Binary Modification with Dyninst

Hot-patching Apache Security Flaws

Modification Uses

- Understanding program behavior:
  - Program testing, dynamic patching, …
- Understanding performance characteristics:
  - Optimization, performance analysis, …
- Understanding security characteristics:
  - Attack detection, behavior monitoring, cyberforensics, …

Structured Binary Editing

- Modify a program by transforming its control flow graph (CFG)
- Uses an algebra of pre-defined structurally valid transformations
- Provides safe modification with no instruction-level user knowledge
- Works on running programs or binaries on disk
- Interactive; user modifications are represented in the CFG and can be further transformed.

Case Study: Hot-Patching Apache

Methodology

- Patch three Apache vulnerabilities:
  - CVE-2011-3368: bypass reverse proxy server
  - CVE-2011-3607: privilege escalation via .htaccess file
  - CVE-2012-0021:daemon crash via malformed cookie
- Convert available patch to CFG transformation
- Apply to running, unmodified Apache server and verify

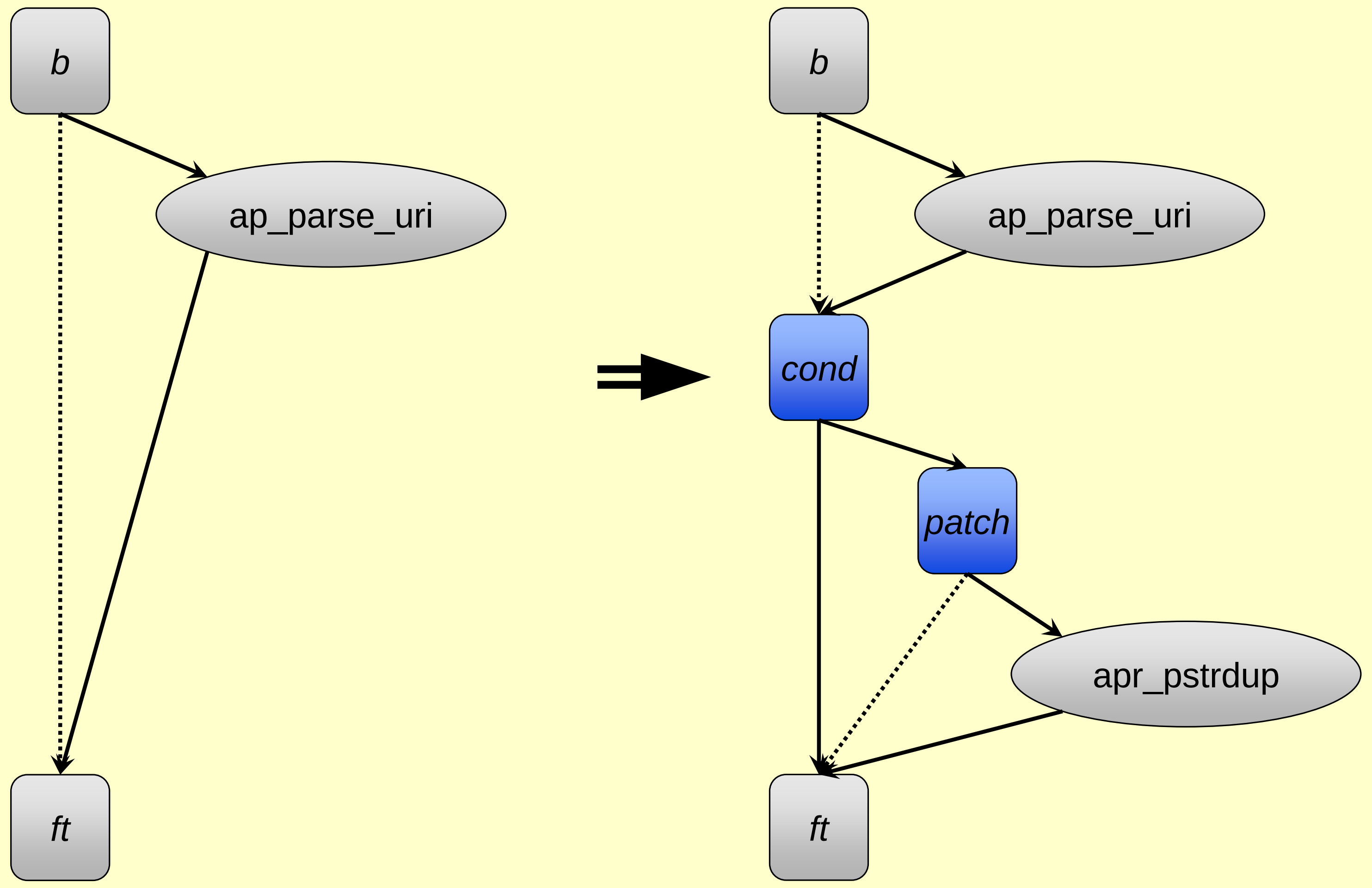
CVE-2011-3368 Patch

```
  ap_parse_uri(r, uri);

+ if (r->method_number != M_CONNECT
+     && !r->parsed_uri.scheme
+     && uri[0] != ‘/’
+     && !(uri[0] == ‘*’ && uri[1] == ‘\0’)) {
+       r->args = NULL;
+       r->hostname = NULL;
+       r->status = HTTP_BAD_REQUEST;
+       r->uri = apr_pstrdup(r->pool, uri);
+ }

  if (ll[0]) {
```

Dyninst Code Sequence

```
bool insertSnippet(PatchBlock *b, SnippetPtr snip, Point *point) {
   // Find post-call block
   PatchBlock *ft = getSuccessor(b, ParseAPI::CALL_FT);

   // Insert new code region into the CFG
   InsertedCode::Ptr ins = PatchModifier::insert(b->obj(), snip, point);

   // Find entry of new code region
   PatchBlock *cond = ins->entry();

   // Redirect the call fallthrough of b to cond instead of ft
   PatchModifier::redirect(getEdge(b, ParseAPI::CALL_FT), cond);

   // Redirect all exits of new code region to ft
   for (unsigned i = 0; i < ins->exits().size(); ++i) {
      PatchModifier::redirect(ins->exits()[i], ft);
   }
   return true;
}
```

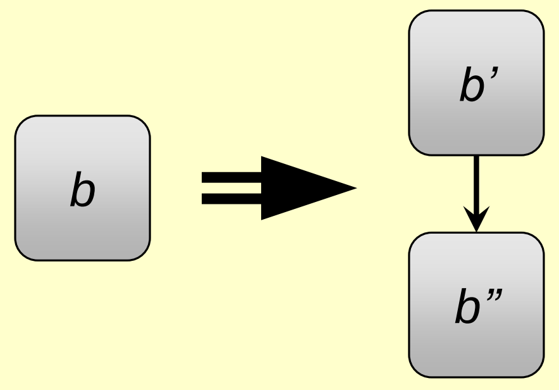
CFG Transformation



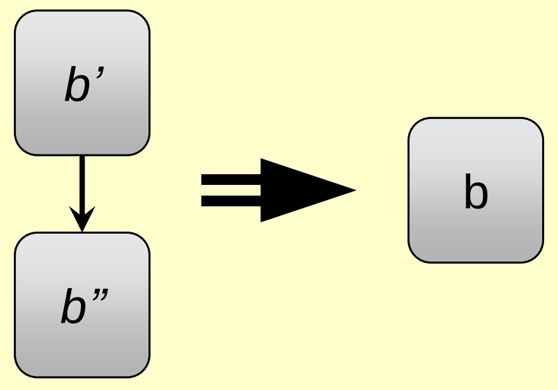
CFG Transformation Examples

Basic Block Transformations

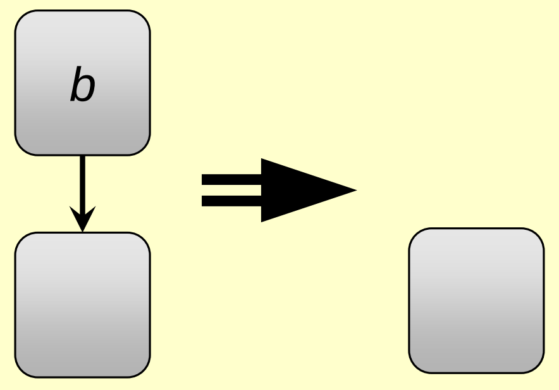
Block Splitting



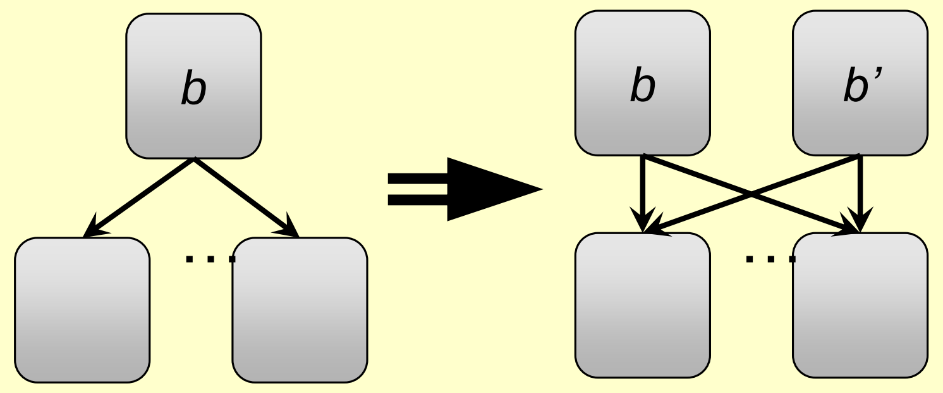
Block Joining



Block Removal

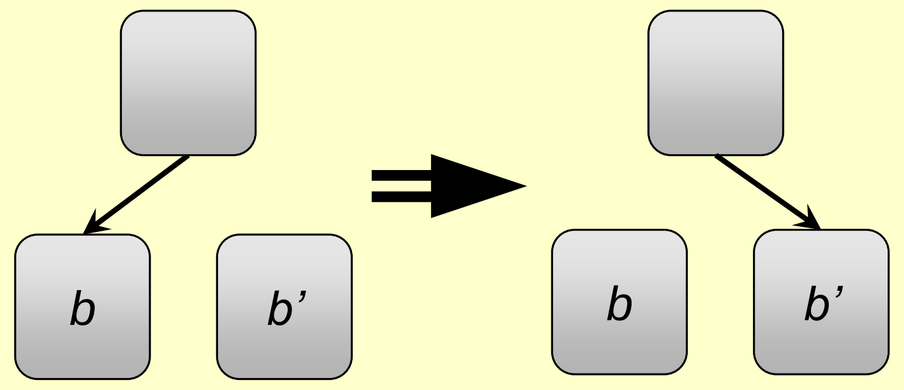


Block Cloning

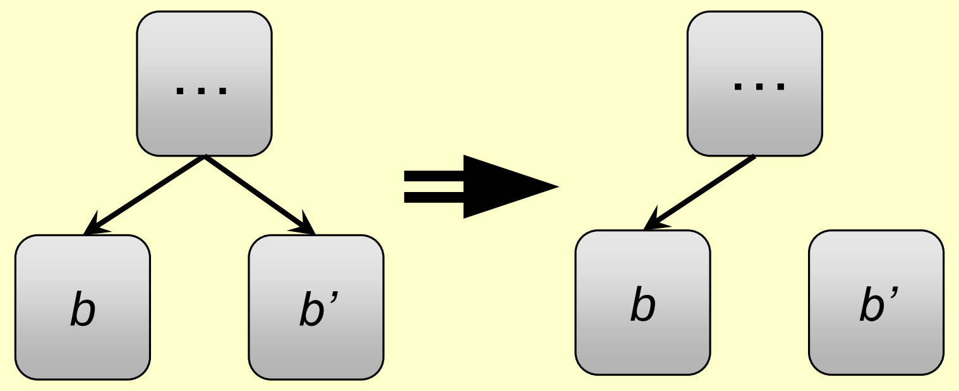


Edge Transformations

Edge Redirection

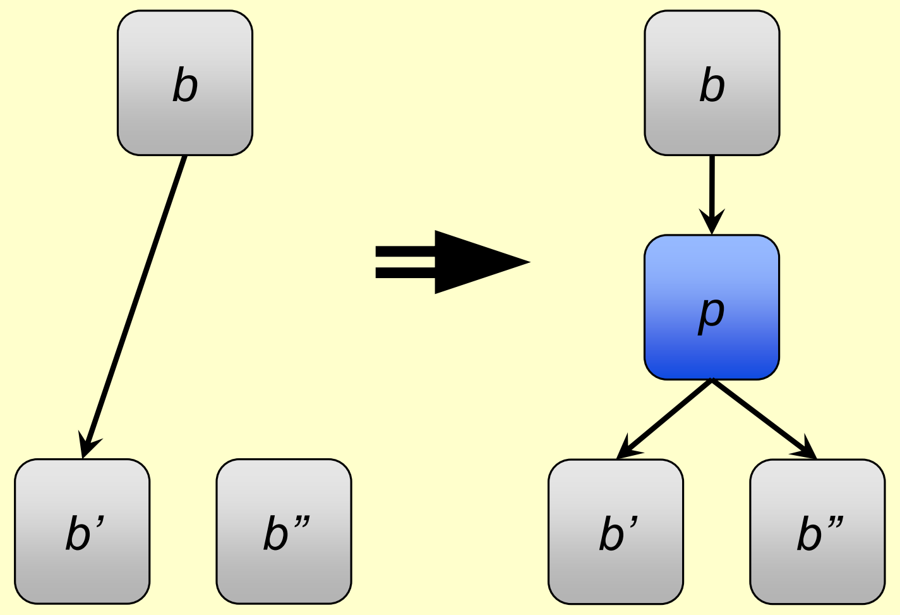


Edge Collapsing

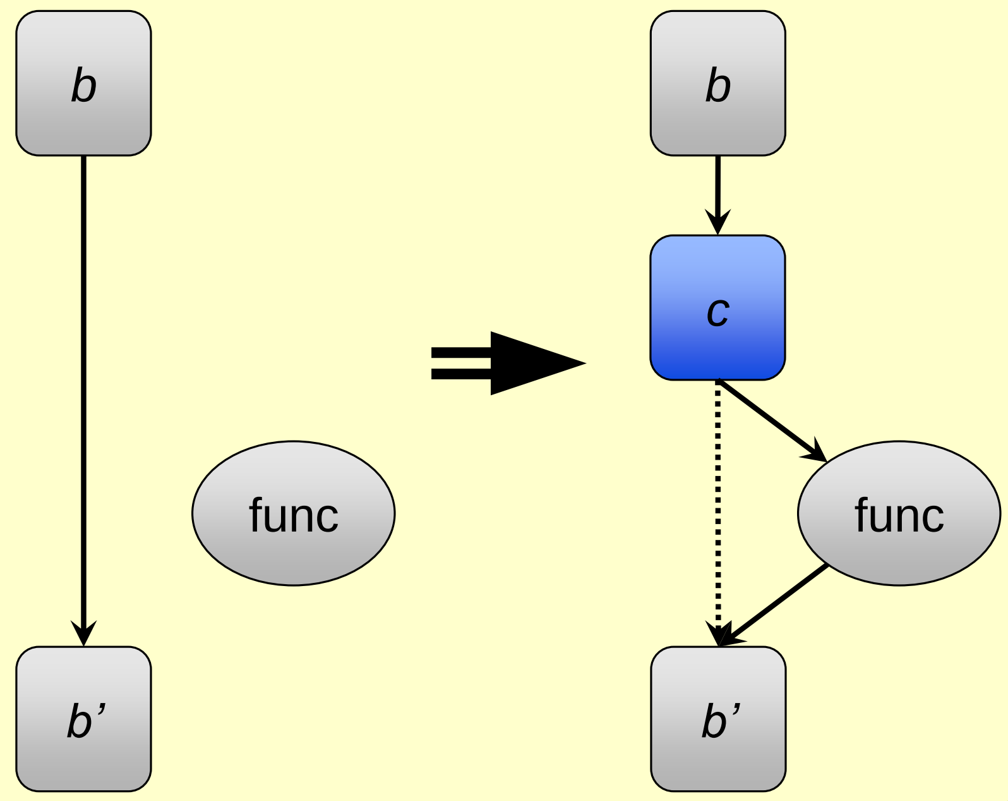


Code Insertion Transformations

Predicate Insertion



Call Insertion



Andrew R. Bernat and Barton P. Miller, “Structured Binary Editing with a CFG Algebra,” *Working Conference on Reverse Engineering (WCRE),* Kingston, Ontario, Canada, October 2012