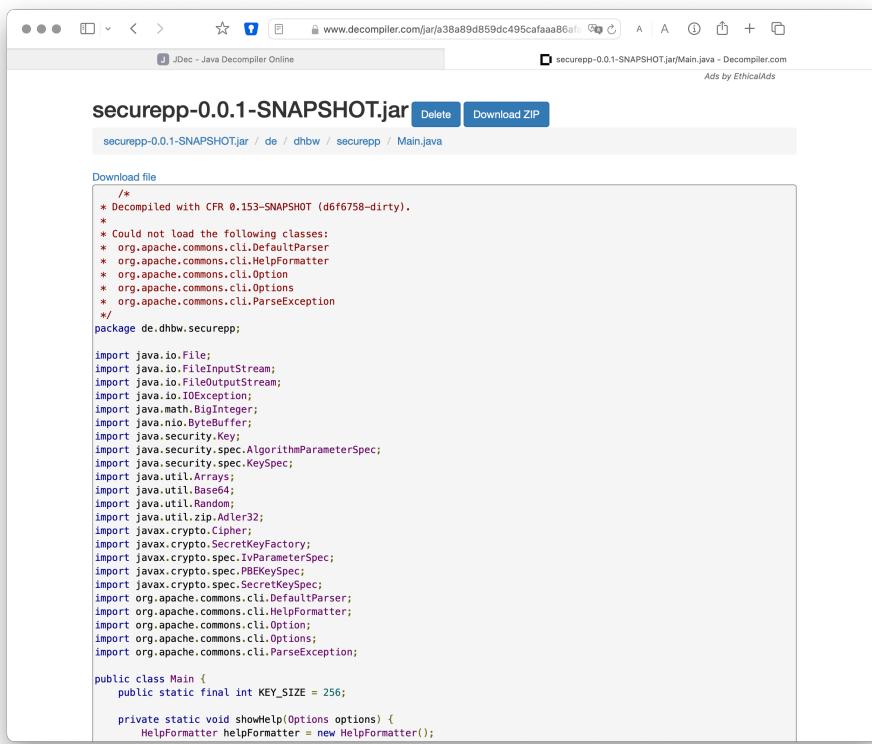
## Hinweis

Generell sehr hilfreich, aber gleichzeitig auch sehr fehlerbehaftet. Vieles, dass im Binärkode möglich ist, hat auf Sourcecode Ebene keine Entsprechung. Zum Beispiel unterstützt Java Bytecode beliebige Sprünge. Solche, die

Mittels Decompiler ist es ggf. möglich Code, der zum Beispiel ursprünglich in Kotlin oder Scala geschrieben und für die JVM kompiliert wurde, als Java Code zurückzubekommen.

Die Ergebnisse sind für Analysezwecke zwar häufig ausreichend gut - von funktionierendem Code jedoch ggf. ((sehr) weit) entfernt.

# cfr Decompiler



The screenshot shows a Java decompiler interface with the following details:

- Title Bar:** JDDec - Java Decompiler Online | www.decompiler.com/jar/a38a89d859dc495cafaaa86af | A A ⓘ +
- Page Title:** securepp-0.0.1-SNAPSHOT.jar>Main.java - Decompiler.com
- Page Subtitle:** Ads by EthicalAdds
- File Information:** securepp-0.0.1-SNAPSHOT.jar / de / dhbw / securepp / Main.java
- Download Options:** Download file | Delete | Download ZIP
- Code Preview:** The code is displayed in a scrollable text area. It includes imports from java.io, java.util, and javax.crypto, as well as org.apache.commons.cli. The Main class contains a static final int KEY\_SIZE = 256; and a private static void showHelp(Options options) method.

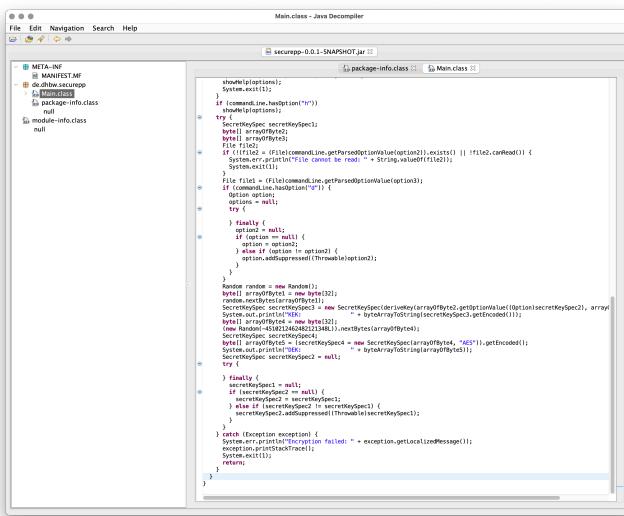
```
/*
 * Decompiled with CFR 0.153-SNAPSHOT (d6f6758-dirty).
 *
 * Could not load the following classes:
 * org.apache.commons.cli.DefaultParser
 * org.apache.commons.cli.HelpFormatter
 * org.apache.commons.cli.Option
 * org.apache.commons.cli.Options
 * org.apache.commons.cli.ParseException
 */
package de.dhbw.securepp;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.security.Key;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.KeySpec;
import java.util.Arrays;
import java.util.Base64;
import java.util.Random;
import java.util.zip.Adler32;
import javax.crypto.Cipher;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.cli.DefaultParser;
import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Option;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class Main {
    public static final int KEY_SIZE = 256;

    private static void showHelp(Options options) {
        HelpFormatter helpFormatter = new HelpFormatter();
    }
}
```

# JD Decompiler



```
SecretKeySpec secretKeySpec2 = null;
try {
} finally {
    secretKeySpec1 = null;
    if (secretKeySpec2 == null) {
        secretKeySpec2 = secretKeySpec1;
    } else if (secretKeySpec2 != secretKeySpec1) {
        secretKeySpec2.addSuppressed((Throwable)secretKeySpec1);
    }
}
```

Beispiel fehlgeschlagener Dekompilierung

# JDec Decompiler

The screenshot shows the JDec Decompiler interface. The left sidebar displays the file structure of the jar file:

- securepp-0.0.1-SNAPSHOT.jar
- META-INF
- de
- dhbw
- securepp
- Main.class
- package-info.class
- module-info.class

The right pane shows the decompiled code for Main.java. The code handles command-line arguments for decryption, password, input file, output file, and help. It uses OptionParser to parse arguments and File objects to handle files.

```
54     }
55     var1.append(")");
56     return var1.toString();
57 }
58
59 public static void main(String[] var0) {
60     Options var1;
61     var1 = new Options().addOption(Option.builder("d").longOpt("decrypt").desc("Performs dec
62     Option var2 = Option.builder("p").longOpt("password").desc("The password.").hasArg().type(
63     var1.addOption(var2);
64     Option var3 = Option.builder("in").desc("The file to process.").hasArg().type(File.class).
65     var1.addOption(var3);
66     Option var4 = Option.builder("out").desc("The output.").hasArg().type(File.class).required
67     var1.addOption(var4);
68     var1.addOption(Option.builder("h").longOpt("help").desc("Get Help.").build());
69     CommandLine var5 = null;
70
71     try {
72         var5 = (new DefaultParser()).parse(var1, var0);
73     } catch (ParseException var70) {
74         System.out.println(var70.getMessage());
75         showHelp(var1);
76         System.exit(1);
77     }
78 }
79
80 if (var5.hasOption("h")) {
81     showHelp(var1);
82 }
83
84 try {
85     File var6;
86     if (!(var6 = (File)var5.getParsedOptionValue(var3)).exists() || !var6.canRead()) {
87         System.err.println("File cannot be read: " + String.valueOf(var6));
88         System.exit(1);
89     }
90
91     File var77 = (File)var5.getParsedOptionValue(var4);
92     Cipher var79;
93     byte[] var84;
94     IvParameterSpec var85;
95     if (var5.hasOption("d")) {
96         Throwable var81 = null;
97
98         try {
99             FileInputStream var88 = new FileInputStream(var6);
100
101         } catch (FileNotFoundException var101) {
102             System.out.println("File not found: " + var6);
103             System.exit(1);
104         }
105
106         var84 = var79.getIV();
107         var85 = var79.getIvParameterSpec();
108         var79.init(var84, var85);
109
110         try {
111             var88.close();
112         } catch (IOException var112) {
113             System.out.println("Error closing file: " + var6);
114             System.exit(1);
115         }
116
117         var84 = var79.getIV();
118         var85 = var79.getIvParameterSpec();
119         var79.init(var84, var85);
120
121         try {
122             var88.close();
123         } catch (IOException var123) {
124             System.out.println("Error closing file: " + var6);
125             System.exit(1);
126         }
127
128         var84 = var79.getIV();
129         var85 = var79.getIvParameterSpec();
130         var79.init(var84, var85);
131
132         try {
133             var88.close();
134         } catch (IOException var134) {
135             System.out.println("Error closing file: " + var6);
136             System.exit(1);
137         }
138
139         var84 = var79.getIV();
140         var85 = var79.getIvParameterSpec();
141         var79.init(var84, var85);
142
143         try {
144             var88.close();
145         } catch (IOException var145) {
146             System.out.println("Error closing file: " + var6);
147             System.exit(1);
148         }
149
150         var84 = var79.getIV();
151         var85 = var79.getIvParameterSpec();
152         var79.init(var84, var85);
153
154         try {
155             var88.close();
156         } catch (IOException var156) {
157             System.out.println("Error closing file: " + var6);
158             System.exit(1);
159         }
160
161         var84 = var79.getIV();
162         var85 = var79.getIvParameterSpec();
163         var79.init(var84, var85);
164
165         try {
166             var88.close();
167         } catch (IOException var167) {
168             System.out.println("Error closing file: " + var6);
169             System.exit(1);
170         }
171
172         var84 = var79.getIV();
173         var85 = var79.getIvParameterSpec();
174         var79.init(var84, var85);
175
176         try {
177             var88.close();
178         } catch (IOException var178) {
179             System.out.println("Error closing file: " + var6);
180             System.exit(1);
181         }
182
183         var84 = var79.getIV();
184         var85 = var79.getIvParameterSpec();
185         var79.init(var84, var85);
186
187         try {
188             var88.close();
189         } catch (IOException var189) {
190             System.out.println("Error closing file: " + var6);
191             System.exit(1);
192         }
193
194         var84 = var79.getIV();
195         var85 = var79.getIvParameterSpec();
196         var79.init(var84, var85);
197
198         try {
199             var88.close();
200         } catch (IOException var190) {
201             System.out.println("Error closing file: " + var6);
202             System.exit(1);
203         }
204
205         var84 = var79.getIV();
206         var85 = var79.getIvParameterSpec();
207         var79.init(var84, var85);
208
209         try {
210             var88.close();
211         } catch (IOException var191) {
212             System.out.println("Error closing file: " + var6);
213             System.exit(1);
214         }
215
216         var84 = var79.getIV();
217         var85 = var79.getIvParameterSpec();
218         var79.init(var84, var85);
219
220         try {
221             var88.close();
222         } catch (IOException var192) {
223             System.out.println("Error closing file: " + var6);
224             System.exit(1);
225         }
226
227         var84 = var79.getIV();
228         var85 = var79.getIvParameterSpec();
229         var79.init(var84, var85);
230
231         try {
232             var88.close();
233         } catch (IOException var193) {
234             System.out.println("Error closing file: " + var6);
235             System.exit(1);
236         }
237
238         var84 = var79.getIV();
239         var85 = var79.getIvParameterSpec();
240         var79.init(var84, var85);
241
242         try {
243             var88.close();
244         } catch (IOException var194) {
245             System.out.println("Error closing file: " + var6);
246             System.exit(1);
247         }
248
249         var84 = var79.getIV();
250         var85 = var79.getIvParameterSpec();
251         var79.init(var84, var85);
252
253         try {
254             var88.close();
255         } catch (IOException var195) {
256             System.out.println("Error closing file: " + var6);
257             System.exit(1);
258         }
259
260         var84 = var79.getIV();
261         var85 = var79.getIvParameterSpec();
262         var79.init(var84, var85);
263
264         try {
265             var88.close();
266         } catch (IOException var196) {
267             System.out.println("Error closing file: " + var6);
268             System.exit(1);
269         }
270
271         var84 = var79.getIV();
272         var85 = var79.getIvParameterSpec();
273         var79.init(var84, var85);
274
275         try {
276             var88.close();
277         } catch (IOException var197) {
278             System.out.println("Error closing file: " + var6);
279             System.exit(1);
280         }
281
282         var84 = var79.getIV();
283         var85 = var79.getIvParameterSpec();
284         var79.init(var84, var85);
285
286         try {
287             var88.close();
288         } catch (IOException var198) {
289             System.out.println("Error closing file: " + var6);
290             System.exit(1);
291         }
292
293         var84 = var79.getIV();
294         var85 = var79.getIvParameterSpec();
295         var79.init(var84, var85);
296
297         try {
298             var88.close();
299         } catch (IOException var199) {
300             System.out.println("Error closing file: " + var6);
301             System.exit(1);
302         }
303
304         var84 = var79.getIV();
305         var85 = var79.getIvParameterSpec();
306         var79.init(var84, var85);
307
308         try {
309             var88.close();
310         } catch (IOException var200) {
311             System.out.println("Error closing file: " + var6);
312             System.exit(1);
313         }
314
315         var84 = var79.getIV();
316         var85 = var79.getIvParameterSpec();
317         var79.init(var84, var85);
318
319         try {
320             var88.close();
321         } catch (IOException var201) {
322             System.out.println("Error closing file: " + var6);
323             System.exit(1);
324         }
325
326         var84 = var79.getIV();
327         var85 = var79.getIvParameterSpec();
328         var79.init(var84, var85);
329
330         try {
331             var88.close();
332         } catch (IOException var202) {
333             System.out.println("Error closing file: " + var6);
334             System.exit(1);
335         }
336
337         var84 = var79.getIV();
338         var85 = var79.getIvParameterSpec();
339         var79.init(var84, var85);
340
341         try {
342             var88.close();
343         } catch (IOException var203) {
344             System.out.println("Error closing file: " + var6);
345             System.exit(1);
346         }
347
348         var84 = var79.getIV();
349         var85 = var79.getIvParameterSpec();
350         var79.init(var84, var85);
351
352         try {
353             var88.close();
354         } catch (IOException var204) {
355             System.out.println("Error closing file: " + var6);
356             System.exit(1);
357         }
358
359         var84 = var79.getIV();
360         var85 = var79.getIvParameterSpec();
361         var79.init(var84, var85);
362
363         try {
364             var88.close();
365         } catch (IOException var205) {
366             System.out.println("Error closing file: " + var6);
367             System.exit(1);
368         }
369
370         var84 = var79.getIV();
371         var85 = var79.getIvParameterSpec();
372         var79.init(var84, var85);
373
374         try {
375             var88.close();
376         } catch (IOException var206) {
377             System.out.println("Error closing file: " + var6);
378             System.exit(1);
379         }
380
381         var84 = var79.getIV();
382         var85 = var79.getIvParameterSpec();
383         var79.init(var84, var85);
384
385         try {
386             var88.close();
387         } catch (IOException var207) {
388             System.out.println("Error closing file: " + var6);
389             System.exit(1);
390         }
391
392         var84 = var79.getIV();
393         var85 = var79.getIvParameterSpec();
394         var79.init(var84, var85);
395
396         try {
397             var88.close();
398         } catch (IOException var208) {
399             System.out.println("Error closing file: " + var6);
400             System.exit(1);
401         }
402
403         var84 = var79.getIV();
404         var85 = var79.getIvParameterSpec();
405         var79.init(var84, var85);
406
407         try {
408             var88.close();
409         } catch (IOException var209) {
410             System.out.println("Error closing file: " + var6);
411             System.exit(1);
412         }
413
414         var84 = var79.getIV();
415         var85 = var79.getIvParameterSpec();
416         var79.init(var84, var85);
417
418         try {
419             var88.close();
420         } catch (IOException var210) {
421             System.out.println("Error closing file: " + var6);
422             System.exit(1);
423         }
424
425         var84 = var79.getIV();
426         var85 = var79.getIvParameterSpec();
427         var79.init(var84, var85);
428
429         try {
430             var88.close();
431         } catch (IOException var211) {
432             System.out.println("Error closing file: " + var6);
433             System.exit(1);
434         }
435
436         var84 = var79.getIV();
437         var85 = var79.getIvParameterSpec();
438         var79.init(var84, var85);
439
440         try {
441             var88.close();
442         } catch (IOException var212) {
443             System.out.println("Error closing file: " + var6);
444             System.exit(1);
445         }
446
447         var84 = var79.getIV();
448         var85 = var79.getIvParameterSpec();
449         var79.init(var84, var85);
450
451         try {
452             var88.close();
453         } catch (IOException var213) {
454             System.out.println("Error closing file: " + var6);
455             System.exit(1);
456         }
457
458         var84 = var79.getIV();
459         var85 = var79.getIvParameterSpec();
460         var79.init(var84, var85);
461
462         try {
463             var88.close();
464         } catch (IOException var214) {
465             System.out.println("Error closing file: " + var6);
466             System.exit(1);
467         }
468
469         var84 = var79.getIV();
470         var85 = var79.getIvParameterSpec();
471         var79.init(var84, var85);
472
473         try {
474             var88.close();
475         } catch (IOException var215) {
476             System.out.println("Error closing file: " + var6);
477             System.exit(1);
478         }
479
480         var84 = var79.getIV();
481         var85 = var79.getIvParameterSpec();
482         var79.init(var84, var85);
483
484         try {
485             var88.close();
486         } catch (IOException var216) {
487             System.out.println("Error closing file: " + var6);
488             System.exit(1);
489         }
490
491         var84 = var79.getIV();
492         var85 = var79.getIvParameterSpec();
493         var79.init(var84, var85);
494
495         try {
496             var88.close();
497         } catch (IOException var217) {
498             System.out.println("Error closing file: " + var6);
499             System.exit(1);
500         }
501
502         var84 = var79.getIV();
503         var85 = var79.getIvParameterSpec();
504         var79.init(var84, var85);
505
506         try {
507             var88.close();
508         } catch (IOException var218) {
509             System.out.println("Error closing file: " + var6);
510             System.exit(1);
511         }
512
513         var84 = var79.getIV();
514         var85 = var79.getIvParameterSpec();
515         var79.init(var84, var85);
516
517         try {
518             var88.close();
519         } catch (IOException var219) {
520             System.out.println("Error closing file: " + var6);
521             System.exit(1);
522         }
523
524         var84 = var79.getIV();
525         var85 = var79.getIvParameterSpec();
526         var79.init(var84, var85);
527
528         try {
529             var88.close();
530         } catch (IOException var220) {
531             System.out.println("Error closing file: " + var6);
532             System.exit(1);
533         }
534
535         var84 = var79.getIV();
536         var85 = var79.getIvParameterSpec();
537         var79.init(var84, var85);
538
539         try {
540             var88.close();
541         } catch (IOException var221) {
542             System.out.println("Error closing file: " + var6);
543             System.exit(1);
544         }
545
546         var84 = var79.getIV();
547         var85 = var79.getIvParameterSpec();
548         var79.init(var84, var85);
549
550         try {
551             var88.close();
552         } catch (IOException var222) {
553             System.out.println("Error closing file: " + var6);
554             System.exit(1);
555         }
556
557         var84 = var79.getIV();
558         var85 = var79.getIvParameterSpec();
559         var79.init(var84, var85);
560
561         try {
562             var88.close();
563         } catch (IOException var223) {
564             System.out.println("Error closing file: " + var6);
565             System.exit(1);
566         }
567
568         var84 = var79.getIV();
569         var85 = var79.getIvParameterSpec();
570         var79.init(var84, var85);
571
572         try {
573             var88.close();
574         } catch (IOException var224) {
575             System.out.println("Error closing file: " + var6);
576             System.exit(1);
577         }
578
579         var84 = var79.getIV();
580         var85 = var79.getIvParameterSpec();
581         var79.init(var84, var85);
582
583         try {
584             var88.close();
585         } catch (IOException var225) {
586             System.out.println("Error closing file: " + var6);
587             System.exit(1);
588         }
589
590         var84 = var79.getIV();
591         var85 = var79.getIvParameterSpec();
592         var79.init(var84, var85);
593
594         try {
595             var88.close();
596         } catch (IOException var226) {
597             System.out.println("Error closing file: " + var6);
598             System.exit(1);
599         }
600
601         var84 = var79.getIV();
602         var85 = var79.getIvParameterSpec();
603         var79.init(var84, var85);
604
605         try {
606             var88.close();
607         } catch (IOException var227) {
608             System.out.println("Error closing file: " + var6);
609             System.exit(1);
610         }
611
612         var84 = var79.getIV();
613         var85 = var79.getIvParameterSpec();
614         var79.init(var84, var85);
615
616         try {
617             var88.close();
618         } catch (IOException var228) {
619             System.out.println("Error closing file: " + var6);
620             System.exit(1);
621         }
622
623         var84 = var79.getIV();
624         var85 = var79.getIvParameterSpec();
625         var79.init(var84, var85);
626
627         try {
628             var88.close();
629         } catch (IOException var229) {
630             System.out.println("Error closing file: " + var6);
631             System.exit(1);
632         }
633
634         var84 = var79.getIV();
635         var85 = var79.getIvParameterSpec();
636         var79.init(var84, var85);
637
638         try {
639             var88.close();
640         } catch (IOException var230) {
641             System.out.println("Error closing file: " + var6);
642             System.exit(1);
643         }
644
645         var84 = var79.getIV();
646         var85 = var79.getIvParameterSpec();
647         var79.init(var84, var85);
648
649         try {
650             var88.close();
651         } catch (IOException var231) {
652             System.out.println("Error closing file: " + var6);
653             System.exit(1);
654         }
655
656         var84 = var79.getIV();
657         var85 = var79.getIvParameterSpec();
658         var79.init(var84, var85);
659
660         try {
661             var88.close();
662         } catch (IOException var232) {
663             System.out.println("Error closing file: " + var6);
664             System.exit(1);
665         }
666
667         var84 = var79.getIV();
668         var85 = var79.getIvParameterSpec();
669         var79.init(var84, var85);
670
671         try {
672             var88.close();
673         } catch (IOException var233) {
674             System.out.println("Error closing file: " + var6);
675             System.exit(1);
676         }
677
678         var84 = var79.getIV();
679         var85 = var79.getIvParameterSpec();
680         var79.init(var84, var85);
681
682         try {
683             var88.close();
684         } catch (IOException var234) {
685             System.out.println("Error closing file: " + var6);
686             System.exit(1);
687         }
688
689         var84 = var79.getIV();
690         var85 = var79.getIvParameterSpec();
691         var79.init(var84, var85);
692
693         try {
694             var88.close();
695         } catch (IOException var235) {
696             System.out.println("Error closing file: " + var6);
697             System.exit(1);
698         }
699
700         var84 = var79.getIV();
701         var85 = var79.getIvParameterSpec();
702         var79.init(var84, var85);
703
704         try {
705             var88.close();
706         } catch (IOException var236) {
707             System.out.println("Error closing file: " + var6);
708             System.exit(1);
709         }
710
711         var84 = var79.getIV();
712         var85 = var79.getIvParameterSpec();
713         var79.init(var84, var85);
714
715         try {
716             var88.close();
717         } catch (IOException var237) {
718             System.out.println("Error closing file: " + var6);
719             System.exit(1);
720         }
721
722         var84 = var79.getIV();
723         var85 = var79.getIvParameterSpec();
724         var79.init(var84, var85);
725
726         try {
727             var88.close();
728         } catch (IOException var238) {
729             System.out.println("Error closing file: " + var6);
730             System.exit(1);
731         }
732
733         var84 = var79.getIV();
734         var85 = var79.getIvParameterSpec();
735         var79.init(var84, var85);
736
737         try {
738             var88.close();
739         } catch (IOException var239) {
740             System.out.println("Error closing file: " + var6);
741             System.exit(1);
742         }
743
744         var84 = var79.getIV();
745         var85 = var79.getIvParameterSpec();
746         var79.init(var84, var85);
747
748         try {
749             var88.close();
750         } catch (IOException var240) {
751             System.out.println("Error closing file: " + var6);
752             System.exit(1);
753         }
754
755         var84 = var79.getIV();
756         var85 = var79.getIvParameterSpec();
757         var79.init(var84, var85);
758
759         try {
760             var88.close();
761         } catch (IOException var241) {
762             System.out.println("Error closing file: " + var6);
763             System.exit(1);
764         }
765
766         var84 = var79.getIV();
767         var85 = var79.getIvParameterSpec();
768         var79.init(var84, var85);
769
770         try {
771             var88.close();
772         } catch (IOException var242) {
773             System.out.println("Error closing file: " + var6);
774             System.exit(1);
775         }
776
777         var84 = var79.getIV();
778         var85 = var79.getIvParameterSpec();
779         var79.init(var84, var85);
780
781         try {
782             var88.close();
783         } catch (IOException var243) {
784             System.out.println("Error closing file: " + var6);
785             System.exit(1);
786         }
787
788         var84 = var79.getIV();
789         var85 = var79.getIvParameterSpec();
790         var79.init(var84, var85);
791
792         try {
793             var88.close();
794         } catch (IOException var244) {
795             System.out.println("Error closing file: " + var6);
796             System.exit(1);
797         }
798
799         var84 = var79.getIV();
800         var85 = var79.getIvParameterSpec();
801         var79.init(var84, var85);
802
803         try {
804             var88.close();
805         } catch (IOException var245) {
806             System.out.println("Error closing file: " + var6);
807             System.exit(1);
808         }
809
810         var84 = var79.getIV();
811         var85 = var79.getIvParameterSpec();
812         var79.init(var84, var85);
813
814         try {
815             var88.close();
816         } catch (IOException var246) {
817             System.out.println("Error closing file: " + var6);
818             System.exit(1);
819         }
820
821         var84 = var79.getIV();
822         var85 = var79.getIvParameterSpec();
823         var79.init(var84, var85);
824
825         try {
826             var88.close();
827         } catch (IOException var247) {
828             System.out.println("Error closing file: " + var6);
829             System.exit(1);
830         }
831
832         var84 = var79.getIV();
833         var85 = var79.getIvParameterSpec();
834         var79.init(var84, var85);
835
836         try {
837             var88.close();
838         } catch (IOException var248) {
839             System.out.println("Error closing file: " + var6);
840             System.exit(1);
841         }
842
843         var84 = var79.getIV();
844         var85 = var79.getIvParameterSpec();
845         var79.init(var84, var85);
846
847         try {
848             var88.close();
849         } catch (IOException var249) {
850             System.out.println("Error closing file: " + var6);
851             System.exit(1);
852         }
853
854         var84 = var79.getIV();
855         var85 = var79.getIvParameterSpec();
856         var79.init(var84, var85);
857
858         try {
859             var88.close();
860         } catch (IOException var250) {
861             System.out.println("Error closing file: " + var6);
862             System.exit(1);
863         }
864
865         var84 = var79.getIV();
866         var85 = var79.getIvParameterSpec();
867         var79.init(var84, var85);
868
869         try {
870             var88.close();
871         } catch (IOException var251) {
872             System.out.println("Error closing file: " + var6);
873             System.exit(1);
874         }
875
876         var84 = var79.getIV();
877         var85 = var79.getIvParameterSpec();
878         var79.init(var84, var85);
879
880         try {
881             var88.close();
882         } catch (IOException var252) {
883             System.out.println("Error closing file: " + var6);
884             System.exit(1);
885         }
886
887         var84 = var79.getIV();
888         var85 = var79.getIvParameterSpec();
889         var79.init(var84, var85);
890
891         try {
892             var88.close();
893         } catch (IOException var253) {
894             System.out.println("Error closing file: " + var6);
895             System.exit(1);
896         }
897
898         var84 = var79.getIV();
899         var85 = var79.getIvParameterSpec();
900         var79.init(var84, var85);
901
902         try {
903             var88.close();
904         } catch (IOException var254) {
905             System.out.println("Error closing file: " + var6);
906             System.exit(1);
907         }
908
909         var84 = var79.getIV();
910         var85 = var79.getIvParameterSpec();
911         var79.init(var84, var85);
912
913         try {
914             var88.close();
915         } catch (IOException var255) {
916             System.out.println("Error closing file: " + var6);
917             System.exit(1);
918         }
919
920         var84 = var79.getIV();
921         var85 = var79.getIvParameterSpec();
922         var79.init(var84, var85);
923
924         try {
925             var88.close();
926         } catch (IOException var256) {
927             System.out.println("Error closing file: " + var6);
928             System.exit(1);
929         }
930
931         var84 = var79.getIV();
932         var85 = var79.getIvParameterSpec();
933         var79.init(var84, var85);
934
935         try {
936             var88.close();
937         } catch (IOException var257) {
938             System.out.println("Error closing file: " + var6);
939             System.exit(1);
940         }
941
942         var84 = var79.getIV();
943         var85 = var79.getIvParameterSpec();
944         var79.init(var84, var85);
945
946         try {
947             var88.close();
948         } catch (IOException var258) {
949             System.out.println("Error closing file: " + var6);
950             System.exit(1);
951         }
952
953         var84 = var79.getIV();
954         var85 = var79.getIvParameterSpec();
955         var79.init(var84, var85);
956
957         try {
958             var88.close();
959         } catch (IOException var259) {
960             System.out.println("Error closing file: " + var6);
961             System.exit(1);
962         }
963
964         var84 = var79.getIV();
965         var85 = var79.getIvParameterSpec();
966         var79.init(var84, var85);
967
968         try {
969             var88.close();
970         } catch (IOException var260) {
971             System.out.println("Error closing file: " + var6);
972             System.exit(1);
973         }
974
975         var84 = var79.getIV();
976         var85 = var79.getIvParameterSpec();
977         var79.init(var84, var85);
978
979         try {
980             var88.close();
981         } catch (IOException var261) {
982             System.out.println("Error closing file: " + var6);
983             System.exit(1);
984         }
985
986         var84 = var79.getIV();
987         var85 = var79.getIvParameterSpec();
988         var79.init(var84, var85);
989
990         try {
991             var88.close();
992         } catch (IOException var262) {
993             System.out.println("Error closing file: " + var6);
994             System.exit(1);
995         }
996
997         var84 = var79.getIV();
998         var85 = var79.getIvParameterSpec();
999         var79.init(var84, var85);
1000
1001         try {
1002             var88.close();
1003         } catch (IOException var263) {
1004             System.out.println("Error closing file: " + var6);
1005             System.exit(1);
1006         }
1007
1008         var84 = var79.getIV();
1009         var85 = var79.getIvParameterSpec();
1010         var79.init(var84, var85);
1011
1012         try {
1013             var88.close();
1014         } catch (IOException var264) {
1015             System.out.println("Error closing file: " + var6);
1016             System.exit(1);
1017         }
1018
1019         var84 = var79.getIV();
1020         var85 = var79.getIvParameterSpec();
1021         var79.init(var84, var85);
1022
1023         try {
1024             var88.close();
1025         } catch (IOException var265) {
1026             System.out.println("Error closing file: " + var6);
1027             System.exit(1);
1028         }
1029
1030         var84 = var79.getIV();
1031         var85 = var79.getIvParameterSpec();
1032         var79.init(var84, var85);
1033
1034         try {
1035             var88.close();
1036         } catch (IOException var266) {
1037             System.out.println("Error closing file: " + var6);
1038             System.exit(1);
1039         }
1040
1041         var84 = var79.getIV();
1042         var85 = var79.getIvParameterSpec();
1043         var79.init(var84, var85);
1044
1045         try {
1046             var88.close();
1047         } catch (IOException var267) {
1048             System.out.println("Error closing file: " + var6);
1049             System.exit(1);
1050         }
1051
1052         var84 = var79.getIV();
1053         var85 = var79.getIvParameterSpec();
1054         var79.init(var84, var85);
1055
1056         try {
1057             var88.close();
1058         } catch (IOException var268) {
1059             System.out.println("Error closing file: " + var6);
1060             System.exit(1);
1061         }
1062
1063         var84 = var79.getIV();
1064         var85 = var79.getIvParameterSpec();
1065         var79.init(var84, var85);
1066
1067         try {
1068             var88.close();
1069         } catch (IOException var269) {
1070             System.out.println("Error closing file: " + var6);
1071             System.exit(1);
1072         }
1073
1074         var84 = var79.getIV();
1075         var85 = var79.getIvParameterSpec();
1076         var79.init(var84, var85);
1077
1078         try {
1079             var88.close();
1080         } catch (IOException var270) {
1081             System.out.println("Error closing file: " + var6);
1082             System.exit(1);
1083         }
1084
1085         var84 = var79.getIV();
1086         var85 = var79.getIvParameterSpec();
1087         var79.init(var84, var85);
1088
1089         try {
1090             var88.close();
1091         } catch (IOException var27
```

# Debugger

Dient der schrittweisen Ausführung des zu analysierenden Codes oder Hardware; ermöglichen zum Beispiel Speicherinspektion und Manipulation.

- gdb
- lldb
- x64dbg (Windows, Open-Source)
- jdb (Java Debugger)

---

Auch für das Debuggen von Hardware gibt es entsprechende Werkzeuge, z.B. [Lauterbach Hardware Debugger](#). Mittels solcher Werkzeuge ist es möglich die Ausführung von Hardware Schritt für Schritt ( *'single step mode'*) zu verfolgen und den Zustand der Hardware (Speicher und Register) zu inspizieren. Dies erfordert (z.Bsp.) eine JTAG Schnittstelle.

## 2. ERSCHWERUNG DES REVERSE ENGINEERING

Prof. Dr. Michael Eichberg

# Obfuscation (☞ Verschleierung)

- Techniken, die dazu dienen das Reverse Engineering zu erschweren.
- Häufig eingesetzt ...
  - von Malware
  - Adware (im Kontext von Android ein häufig beobachtetes Phänomen)
  - zum Schutz geistigen Eigentums
  - für DRM / Durchsetzung von Kopierrechten
  - zur Prävention von „Cheating“ (insbesondere im Umfeld von Online Games)
  - Wenn das Programm als Source Code vertrieben wird (JavaScript)
- Arbeiten auf Quellcode oder Maschinencode Ebene
- Grenze zwischen *Code Minimization*, *Code Optimization* und *Code Obfuscation* ist fließend.
- Mögliche Werkzeuge (ohne Wertung der Qualität/Effektivität):
  - [Java] Proguard / Dexguard
  - [C/C++] Star Force

17

Gerade im Umfeld von klassischen *Binaries* für Windows, Mac und Linux erhöhen Compiler Optimierungen, z.B. von C/C++ und Rust Compilern (**-O2** / **-O3**), bereits den Aufwand, der notwendig ist den Code zu verstehen, erheblich.

## Hinweis

Einen ambitionierten und entsprechend ausgestatteten Angreifer wird **Code Obfuscation** bremsen, aber sicher nicht vollständig ausbremsen und das Vorhaben verteilen.

# Obfuscation - Techniken (Auszug)

- entfernen aller Debug-Informationen
- das Kürzen aller möglichen Namen (insbesondere Methoden und Klassennamen)
- das Verschleiern von Konstanten durch den Einsatz vermeintlich komplexer Berechnungen zu deren Initialisierung.

```
~((int)Math.PI) ^ Integer.MAX_VALUE >> 16)+Short.MAX_VALUE  
= 2
```

# Obfuscation - Techniken (Auszug)

- die Verwendung von Unicode Codepoints für Strings oder die Verschleierung von Strings mittels rot13 Verschlüsselung.

```
/* ??? */ printf("\x48""e\x154l\x6F"" \127o\x72""l\144!");  
/* = */ printf("Hello World!");
```

- das Umstellen von Instruktionen, um das Dekompilieren zu erschweren
- das Hinzufügen von totem Code
- den relevanten Teil der Anwendung komprimieren und verschlüsseln und erst bei Verwendung entpacken und entschlüsseln.
- ...

## Umstellen von Instruktionen

Das Umstellen von Instruktionen erschwert die Analyse, da viele Werkzeuge zum Dekompilieren auf die Erkennung von bestimmten Mustern im Code angewiesen sind und ansonsten nur sehr generischen (Spaghetti Code) oder gar unsinnigen Code zurückgeben.

## Verschleierung von Strings

Das Verschleiern von Strings kann insbesondere das Reversen von Binärkode erschweren, da ein Angreifer häufig „nur“ an einer ganz bestimmten Funktionalität interessiert ist und dann Strings ggf. einen sehr guten Einstiegspunkt für die weitergehende Analyse bieten.

Stellen Sie sich eine komplexe Java Anwendung vor, in der alle Namen von Klassen, Methoden und Attributen durch einzelne oder kurze Sequenzen von Buchstaben ersetzt wurden und sie suchen danach wie von der Anwendung Passworte verarbeitet werden. Handelt es sich um eine GUI Anwendung, dann wäre zum Beispiel die Suche nach Text, der in den Dialogen vorkommt (z.B. **"Password"**) z.B. ein sehr guter Einstiegspunkt.

# 3. EINE SEHR KURZ EINFÜHRUNG IN JAVA

## BYTECODE

Prof. Dr. Michael Eichberg

# Die Java Virtual Machine

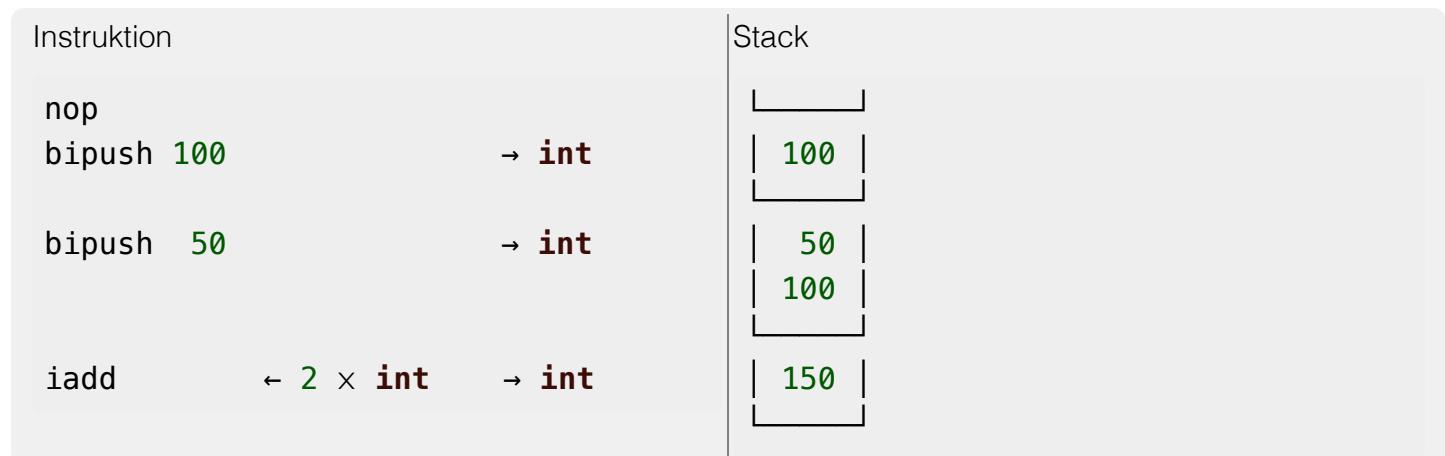
- **Java Bytecode** ist die Sprache, in der Java (oder Scala, Kotlin, Groovy, ...) Programme auf der Java Virtual Machine (JVM) [1] ausgeführt werden.
- In den meisten Fällen arbeiten Java Decompiler so gut, dass ein tiefgehendes Verständnis von Java Bytecode selten notwendig ist.
- Java Bytecode kann, muss aber nicht interpretiert werden. (z.B. können virtuelle Methodenaufrufe in Java schneller sein als in C++)

[1] Java Bytecode Spezifikation

21

# Java Bytecode - stackbasierte virtuelle Maschine

Die JVM ist eine stackbasierte virtuelle Maschine; die getypten Operanden eines Befehls werden auf einem Stack abgelegt und die Operationen arbeiten auf den obersten Elementen des Stacks. Jeder Thread hat seinen eigenen Stack.



- Die benötigte Höhe des Stacks wird vom Compiler berechnet und von der JVM überprüft.

# Java Bytecode - Methodenaufrufe und lokale Variablen

- Die Java Virtual Machine verwendet lokale Variablen zur Übergabe von Parametern beim Methodenaufruf.
- Beim Aufruf von *Klassenmethoden* (**static**) werden alle Parameter in aufeinanderfolgenden lokalen Variablen übergeben, beginnend mit der lokalen Variable 0. d.h. in der aufrufenden Methode werden die Parameter vom Stack geholt und in lokalen Variablen gespeichert.
- Beim Aufruf von *Instanzmethoden* wird die lokale Variable 0 dazu verwendet, um die Referenz (**this**) auf das Objekt zu übergeben, auf dem die Instanzmethode aufgerufen wird. Anschließend werden alle Parameter in aufeinanderfolgenden lokalen Variablen übergeben, beginnend mit der lokalen Variable 1.
- Die Anzahl der benötigten lokalen Variablen wird vom Compiler berechnet und von der JVM überprüft.

## Beispiel: *Default Constructor In Java Bytecode*

Ein *Constructor* welcher keine expliziten Parameter hat und nur den super Konstruktor aufruft.

```
// Method descriptor #8 ()V
// Stack: 1, Locals: 1
public Main();
  0  aload_0 [this]
  1  invokespecial java.lang.Object() [31]
  4  return
```

Die Zeilennummern und die Informationen über die lokalen Variablen ist optional und wird nur für Debugging Zwecke benötigt.

```
Line numbers:      [pc: 0, line: 9]
Local variable table: [pc: 0, pc: 5]  local: this
                           index: 0
                           type: de.dhbw.simplesecureapp.Main
```

Es gibt weitere Metainformationen, die „nur“ für Debugging-Zwecke benötigt werden, z.B. Informationen über die ursprünglich Quelle des Codes oder die sogenannte "Local Variable Type Table" in Hinblick auf generische Typinformationen. Solche Informationen werden häufig vor Auslieferung entfernt bzw. nicht hineinkompiliert.

# Beispiel: Aufruf einer komplexeren Methode

```
// Method descriptor #36 ([Ljava/lang/String;)V
// Stack: 5, Locals: 8
public static void main(java.lang.String[] args) throws ...;
  0  aload_0 [args]
  1  arraylength
  2  iconst_2
  3  if_icmpne 74           // integer comparison for equality
  6  getstatic java.lang.System.err : java.io.PrintStream
  9  ldc <String "SimpleSecure++">
11  invokevirtual java.io.PrintStream.println(java.lang.String) : void
...
...
```

## 4. VERSCHLÜSSELUNG VON DATEN

Prof. Dr. Michael Eichberg

# Alternativen zur Speicherung von Passwörtern

In einigen Anwendungsgebieten ist es möglich auf das explizite Speichern von Passwörtern ganz zu verzichten [\*].

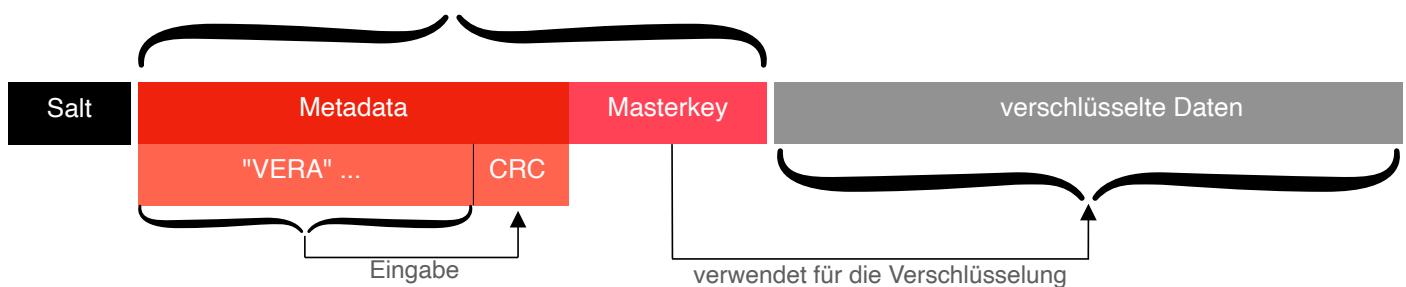
Stattdessen wird z.B. einfach versucht das Ziel zu entschlüsseln und danach evaluiert ob das Passwort (vermutlich) das Richtige war.

Kann darauf verzichtet werden zu überprüfen ob das Passwort korrekt war, dann sind keine Metainformationen notwendig und die verschlüsselte Datei kann genau so groß sein wie die unverschlüsselte Datei.

[\*] Bei einer Verschlüsselung mit OpenSSL wird das Passwort nicht gespeichert.

# schematische Darstellung der Verschlüsselung von Containern (z. B., Veracrypt)

Der Schlüssel wird mit Hilfe bekannter Schlüsselableitungsfunktionen aus dem Nutzerpasswort berechnet.



# Generische Dateiverschlüsselung ohne explizite Speicherung des Passworts

verschlüsselt mit einem vom Passwort des Nutzers abgeleiteten Schlüssel



Die Prüfsumme (CRC) oder der wohldefinierte Header werden zur Validieren des Passworts verwendet.

**Bleibe fokussiert!**

Analysiere nur was notwendig ist.