Code: analysis, bugs, and security

supported by Bitdefender

Obfuscation

Marius Minea

marius@cs.upt.ro

25 October 2017

Just for fun

The International Obfuscated C Code Contest
`http://www.ioccc.org/`

Best one-liner 2015: Visual factorization

```
f(y,x){int m,z;for(m=z=1;m*m<=y?z=y%m?z:m:x+1?z<2?y&&f(x,0):
f(z,x),putchar(x?10:32<<!y),y-=z*!!y:(f(z,y/z),0);)m++;}
main(y){f(y-1,-1);}
```

```
./a.out @ @ @ @ @ @ @ @
@ @ @ @
@ @ @ @
```

Obfuscation: what and why

Obfuscation = make code difficult to understand
but retain functionality (equivalent to original program)

Obfuscation: what and why

Obfuscation = make code difficult to understand
but retain functionality (equivalent to original program)

prevent reverse engineering

protect intellectual property

tamperproofing

Obfuscation can add *variability*

Obfuscation: what and why

Obfuscation = make code difficult to understand
but retain functionality (equivalent to original program)

prevent reverse engineering

protect intellectual property

tamperproofing

Obfuscation can add *variability*

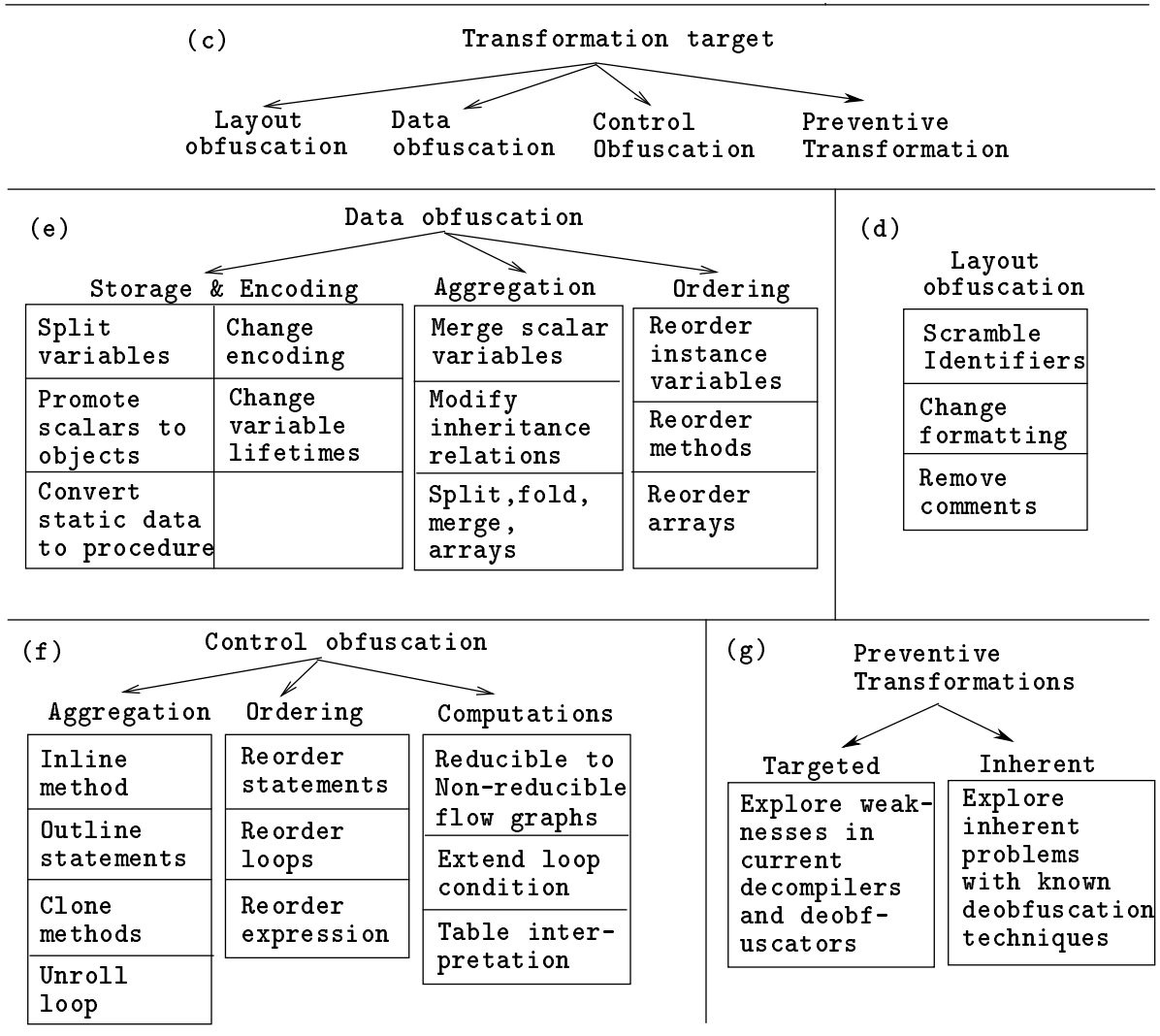
watermarking: trace origin of copies

prevent malware detection

but also protect against systematic attacks

Fred Cohen: Operating System Protection Through Program Evolution, 1992

Transformations for Obfuscation



Collberg, Thomborson, Low: A Taxonomy of Obfuscating Transformations, 1997

Evaluating Obfuscations

Criteria:

potency
To what degree is a human reader confused?

resilience
How well are automated deobfuscation attacks resisted ?

cost
How much space/time overhead is added ?

stealth
How well does obfuscated code blend in with original code ?

Collberg, Thomborson, Low, 1998

Complexity metrics

Metric	Metric Name	Citation
μ_1	Program Length	Halstead [8]
	$E(P)$ increases with the number of operators and operands in P.	
μ_2	Cyclomatic Complexity	McCabe [20]
	$E(F)$ increases with the number of predicates in F.	
μ_3	Nesting Complexity	Harrison [9]
	$E(F)$ increases with the nesting level of conditionals in F.	
μ_4	Data Flow Complexity	Oviedo [23]
	$E(F)$ increases with the number of inter-basic block variable references in F.	
μ_5	Fan-in/out Complexity	Henry [10]
	$E(F)$ increases with the number of formal parameters to F, and with the number of global data structures read or updated by F.	
μ_6	Data Structure Complexity	Munson [21]
	$E(P)$ increases with the complexity of the static data structures declared in P. The complexity of a scalar variable is constant. The complexity of an array increases with the number of dimensions and with the complexity of the element type. The complexity of a record increases with the number and complexity of its fields.	
μ_7	OO Metric	Chidamber [3]
	$E(C)$ increases with (μ_7^{a}) the number of methods in C, (μ_7^{b}) the depth (distance from the root) of C in the inheritance tree, (μ_7^{c}) the number of direct subclasses of C, (μ_7^{d}) the number of other classes to which C is coupled[a], (μ_7^{e}) the number of methods that can be executed in response to a message sent to an object of C, (μ_7^{f}) the degree to which C's methods do not reference the same set of instance variables. Note: μ_7^{f} measures *cohesion*; i.e. how strongly related the elements of a module are.	

Collberg, Thomborson, Low, 1998

Opaque constructs

should be easy to create, hard to analyze

opaque variable: has a property known a priori to the obfuscator, but hard to deduce otherwise

always constant value at some point, divisible by 7, etc.

opaque predicate:

outcome known at obfuscation time, hard to determine otherwise

from problems in math, number theory, etc.

Advanced obfuscation

control flow flattening (switch/automaton-style)

make building call graph hard
- insert calls through function pointers

introduce aliasing – alias analysis is hard

jumps through branch functions (change return address)

Virtualize
- encode program in virtual instruction set
- combine with interpreter
  - vary instruction set, even at runtime
- multi-level emulation

Deobfuscation

Obfuscation-specific

identify control flow (branches)

memory accesses: write then execute (for unpackers)

find virtual program counter $\Rightarrow$ reverse-engineer emulator

General

perform taint analysis (data flow)

compute input-output mapping

Obfuscation tools

Code Virtualizer `http://oreans.com/codevirtualizer.php`

ExeCryptor `http://www.strongbit.com/execryptor.asp`

Themida `http://www.oreans.com/themida.php`

VMProtect `http://vmpsoft.com/`

Tigress: `http://tigress.cs.arizona.edu/`
source-based transformations, using CIL infrastructure
+ virtualizer, JIT, and dynamic JIT

Obfuscator-LLVM:
`https://github.com/obfuscator-llvm/obfuscator`