Software Exploit Prevention and Remediation via Software Memory Protection

Clark Coleman, Jack Davidson, David Evans, John Knight, and Anh Nguyen-Tuong

Department of Computer Science, University of Virginia

Project Staffing

- **Investigators**
  - Clark Coleman (Research Scientist)
  - Jack Davidson, Dave Evans, John Knight (Faculty)
  - Anh Nguyen-Tuong (Senior Scientist)

- **Staff**
  - Ann Bailey (Research Assistant)
  - Mark Bailey (Visiting Associate Professor)
  - Michele Co (Research Scientist)
  - Jason Hiser (Research Scientist)
  - Nicholas Williams (Undergraduate)
Threat Model

- A program contains memory access vulnerabilities (not all memory operations are guaranteed to stay within proper bounds)
- An attacker can provide malicious input causing a memory overwriting error which compromises the program

Software Memory Protection (SMP)

- A general memory overwriting defense, including remediation
- Two components will operate directly on program binaries:
  - A Static Analyzer/Rewriter will analyze memory operations and vulnerabilities off line
  - A Memory Monitor SDT, a virtual execution environment that will watch vulnerable operations at run time and prevent an attacker from compromising the program
Software Exploit Prevention and Remediation via Software Memory Protection

- Find vulnerable program input locations
- Find critical data items that could be overwritten
- Record annotations to guide runtime SDT monitoring of input & critical data
- Ultra-efficient software dynamic translator analyzes and translates instructions at run time
- Static Binary Analyzer annotations guide attack detection and remediation

Packaging the Solution

- Customer will invoke a Linux shell script to protect an executable, e.g., `SMP-protect.sh foo.exe`
- Script will invoke the Static Analyzer, which produces the annotations file for use by mmStrata (the Memory Monitor SDT)
- Binary modification tool puts the executable under control of mmStrata
Packaging: Binary Modification Tool

- After modification, the protected executable will always run under control of mmStrata

![Diagram showing the flow of program code and data through the Binary Rewriter and mmStrata]

Staff Responsibilities

- Static Analyzer: Clark Coleman; design consultation provided by Mark Bailey and Jack Davidson
- mmStrata (the Memory Monitor SDT): Jason Hiser with heap memory portion assisted by Michele Co
- Binary modification tool work by Ann Bailey
- Evaluation, including development of exploits test suite: Michele Co, Nicolas Williams, Anh Nguyen-Tuong, with consultation provided by John Knight
Software Exploit Prevention and Remediation via Software Memory Protection

Milestones and Progress

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prot. 1: Coarse-grained Static Data</td>
<td>5/7/2007</td>
<td>8/31/2007</td>
<td>17w</td>
</tr>
<tr>
<td>4</td>
<td>Prot. 4: Fine-grained Heap</td>
<td>3/31/2008</td>
<td>5/2/2008</td>
<td>5w</td>
</tr>
<tr>
<td>5</td>
<td>Adaptive learning scheme</td>
<td>5/5/2008</td>
<td>7/4/2008</td>
<td>9w</td>
</tr>
<tr>
<td>6</td>
<td>Repair and recovery</td>
<td>7/7/2008</td>
<td>9/5/2008</td>
<td>9w</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation and documentation</td>
<td>9/8/2008</td>
<td>10/3/2008</td>
<td>4w</td>
</tr>
<tr>
<td>8</td>
<td>Optimization</td>
<td>9/3/2007</td>
<td>5/2/2008</td>
<td>35w</td>
</tr>
<tr>
<td>9</td>
<td>Improving Granularity</td>
<td>9/3/2007</td>
<td>5/2/2008</td>
<td>35w</td>
</tr>
</tbody>
</table>

![Progress: September - present](image)

- Prototype 2 (coarse-grained defense of stack and heap pointers) is nearing completion on schedule
  - False negatives are already near zero, will be zero soon
  - Working on reducing false positives
- At late January PI meeting, we will demonstrate a coarse-grained defense of all pointers (global, stack, and heap)
  - Packaged for easy invocation of the system
  - Most published exploits are coarse-grained
  - We will measure the run-time benefits of optimizations that will be completed in December and January
Software Exploit Prevention and Remediation via Software Memory Protection

Evaluation

- Several sources of exploits and benchmarks
  - Wilander buffer overflow test suite
  - NIST SAMATE (Software Assurance Metrics And Tool Evaluation) vulnerability detection test suite (http://samate.nist.gov/)
  - SPEC 2000 benchmark suite (for measuring overhead)
  - Vulnerable version of Apache web server
  - Others will be added over time

- Testing and evaluation are being performed on x86 Linux servers dedicated to our work

Summary of Expenses

<table>
<thead>
<tr>
<th>Month</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2007</td>
<td>$30,298</td>
</tr>
<tr>
<td>May 2007</td>
<td>$38,400</td>
</tr>
<tr>
<td>June 2007</td>
<td>$21,448</td>
</tr>
<tr>
<td>July 2007</td>
<td>$88,748</td>
</tr>
<tr>
<td>August 2007</td>
<td>$74,010</td>
</tr>
<tr>
<td>September 2007</td>
<td>$74,279</td>
</tr>
<tr>
<td>October 2007</td>
<td>$83,224</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>$410,407</strong></td>
</tr>
</tbody>
</table>
Upcoming Work

- Provide fine-grained defense of pointers
  - Static Analyzer will need significant new analyses
  - mmStrata should need little additional work, being driven by the new annotations
  - Binary modification tool and script are unaffected
- Design and implement remediation (recovery and repair schemes)
- Continual optimization to reduce run time overhead

Technology Transfer

- Primarily a matter of porting software to new targets
  - Current project only targets x86/Linux
- Strata is designed to be portable and already has been ported to ARM/Linux, Sparc/Solaris
  - Windows port would be useful
- mmStrata will need some porting work for each target
- Static Analyzer is designed to separate, and identify with comments, the machine dependent code
  - IDA Pro has been ported to more than 40 targets
- Binary modification tool will need a small amount of work for each new target
Summary

- A general memory overwriting defense has been developed using several components
- Each component is well staffed
- Prototypes are being delivered on schedule
- Finances are right on target
- Feedback from James Sidoran:
  - Develop specific plans for follow-on work
  - Develop ideas for technology transfer
  - Plan ahead for both the late January demo and the June demo; prepare discussion points in advance for both