Synthesis of a Complex Software Vulnerability Analyzer (SVA)

Kestrel Institute
Jim McDonald
March 7, 2002
Outline

- Goals
- Project strategy and flow
- Initial success
- Implementation of tool
- Demo
- Taxonomies
- Current vision
SVA Project Goals

- Build characterization of vulnerabilities to support automated analysis
  - Semantic rigor
  - Organized / Modular
  - Reusable
  - Extendable

- Build inference & analysis tools to detect vulnerabilities
  - Automation
  - Mixed initiative

- Demonstrate detection of real vulnerabilities
Subtle flaws

- Elude smart compiler – buffer overflow detection increasingly tractable
- Multiple element interactions – possibly great complexity
- Handle protocol implementations – optimization can cloud interactions
- Typically require human assessment & guided search to assess impact
SVA Project flow

Actual flaw case → Flaws language → Semantic taxonomy → Automated tools
I've discovered a pair of new capabilities in Java, one residing in the Java core and the other in Netscape's Java distribution. The first (exploited in $\text{BOServerSocket}$ and $\text{BOSocket}$) allows Java to open a server which can be accessed by arbitrary clients. The second ($\text{BOURLConnection}$ and $\text{BOURLInputStream}$) allows Java to access arbitrary URLs, including local files.

As a demonstration, I've written $\text{BOHTTPD}$ for Netscape Communicator. $\text{BOHTTPD}$ is a browser-resident web server and file-sharing tool that demonstrates these two problems in Netscape Communicator. $\text{BOHTTPD}$ will serve files from a directory of your choice, and will also act as an HTTP/FTP proxy server. [ed note: “open door”]
Two days later

[ed note: text taken from Dan Brumleve’s website]

2000.08.05

Right now I'm at the internet cafe (Club I) at 850 Folsom in San Francisco (between 4th and 5th street). I'll be here until 2:00 a.m. showing demos to anybody interested.

A guy showed up here and made BOHTTPD multithreaded. This new functionality is live right now…

WHOA! I just saw a Windows 2000 system that was still running BOHTTPD even after Netscape had been apparently terminated. Even the "Task Manager" showed no trace.  

[ed note: “door stays open”]
Connecting flaw concepts to code

Mine rich sources of flaws

Select flaw primitives for a language

Create tools for making queries on (byte)code

- CERT
- BUGTRAQ

- Sensitive Regions
- Spoofable Methods

Byte code queries

Code queries
Work with Brumleve’s “BO” attack

Known flaws prompt selection of vulnerability primitives

CERT

Sensitive Regions

Spoofable Methods

Data:

1031 Netscape class files

public class Socket {
  ...
  ...
  ...
  ...
  ...
  override non-final methods
  in a region handling security

iadd
iload 3
pop
sipush 334
getfield [cn. Fn:t]
aload_0
...
public class BOHTTP extends Applet {

    ...
    public void init () {
        ...
        ess = new BOServerSocket(port);
        ...
    }
    ...

    public void run () {
        BOSocket client;
        ...
        client = ess.accept.any();
        BOHTTPConnection ff = new BOHTTPConnection();
        ...
        (new Thread(ff)).start();
    }
    }
    ...
public class BOServerSocket extends ServerSocket {

    ...

    public BOSocket accept_any () throws IOException {
        BOSocket s = new BOSocket();
        try { implAccept(s); }
        catch (SecurityException se) { }
        return s;
    }

}

public class BOSocket extends Socket {

    public void close_real () throws IOException {
        super.close();
    }

    public void close () {
    }

}

Anatomy of the “BO” attack

Does Nothing!

Does Nothing!
protected final void implAccept (Socket socket) throws IOException
{
    try
    {
        socket.impl.address = new InetAddress();
        socket.impl.fd = new FileDescriptor();
        impl.accept(socket.impl);
        SecurityManager securitymanager = System.getSecurityManager();
        if (securitymanager != null)
            { securitymanager.checkAccept(socket.getInetAddress().getHostAddress(),
                socket.getPort());
                return; }
    }
    catch (SecurityException securityexception)   {
        socket.close();
        throw securityexception;
    }
    return; }

public void close () throws IOException
{
    impl.close
}
Anatomy of the “BO” attack

Class `BOURLConnection` extends `URLConnection` {
    ...
    public `BOURLConnection` (URL u) {
        super(u);
        connected = true;
    }
}

Class `BOURLInputStream` extends `URLInputStream` {
    ...
    public `BOURLInputStream` (URLConnection uc) throws IOException {
        super(uc);
        open();
    }
}
class BOHTTPDConnection implements Runnable {

    ...
    euc = new BOURLConnection(uu);
    euis = new BOURLInputStream(euc);
    while ((b = euis.read()) >= 0) os.write(b);
    ...
}

Files exposed across the net
Concepts lead to queries

Find all spoofable methods

Non-final methods that can be overridden

Compute their traces

Leverage from bytecode verifier

Find all sensitive regions

In particular, those handling security mechanisms

Look for invocations of spoofable methods that pass through sensitive regions
Code synthesis analysis

Semi-automated refinement

Specs for Application Domain

Analysis Tools

Library

Resource
Privilege
Protocol

Formal specifications

Target Code

public class Socket {

... 

...

}

Kestrel Institute
spec Spoofable_Invocation is
  op final? : method → Boolean
  op virtual? : invocation → Boolean
  op spoofable? : invocation → Boolean
  ...
end-spec

spec Sensitive_Region is
  sort Code_Region =
    {context : method,
     start : pc,
     end : pc,
     attributes : set CR_Attribute}
  sort CR_Attribute = | privileged
                      | ...
  ...
end-spec
Initial queries on Brumleve’s code

New entries for the semantic taxonomy

Queries:
- Where are sensitive regions R?
- Where are spoofable methods M invoked?
- What is intersection?

public class Socket {
    ...
    ...
    ...
    ...
    ...
    ...
    }
    
Known Flaw “rediscovered”
public class Socket {
    ...
    ...
    ...
    ...
    ...
    
    Known Flaw "rediscovered"

    Newly discovered Flaw (one of 5 new ones; exploitation assessment TBD)

Queries:
  • spoofable methods
  • sensitive regions
Finding more than expected

From java.net.DatagramSocket:

```java
public synchronized void receive (DatagramPacket datagrampacket)
    throws IOException
{
    SecurityManager securitymanager = System.getSecurityManager();
    synchronized(datagrampacket)
    {
        if (securitymanager != null) do
        {
            InetAddress inetaddress = new InetAddress();
            int I = impl.peek(inetaddress);
            try
            {
                securitymanager.checkConnect(inetaddress.getHostAddress(), I);
                break;
            }
            catch (SecurityException _ex)
            {
                DatagramPacket datagrampacket2 = new DatagramPacket (new byte[1], 1);
                impl.receive(datagrampacket2);
            }
        }
    }
    impl.receive(datagrampacket);
}
```
Current vision

Flaw cases

- Important app’s
- Common flaws
- ...

Flaws language

- Sensitive regions
- Spoofable methods
- ...

Semantic taxonomy

Automated tools

Specware Flawfinder™ WorkStation
Infrastructure

JVM type checker

Dataflow Engine

Transfer functions

JVM spoof checker

JVM structures

JCF structures

.isomorphi.class files
Several Enhancements

- Multiple entries for curried functions
- Extensive use of hash codes
- Canonical print routines
- Various algorithmic improvements
- Multiple refinements of maps, sequences, etc.
Many ways to implement maps

<table>
<thead>
<tr>
<th></th>
<th>update</th>
<th>access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Arrays</td>
<td>$O(N)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Trees</td>
<td>$O(\log N)$</td>
<td>$O(\log N)$</td>
</tr>
</tbody>
</table>
**Which Refinement?**

Assume N updates followed by N accesses:

<table>
<thead>
<tr>
<th>Refinement</th>
<th>Access Complexity</th>
<th>Update Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map → List</td>
<td>$O(N^{**2})$</td>
<td></td>
</tr>
<tr>
<td>Map → Array</td>
<td>$O(N^{**2})$</td>
<td></td>
</tr>
<tr>
<td>Map → Tree</td>
<td>$O(N \log N)$</td>
<td>access/update</td>
</tr>
</tbody>
</table>
Multiple Refinements!

Map \rightarrow \{ List, Array, Tree \}
Multiple Refinements!

- **List**
  - Updates: $O(N)$
  - Accesses: $O(N)$
  - $O(1)$

- **Array**
  - Updates: $O(N)$
  - Accesses: $O(N)$
  - $O(N)$

- **Tree**
  - Updates: $O(N)$
  - Accesses: $O(N)$
  - $O(logN)$
Description of Demonstration

- **Background:**
  - Show infrastructure for analyzing Java byte code

- **Ideas:**
  - *spoofable invocation* — virtual invocation of non-final method
  - *sensitive region* — try/catch/throw involving security, etc.
  - Intersection is a vulnerability

- **Demo:**
  - Write specs to instantiate these ideas
  - Generate code to find and report vulnerabilities
Start Demo!
- Semantically rich connections
  Arrows embed one theory into another
- Exploited in semi-automated ways
  Results for theories propagate
- Morphisms from one taxonomy node into a domain theory provide leverage for constructing the embedding of children or sibling nodes
Developing a useful taxonomy of vulnerabilities requires:

- Languages for describing flaws
- Theories to express properties of flaws
- Morphisms to relate those theories
- Power tools to exploit morphisms
Refinements (green arrows) are organized into a taxonomy.

Refinements are accessed and applied incrementally via a ladder construction.

- Classifying the structure of $S_0$:
  - $A \rightarrow S_0$
  - $B \rightarrow S_0^+$
  - $C \rightarrow S_0^{++}$
  - $D \rightarrow S_0^{+++}$
  - $E \rightarrow S_1$

- Applying the refinement $D \rightarrow E$.
Taxonomy of Collection Datatypes

PROTO-COLLECTION

PROTO-SEQ

PROTO-BAG

PROTO-SET

ORDERED-SEQ

SET-TUPLE

INDEXED-PARTITION

BIT-VECTOR

INFINITE-SEQ

BOUNDARY-SET

ARRAY

SEQ

BOUNDARY-SEQ

LIST

SEQ

BOUNDARY-SET
Taxonomy of Algorithm Theories

- **Constraint Satisfaction**
  - \( R = \text{set of maps} \)
- **Global Structure**
  - \( R = \text{set + recursive partition} \)
  - *generate-and-test*
  - *generate-and-test*
- **Local Structure**
  - \( R = \text{set + relation} \)
  - *genetic algorithms*
- **Local Structure**
  - \( R = \text{set + relation} \)
  - *hill climbing*
  - *tabu search*
- **Global Structure**
  - \( R = \text{set + recursive partition} \)
  - *binary search*
  - *backtrack*
- **Local Structure**
  - \( R = \text{set + relation} \)
  - *local search*
- **Local Structure**
  - \( R = \text{set + partial order} \)
- **Problem Reduction**
  - *sieves*
- **Problem Reduction**
  - *dynamic programming*
- **Problem Reduction**
  - *branch-and-bound*
- **Problem Reduction**
  - *game tree search*
- **Divide-and-Conquer**
  - *divide-and-conquer*
- **Network Flow**
  - \( R = \text{set of maps} \)
  - *specialized simplex method*
  - *Ford-Fulkerson*
- **Transportation**
  - *NW algorithm*
- **Assignment Problem**
  - *Hungarian method*
- **Local Poset Structure**
  - \( R = \text{set + partial order} \)
- **Local Semilattice Structure**
  - \( R = \text{semilattice} \)
- **GS-CSP**
  - \( R = \text{recursively partitioned set of maps} \)
- **GS-Horn-CSP**
  - \( R = \text{Horn-like Constraints} \)
  - *constraint propagation*
- **Monotone Deflationary Function**
  - *fixed point iteration*
Taxonomy of Resource Theories

Resource

(Start-time, Resource-type, Instantaneous demand, Precedes...)

Consumable
examples: fuel, crew time
constraint: cum. use ≤ cum. avail

Reusable
(Duration, Finish-time, max/min-capacity, ...)
examples: parking lots, ramp space, parallel processors, power
constraints: upper bound on capacity
finite usage intervals

Synchronously Reusable
(Separation)
examples: transportation, washing machine
constraints: synchronized blocks of reservations
min separation between blocks

Exact Capacity
example: wafer oven
constraint: lb = ub on capacity

Nonsharable
examples: berth, runway, crew
constraint: capacity = 1

Transportation Resource
examples: ship, aircraft, truck

(Origin, Destination, speed, Duration = distance/speed)
1. Library of Refinements

- Global Search
- Set
- Resource
- Global Search Algorithm
- Sequence
- Transportation Resource

2. Library of Refinement Generators

- Rewrite Simplification
- Context-dependent Simplification
- Finite Differencing
- Case Analysis
- Partial Evaluation

Scheduling

- Scheduling_0
- Scheduling_1
- Scheduling_2
- Scheduling_3
- Scheduling_4

Global Search

- Global Search Algorithm

Finite Differencing
Languages for Vulnerabilities

- Ontology:
  - Resource, Agent, Action, Manager, …
  - Privilege, Authorization, Friend, Enemy, …
  - Message, Channel, Send, Receive, Request, …
  - File, Owner, Read, Write, Modify, …
  - Process, Thread, Exception, Interrupt, …

- Modal, Meta, or Higher-Order Concepts
  - Time, Knowledge, Necessity, Desirability, …
  - Race, Deadlock, Cost, …

- Objectives
  - Security, Reliability, Availability, Efficiency, …
Typical Expressions

- Requests(x, y, action)  trusts(y, x)
  Executes(y, action)

- Receives(x, msg)
  Believes(x, sent(author(msg), msg))
Theory of a Flaw

- Receives(x,request)  Validates(x,request) Executes(x,request)

- Send(x,y,request)   author(request) = x

- Validates(x,request) ⇔
  Friend(author(request),x)  ¬ Dangerous(request)

- ¬ Dangerous(send(x,y,z))

- Send(Intruder, Dupe, ‘Send(Dupe,Victim,bomb)’)
Morphisms

- **Resource** => Space, Processor, Data, ...

- **File** => Unix-File, NT-File, ...

- **Privilege** => Read, Write, Execute, ...

- **Read** => fread, mmap, ...
Semantic Taxonomy of Flaws

Boxes are theories.
Arrows are semantic!

- FLAW
  - LIE
  - SPOOFING
    - FORGERY
      - ALTERED MSG HDR
    - TROJAN HORSE
      - B.O. EXAMPLE
Current vision

- Important app’s
- Common flaws
- ...

Flaw cases

- Sensitive regions
- Spoofable methods
- ...

Flaws language

Semantic taxonomy

Automated tools

- Q1
- Q2

Specware Flawfinder™ WorkStation
Questions
Code generation is accomplished via a logic morphism from SPEC to the logic of a programming language.
Planware Refinements

Resource

Transportation Resource

Semilattice Attribute of Task

Definite Constraint

Abstract Scheduling

Transportation Scheduling 0

TS 1

TS2

TS3

user chooses from the Resource Taxonomy

user edits the Spreadsheet

the rest is automatic!

Set(\(A \times B \times C\))

Indexed-Partition map\((A, Set(\(A \times B \times C\)))\)
DRO

Global Search with CP

Global Search program

TS4

TS5

TS6

TS7

TS8

TS9

Definite Constraints

Constraint Propagation algorithm

Sort + n-attributes

n-tuple

CommonLisp code

algorithm design and program optimizations

Constraint Propagation algorithm

Context-Dependent Simplification

n-attributes

CommonLisp code