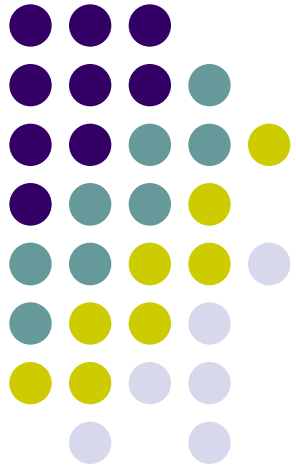
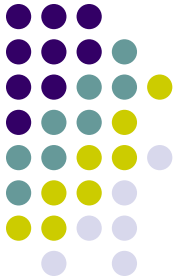
Sencun Zhu
Assistant Professor
CSE and IST, PSU

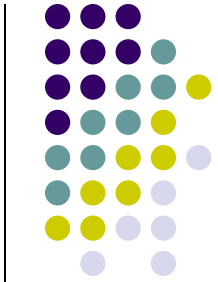


Research



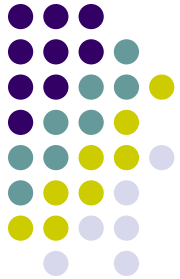
- Interests
  - Security, networking
- Recent professional activities
  - Program Co-Chair: ACM SASN'06.
  - TPC member: ACM CCS'07, IEEE Infocom'07, ESORICS'08, ACM WiSec'09, IEEE ICDCS'09, IEEE Security and Privacy'10…
  - Treasurer: ACM CCS'07-'10, AsiaCCS'10
- Research supports
  - Army Research Office (ARO), ARL, NSF CyberTrust, TTC,

Current Projects



- Security and privacy for sensor networks
  - Source location anonymity under different attack models
  - Applications of sensor networks to public safety
  - Sensor worm containment via software diversity
  - Defending against channel jamming and interference
- Security for MANET/DTN
  - Traceback in mobile ad hoc networks
  - Broadcast authentication in DTN
  - DoS detection and robust forwarding in DTN
- Security for cellular networks
  - Cellphone worm/malware detection and containment
- Code security
  - Blocking buffer overflow attacks by static code analysis
  - Code plagiarism detection

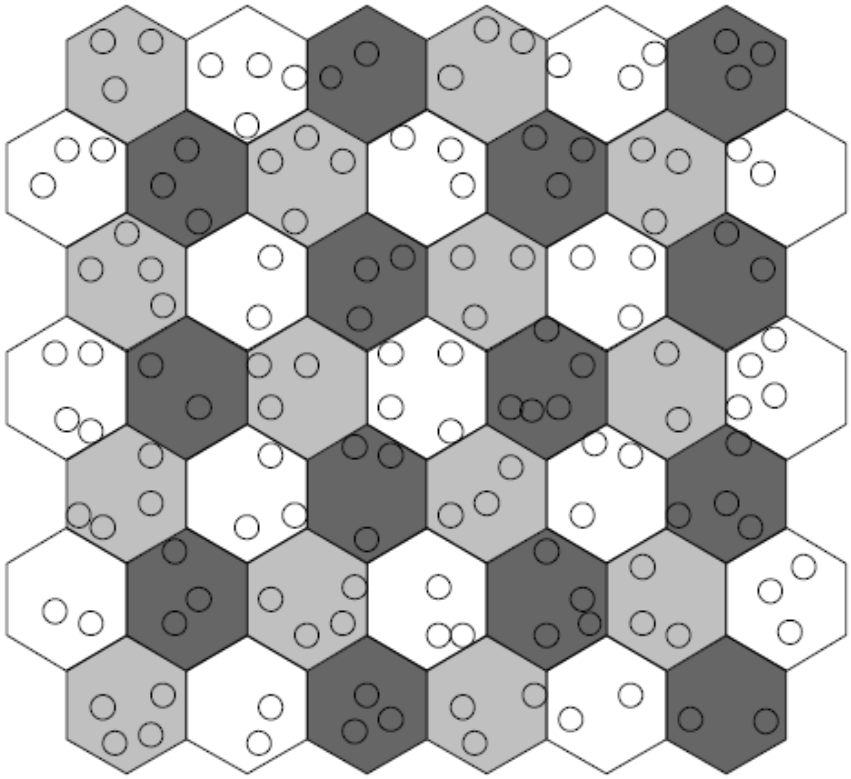
Sensor Worm Containment



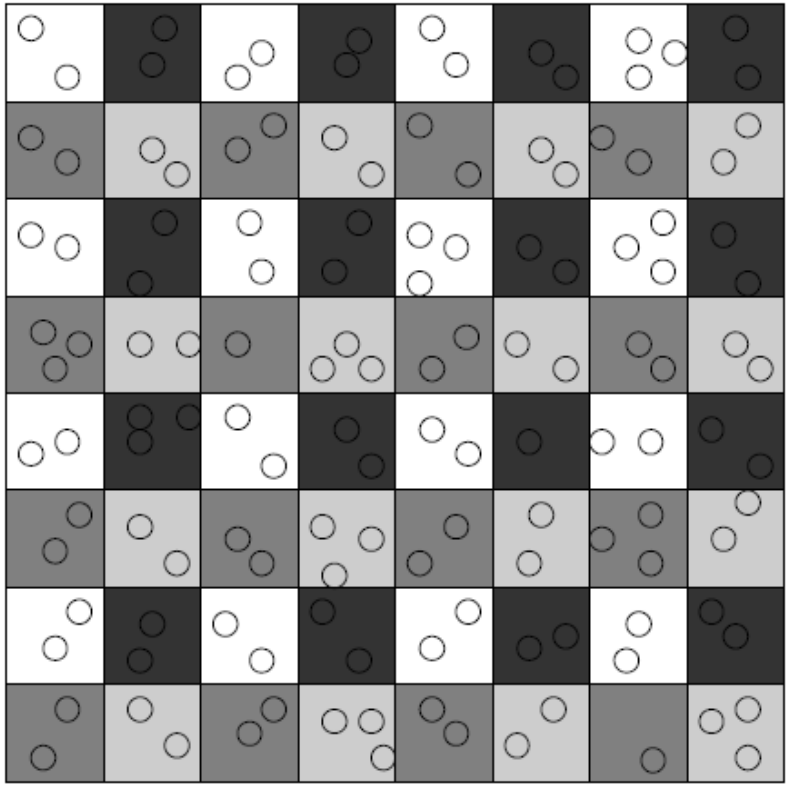
- Contribution I:
  - We illustrate the feasibility of sensor worms through trial experiments on Mica2 motes
- Contribution II:
  - In spirit of *survivability through heterogeneity* philosophy, we explore *software diversity* for sensor worm defense
    - Graph construction
    - Program assignment algorithm
    - Analysis on impact of deployment error to worm propagation
    - Simulation results to validate effectiveness of proposed scheme

Sensor Worm Defense

- Our solution
  - Partition the sensor field into cells, and assign a color to a cell



(a) 3-color case

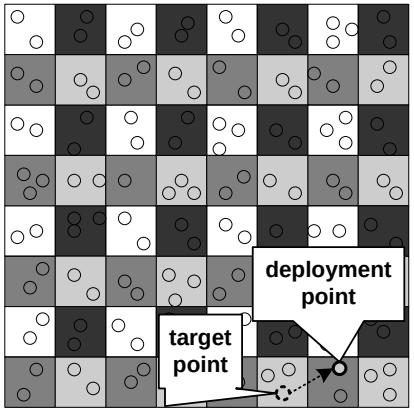


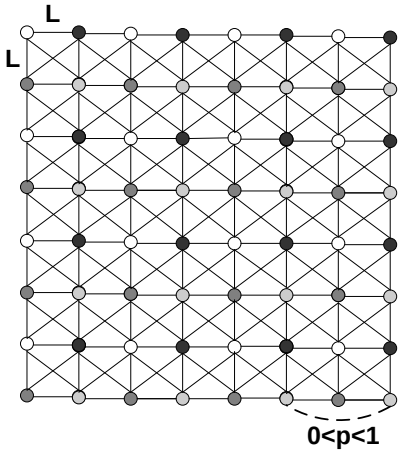
(b) 4-color case

The Impact of sensor deployment error

- *Deployment points* are modeled by *two-dimensional normal distribution* with *target points* as mean
- p_0: prob. that two nodes from neighboring cells with same color are connected

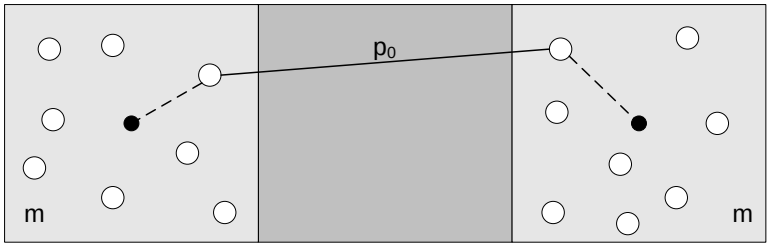
$$p_0 = \int_{x=0}^{X} \int_{y=0}^{Y} \frac{p_{n2}}{2\pi\sigma^2} e^{-[(x-x_1)^2+(y-y_1)^2]/2\sigma^2} dxdy$$



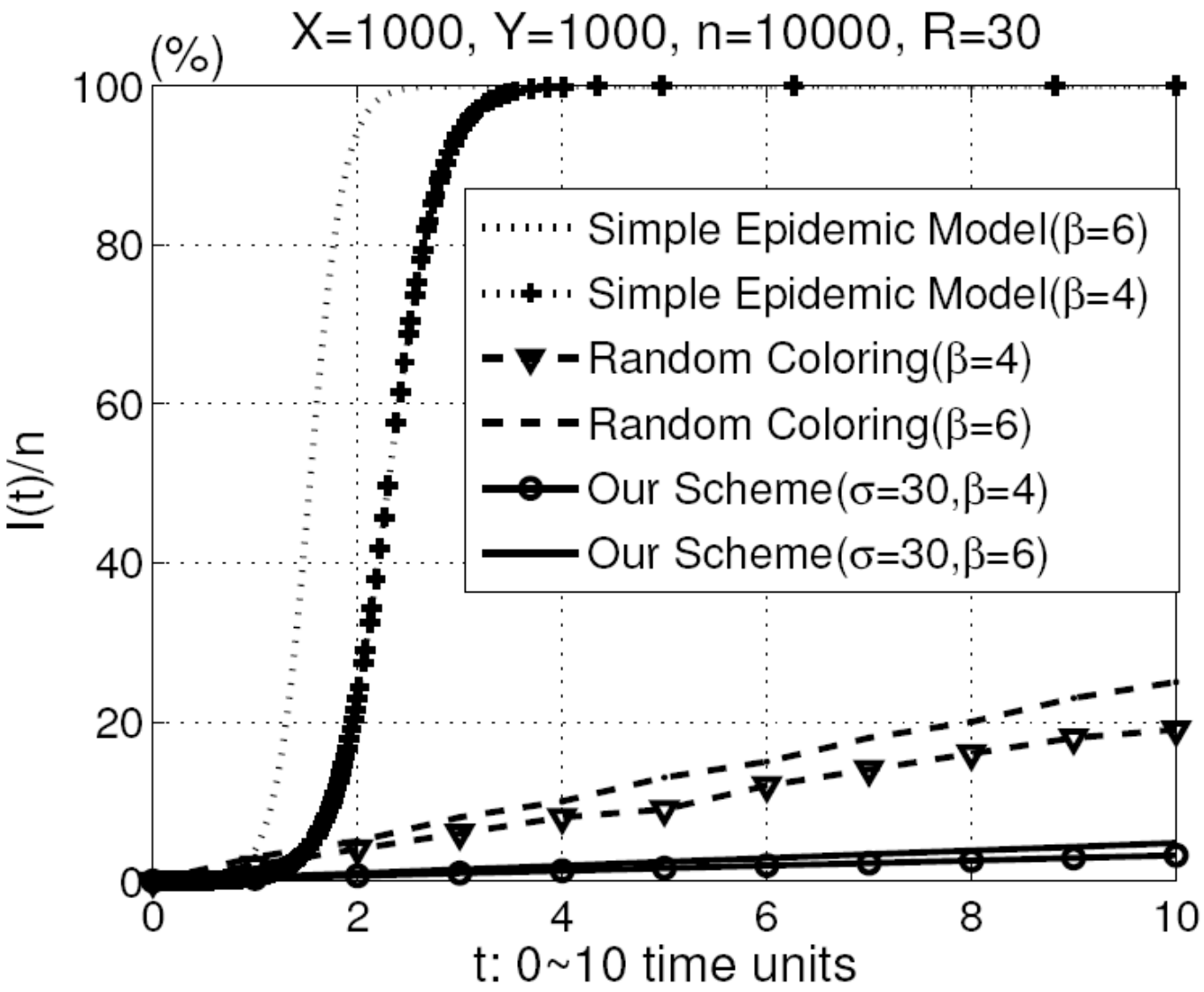


- p: prob. that two neighboring cells with same color are connected

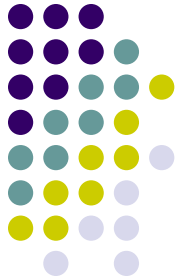
$$p = 1-(1-p_0)^{m^2}$$



Comparison

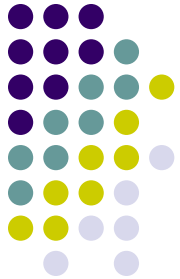

X=1000, Y=1000, n=10000, R=30
(%)
100
80
60
40
20
0
I(t)/n
0
2
4
6
8
10
t: 0~10 time units
Simple Epidemic Model(β=6)
Simple Epidemic Model(β=4)
Random Coloring(β=4)
Random Coloring(β=6)
Our Scheme(σ=30,β=4)
Our Scheme(σ=30,β=6)

Software Theft



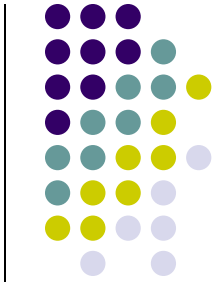
- A program steals code from open source projects
  - SourceForge.net has over 180,000 registered open source projects as of November 2008
- A company steals code from its competitor

Requirements for Software Theft Detection



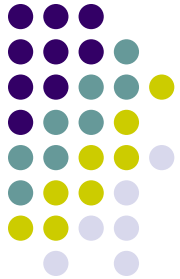
- Resiliency to obfuscation techniques
- Capability to detect theft of components
- Scalability to detect large-scale programs
- Applicability to binary executables

Existing Approaches



- Static analysis based theft detection
  - String-based
  - Token-based
  - Java class inheritance structure birthmark
  - …
- Dynamic analysis based theft detection
  - Java API birthmark
  - Whole program path birthmark
  - …

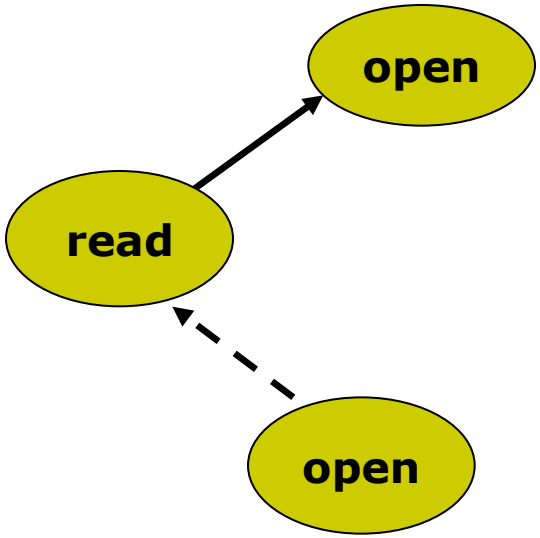
System Call Based Software Theft Detection



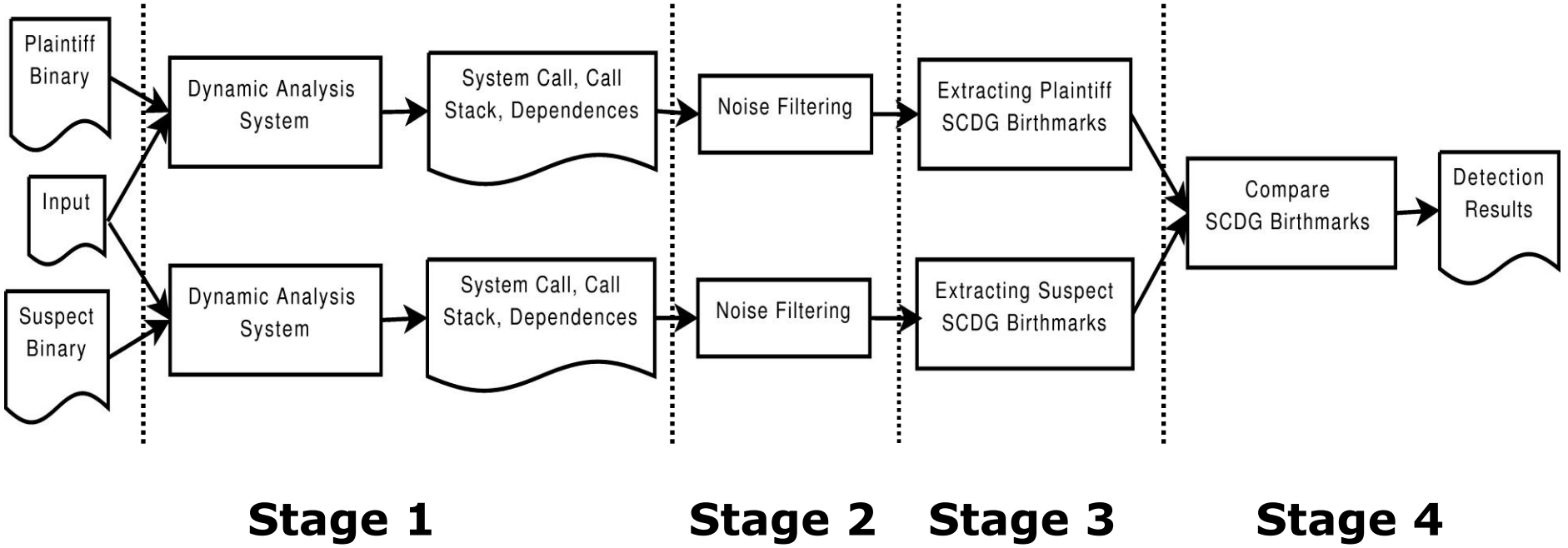
- System call -- the only way to talk with kernel
- System call dependence -- relation between system calls
- A dynamic system call based software birthmark for theft detection

SCDG (System Call Dependence Graph)

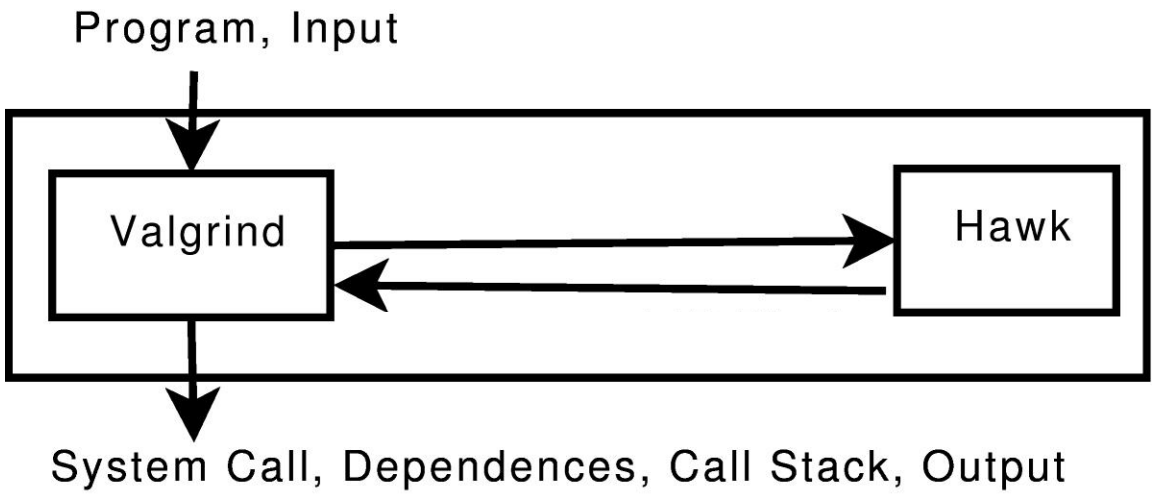
```
fd = open(path, …);
n = read(fd,buffer,10);
if (n < 10) {
   open(…);
}
```



System Overview

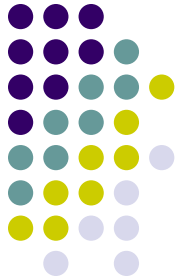


Dynamic Analysis System



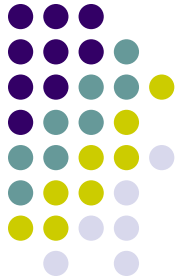
- Valgrind : A open source instrumentation framework for building dynamic analysis tools
- Hawk: A plugin tool we designed and developed for Valgrind

Noise Filtering



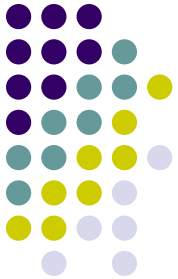
- System calls dependent on runtime environment are ignored
  - e.g. gettimeofday
  - e.g. memory management system calls
- Some system calls are considered the same
  - e.g. fstat(int fd, struct stat *sb) vs. stat(const char *path, struct stat *sb)
- Failed system calls are ignored

Birthmarks Extraction



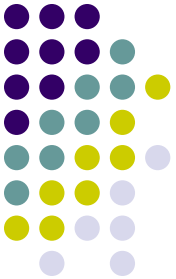
- Extraction of plaintiff birthmarks
  - Step 1: Build an SCDG for the plaintiff program
  - Step 2: Extract the SCDG for the component of interest
  - Step 3: Divide the extracted SCDG into subgraphs
  - Step 4: Remove the subgraphs which represent common behaviors
- Extraction of suspect birthmarks
  - Divide the suspect SCDG into subgraphs

Birthmarks comparison by γ-Isomorphism



- Graph Isomorphism
  - An equivalence relation
- Subgraph Isomorphism
  - A graph G is subgraph isomorphic to a graph G′, if there exists a subgraph $S \subset G'$ such that G is isomorphic to S
- γ-Isomorphism
  - A graph G is γ-isomorphic to G′ if there exists a subgraph $S \subseteq G$ such that S is subgraph isomorphic to G′, and $|S| \geq \gamma |G|$, $\gamma \in (0, 1]$

Experiment 1 – Two Component Birthmarks



- Subject software components
  - Aspell
  - Gecko
- Part of training data set

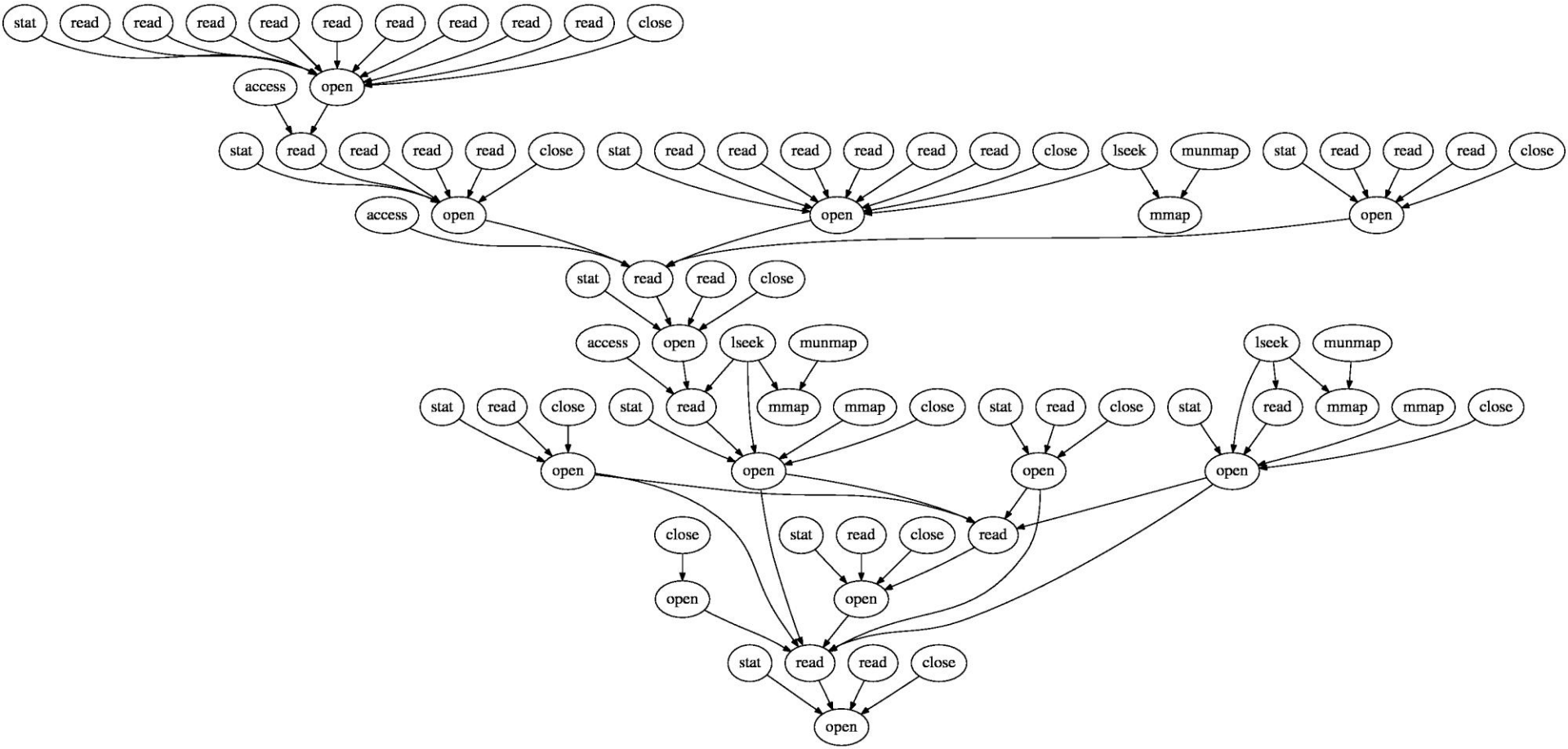
Program	Version	Type	SCDG	
			Node #	Edge #
Dillo	0.8.6	Web Browser	2612	1510
Yudit	2.4.1	Text Editor	4576	1023
Meld	1.1.5.1	Diff Viewer	12314	7084
Gimp	2.4.5	Graph Editor	59372	5972
Totem	2.22.1	Media Player	21865	6762
Pdfedit	0.3.2	PDF Editor	8937	4867
Dia	0.96.1	Diagram Drawing	27145	29185

Experiment 1 – Testing data set

Program	Version	Type	SCDG Node #	SCDG Edge #
Flock	2.0.3	Web Browser	21337	9343
Epiphany	2.22.2	Web Browser	16864	9011
Konqueror	3.5.10	Web Browser	11850	5589
Amaya	10	Web Browser	42701	23958
Opera	9.52	Web Browser	58485	21361
Songbird	1.1.2	Web Browser	37103	25547
Galeon	2.0.7	Web Browser	19825	7450
AbiWord	2.4.6	Word Processor	12975	5642
KWord	1.6.3	Word Processor	15408	6630
LyX	1.5.3	Latex Editor	21977	18656
Texmaker	1.6	Latex Editor	6897	3223
Kile	2.0.0	Latex Editor	50937	24615
Gedit	2.22.3	Text Editor	25113	5867
Bluefish	1.0.7	Text Editor	10952	3502
GNU Emacs	22.2.1	Text Editor	14807	4734
Vim	7.1.138	Text Editor	2582	1979
Pidgin	2.5.2	Messenger	10816	8014
Kopete	0.12.7	Messenger	16319	7144
Kmess	1.5	Messenger	10830	6247
GnoCHM	0.9.9	CHM Viewer	21191	8354
Evince	2.22.2	Doc. Viewer	16179	7095
GV	3.6.3	Doc. Viewer	6508	3267
Quod Libet	1.0	Media Player	15839	10725

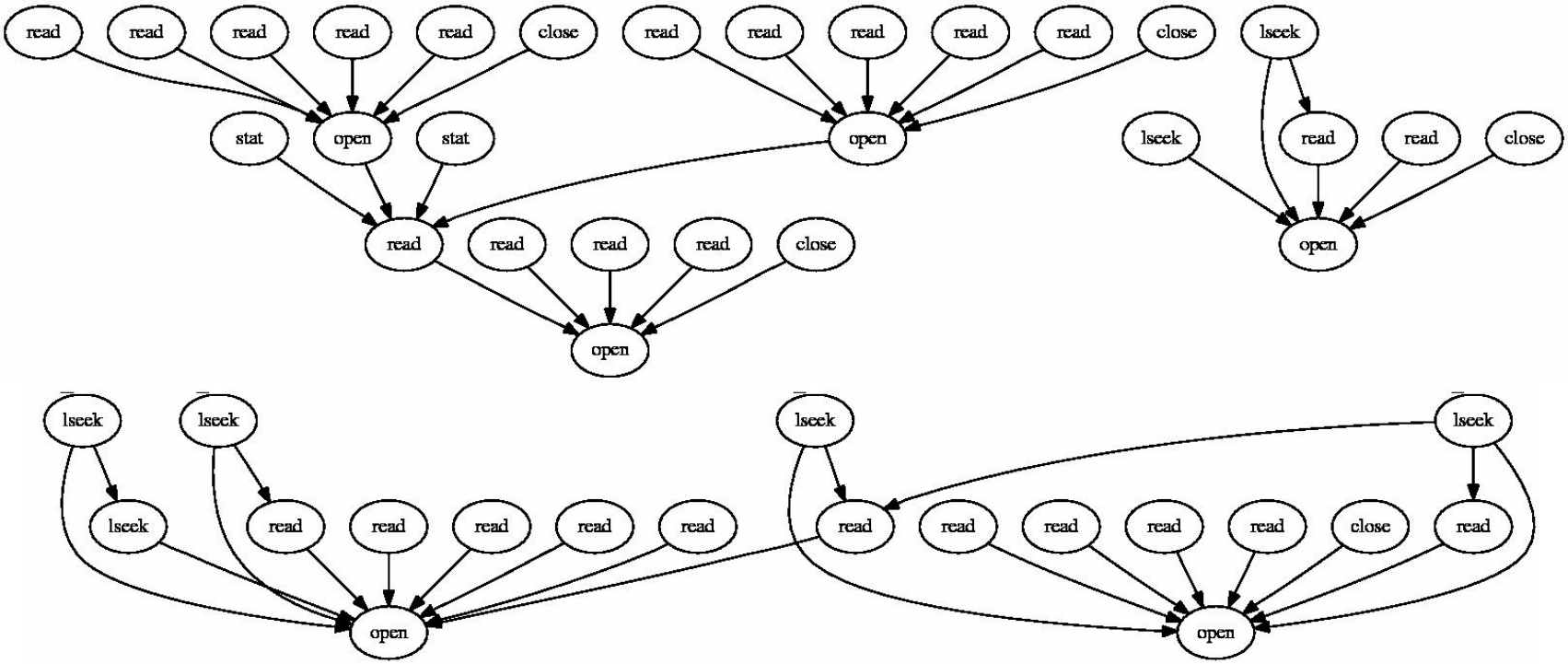
Experiment 1 – SCDG Birthmark of Aspell

- Software contains Aspell birthmark:
  - *Opera, Kword, Lyx, Bluefish, Pidgin*

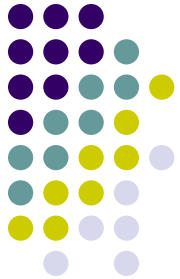


Experiment 1 - SCDG Birthmark of Gecko

- The following software contains Gecko birthmarks:
  - Flock, Epiphany, SongBird and Galeon

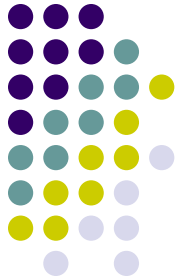


Experiment 2: Impact of Compiler Optimization Level



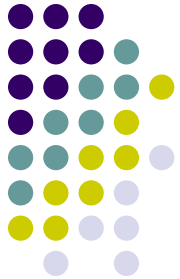
- Setup
  - Three programs: gzip, ogg vorbis, bzip2
  - Five optimization options (-O0,-O1,-O2,-O3 and -Os) of GCC
- Result
  - No change in system call traces for gzip and bzip2
  - The system call traces for ogg vorbis with option (-O0,-O1,-O2) have one “write” system call less than that with option –O3 and -Os

Experiment 3: Impact of Different Compilers



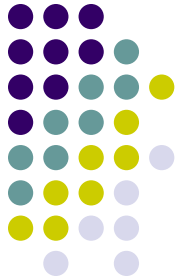
- Setup
  - Three programs: gzip, ogg vorbis, bzip2 with three compiler: GCC, TCC, Watcom
- Results
  - TCC = GCC ≠ Watcom (system call traces)
  - There are three types of differences between the traces of GCC and Watcom
    - Equivalent system calls (stat vs stat64)
    - Failed system calls
    - Memory management system calls
  - The difference can be removed by the noise filtering stage

Experiment 4: Impact of State-of-the-art Obfuscation Tools



- Setup
  - Three programs: gzip, ogg vorbis, bzip2 with two state-of-the-art obfuscation tools
  - Semantic Designs Inc's C obfuscator
    - Identifier scrambling, format scrambling, loop rewriting, and if-then-else rewriting and so on
  - Loco
    - Control flow flattening
- Result
  - The traces and SCDGs between original and obfuscated one are exactly the same

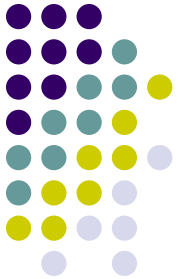
Limitation



- SCDG birthmarks is not applicable to the following cases
  - Programs or components which do not involve any system calls or have few system calls
  - Programs or components which do not have unique system call behaviors

Summary of SCDG Birthmark

- A new type of birthmarks – SCDG Birthmark
- *Resilient to various obfuscation techniques* by experiments
- *Detect component theft* in a set of *large software* by experiments



Thanks!

szhu@cse.psu.edu
www.cse.psu.edu/~szhu