Automatic Worm Defense (II) -- More on Automatic Signature Generation

Dawn Song

dawnsong@cs.berkeley.edu

Central Question

- **Given an exploit to a vulnerability, how to generalize to create an effective signature?**
- **Key: identify constraints on inputs**
  - **Reachability condition**
    - » **Program execution reaches vulnerability point**
  - **Vulnerability condition**
    - » **Triggers vulnerability at vulnerability point**
- **Idea: given an exploit**
  - **Identify vulnerability condition**
  - **Generalize reachability condition**

Background: Exploit Detector (I)

- **Exploit detector monitors for runtime memory safety violations**
- **Source-based mechanisms**
  - **Runtime type check: e.g., CCured**
  - **Array bounds check: e.g., CRED**
  - **Detect illegitimate writes: e.g., DFI (Data Flow Integrity)**
  - **Protecting activation records: e.g., StackGuard**
- **Binary-only mechanisms**
  - **Dynamic taint analysis**

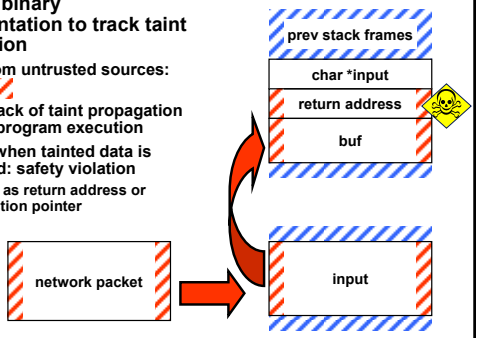
HTTP-like Example

```
1.  int check_http( char *input ) {
2.    char buf[8];
3.    if (strncmp(input, “get”,3) != 0 &&
4.        strncmp(input, “put”,3) != 0 )
5.      return -1;
6.    if (input[3] != ‘/‘ ) return -1;
7.    strncpy( buf, input, 4);
8.    int i = 4;
9.    while ( input[i] != ‘\n‘)
10.     { buf[i] = input[i];
11.       i++; }
12.   return i;
13. }
```

Vulnerability condition: i ≥ 8

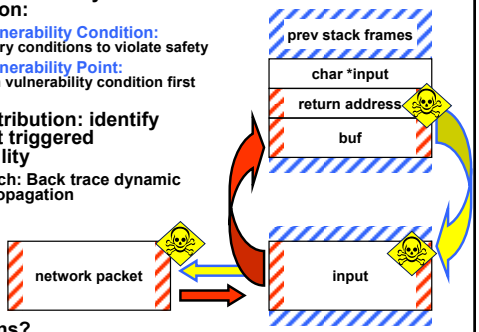
Dynamic Taint Analysis

- **Dynamic binary instrumentation to track taint propagation**
  - Data from untrusted sources: tainted
  - Keep track of taint propagation during program execution
  - Detect when tainted data is misused: safety violation
    - » *e.g.*, as return address or function pointer



Automatic Diagnosis

- **Extract vulnerability information:**
  - The Vulnerability Condition: Necessary conditions to violate safety
  - The Vulnerability Point: Location vulnerability condition first satisfied
- **Attack attribution: identify input that triggered vulnerability**
  - Approach: Back trace dynamic taint propagation
- **Limitations?**



Background: Exploit Detector (II)

- **Necessary first step for automatic signature generation**
- **Why not just use exploit detector instead of input filter?**
  - **Runtime overhead**
  - **When detecting the attack, may already be too late**
    - » **May have to restart server**
    - » **Even exceptions may not be handled well in type-safe languages**

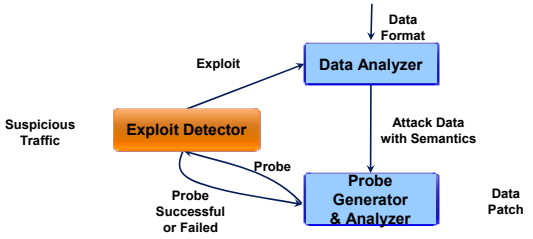
ShieldGen: Automatic Data Patch Generation for Unknown Vulnerabilities with Informed Probing

Main Idea (I)

- **What to generalize from original exploit?**
  - **Vulnerability condition**
    - » **Buffer length condition for buffer overflows**
  - **Reachability condition**
    - » **Remove unnecessary fields/iterations**
    - » **Widening field values**

Main Idea (II)

- How to generalize from original exploit?
  - Guided probing to generate new exploits
  - Use new exploits to relax condition



Why Use a Data Analyzer?

- Constraints are often on substrings in message with semantics
  - Express constraints and perform matching
- To generate legitimate probes
  - Reduce # of probes tested
  - Not to overly constraint certain values

Probe & Signature Generation

- Vulnerability condition
  - Heuristics to identify buffer overflows
  - Heuristics to identify buffer length condition for buffer overflows
- Reachability condition
  - Remove unnecessary fields/iterations
    - » Remove them and gradually add back in to generate probes
    - » Remove from signature if not needed for a successful exploit
  - Widening field values
    - » Sampling field values to generate probes
    - » Remove don't-care fields from signature

Comparison with Pattern-Extraction based Approach

- **Pattern-extraction based approach**
  - Passively wait for more exploits
  - Learning without semantics/protocol parsing
- **Added assumptions**
  - Access to exploit detector
  - Access to data analyzer

Limitations (I)

- **Data analyzer assumption**
  - Not always available
    - » Important for new attacks
    - » May be deeper level than message parsing
  - Difference btw protocol specification & real implementation
    - » How did ShieldGen try to address this issue?
- **Buffer overflow heuristics**
  - How to fix it?
- **Offending byte identification**
  - Complex calculation could involve many bytes in input
- **Probe generation**
  - Require accurate data analyzer
  - Iteration removals/Sampling techniques miss values
    - » How to fix it?
  - Combinatoric explosion for complex conditions

Limitations (II)

- **Signature generation**
  - No guarantees
  - False positives?
  - False negatives?
- **What types of vulnerabilities is this applicable to?**
- **Other thoughts?**

Star Paper Summary #1

- **Que 1: Design your favorite botnet**
  - **Emphasize on attack-resilient strategies & technologies**
  - **How to design architecture for command-&-control & communication**
- **Que 2: What do you think are the necessary ingredients for defending against future botnets?**
  - **E.g., absolute host security?**
  - **E.g., authenticated traffic?**
- **Que 3: Can you think of a sufficient recipe for defending against future botnets?**
- **Hand-in:**
  - **Hard copy in class at beginning of Mon class**
  - **Electronic copy before Mon class**