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

$$
p_0 = \int_0^X \int_0^Y \frac{p}{2\pi\sigma^2} e^{-\frac{[(x-x_1)^2+(y-y_1)^2]}/2\sigma^2} dxdy
$$

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

$$
p = 1 - (1 - p_0)^m^2
$$
Comparison

$X=1000, Y=1000, n=10000, R=30$

- Simple Epidemic Model ($\beta=6$)
- Simple Epidemic Model ($\beta=4$)
- Random Coloring ($\beta=4$)
- Random Coloring ($\beta=6$)
- Our Scheme ($\sigma=30, \beta=4$)
- Our Scheme ($\sigma=30, \beta=6$)

$t: 0 \sim 10$ time units
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
fd = open(path, ...);
n = read(fd, buffer, 10);
if (n < 10) {
    open(...);
}
System Overview

Stage 1

Dynamic Analysis System → System Call, Call Stack, Dependences → Noise Filtering

Stage 2

Dynamic Analysis System → System Call, Call Stack, Dependences → Noise Filtering

Stage 3

Extracting Plaintiff SCDG Birthmarks

Stage 4

Compare SCDG Birthmarks → Detection Results
Dynamic Analysis System

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

Diagram:
- Program, Input
- System Call, Dependences, Call Stack, Output
- Valgrind
- Hawk

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 $\gamma$-Isomorphism

- Graph Isomorphism
  - An equivalence relation

- Subgraph Isomorphism
  - A graph $G$ is subgraph isomorphic to a graph $G'$, if there exists a subgraph $S \subseteq G'$ such that $G$ is isomorphic to $S$

- $\gamma$-Isomorphism
  - A graph $G$ is $\gamma$-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

<table>
<thead>
<tr>
<th>Program</th>
<th>Version</th>
<th>Type</th>
<th>SCDG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Node #</td>
</tr>
<tr>
<td>Dillo</td>
<td>0.8.6</td>
<td>Web Browser</td>
<td>2612</td>
</tr>
<tr>
<td>Yudit</td>
<td>2.4.1</td>
<td>Text Editor</td>
<td>4576</td>
</tr>
<tr>
<td>Meld</td>
<td>1.1.5.1</td>
<td>Diff Viewer</td>
<td>12314</td>
</tr>
<tr>
<td>Gimp</td>
<td>2.4.5</td>
<td>Graph Editor</td>
<td>59372</td>
</tr>
<tr>
<td>Totem</td>
<td>2.22.1</td>
<td>Media Player</td>
<td>21865</td>
</tr>
<tr>
<td>Pdfedit</td>
<td>0.3.2</td>
<td>PDF Editor</td>
<td>8937</td>
</tr>
<tr>
<td>Dia</td>
<td>0.96.1</td>
<td>Diagram Drawing</td>
<td>27145</td>
</tr>
</tbody>
</table>
## Experiment 1 – Testing data set

<table>
<thead>
<tr>
<th>Program</th>
<th>Version</th>
<th>Type</th>
<th>SCDG Node #</th>
<th>SCDG Edge #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flock</td>
<td>2.0.3</td>
<td>Web Browser</td>
<td>21337</td>
<td>9343</td>
</tr>
<tr>
<td>Epiphany</td>
<td>2.22.2</td>
<td>Web Browser</td>
<td>16864</td>
<td>9011</td>
</tr>
<tr>
<td>Konqueror</td>
<td>3.5.10</td>
<td>Web Browser</td>
<td>11850</td>
<td>5589</td>
</tr>
<tr>
<td>Amaya</td>
<td>10</td>
<td>Web Browser</td>
<td>42701</td>
<td>23958</td>
</tr>
<tr>
<td>Opera</td>
<td>9.52</td>
<td>Web Browser</td>
<td>58485</td>
<td>21361</td>
</tr>
<tr>
<td>Songbird</td>
<td>1.1.2</td>
<td>Web Browser</td>
<td>37103</td>
<td>25547</td>
</tr>
<tr>
<td>Galeon</td>
<td>2.0.7</td>
<td>Web Browser</td>
<td>19825</td>
<td>7450</td>
</tr>
<tr>
<td>AbiWord</td>
<td>2.4.6</td>
<td>Word Processor</td>
<td>12975</td>
<td>5642</td>
</tr>
<tr>
<td>KWord</td>
<td>1.6.3</td>
<td>Word Processor</td>
<td>15408</td>
<td>6630</td>
</tr>
<tr>
<td>LyX</td>
<td>1.5.3</td>
<td>Latex Editor</td>
<td>21977</td>
<td>18656</td>
</tr>
<tr>
<td>Texmaker</td>
<td>1.6</td>
<td>Latex Editor</td>
<td>6897</td>
<td>3223</td>
</tr>
<tr>
<td>Kile</td>
<td>2.0.0</td>
<td>Latex Editor</td>
<td>50937</td>
<td>24615</td>
</tr>
<tr>
<td>Gedit</td>
<td>2.22.3</td>
<td>Text Editor</td>
<td>25113</td>
<td>5867</td>
</tr>
<tr>
<td>Bluefish</td>
<td>1.0.7</td>
<td>Text Editor</td>
<td>10952</td>
<td>3502</td>
</tr>
<tr>
<td>GNU Emacs</td>
<td>22.2.1</td>
<td>Text Editor</td>
<td>14807</td>
<td>4734</td>
</tr>
<tr>
<td>Vim</td>
<td>7.1.138</td>
<td>Text Editor</td>
<td>2582</td>
<td>1979</td>
</tr>
<tr>
<td>Pidgin</td>
<td>2.5.2</td>
<td>Messenger</td>
<td>10816</td>
<td>8014</td>
</tr>
<tr>
<td>Kopete</td>
<td>0.12.7</td>
<td>Messenger</td>
<td>16319</td>
<td>7144</td>
</tr>
<tr>
<td>Kmess</td>
<td>1.5</td>
<td>Messenger</td>
<td>10830</td>
<td>6247</td>
</tr>
<tr>
<td>GnoCHM</td>
<td>0.9.9</td>
<td>CHM Viewer</td>
<td>21191</td>
<td>8354</td>
</tr>
<tr>
<td>Evince</td>
<td>2.22.2</td>
<td>Doc. Viewer</td>
<td>16179</td>
<td>7095</td>
</tr>
<tr>
<td>GV</td>
<td>3.6.3</td>
<td>Doc. Viewer</td>
<td>6508</td>
<td>3267</td>
</tr>
<tr>
<td>Quod Libet</td>
<td>1.0</td>
<td>Media Player</td>
<td>15839</td>
<td>10725</td>
</tr>
<tr>
<td></td>
<td>3.20.2</td>
<td>File Editor</td>
<td>12798</td>
<td>6707</td>
</tr>
</tbody>
</table>
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