Revamping Security Patching with Virtual Patches

Gautam Altekar

University of California, Berkeley
Problem

• Conventional security patching is ineffective
  – Users don’t patch their systems in time
• Why don’t people patch?
  – Patches are unreliable
  – Patches are disruptive
  – Patches are irreversible
• We need to rethink the way we create and apply security patches
A simple observation

- Many existing security patches have two parts:
  1. a check
  2. a fix
- Many bug fixes can be written this way

Bind 8.2.2 division by 0 bug

```c
choice = (((u_int)rand())>>>3) %
         non_sig_count;
```

Bind 8.2.2 vendor patch

```c
if (non_sig_count <= 0) {
    non_sig_count = 1;
}

choice = (((u_int)rand())>>>3) %
         non_sig_count;
```
What is a “virtual patch”?

- Programmer inserted code that has two clearly denoted parts:
  1. a check and
  2. a fix
- Sandbox the check, but not the fix

Binder 8.2.2 division by 0 bug

```
choice = ((u_int)rand() >> 3) %
       non_sig_count;
```

Binder 8.2.2 virtual patch

```
BEGIN_CHECK;
if (non_sig_count <= 0) {
  BEGIN_FIX;
  non_sig_count = 1;
  END_FIX;
}
END_CHECK;
```

```
choice = ((u_int)rand() >> 3) %
       non_sig_count;
```
Virtual patches are reliable

• Guarantee: the patch will not side-effect your application until the fix is applied
  – Sandbox the check using Software Fault Isolation (Wahbe et al. ‘93)
  – Internally represent each check and fix as a nested C function
  – Much SFI overhead can be optimized out
    • Total overhead = ~50 cycles for patches we have tested
Most writes are to the stack and can be statically optimized out.

Most jumps are direct and can be statically verified.
Virtual patches are non-disruptive

- Put virtual patch code in dynamic library
- Use ptrace(2) to:
  - Dynamically load the virtual patch DLL
  - Modify process to invoke virtual patch code at programmer inserted location
- ==> Virtual patches are reversible
Is it practical?

- Problem: programmer has to explicitly denote the check and the fix
  - Departs from established patching practices
- Question: is it possible to automatically derive the check and fix?
  - Assume you have access to the conventional security patch
- Conjecture: there exists a virtual patch for any conventional security patch
Limitations

• Patch programmer may screw up the check
  – False negatives - benign
  – False positives - dangerous

• Patch programmer may screw up the fix
  – Program may crash or worse…