CS 6V81-05
Vulnerability Analysis II: Crash dump analysis using BitBlaze

Kevin Ross Weaver
Department of Computer Science
The University of Texas at Dallas

September 9th, 2011
Outline

1. Introduction
   - What is BitBlaze?

2. The BitBlaze Platform
Outline

1 Introduction
   - What is BitBlaze?

2 The BitBlaze Platform
1. Introduction
   - What is BitBlaze?

2. The BitBlaze Platform
What is BitBlaze?

**BitBlaze** is a binary analysis tool used to aid in reverse engineering by analyzing binary code.
What is BitBlaze?

**BitBlaze** is a binary analysis tool used to aid in reverse engineering by analyzing binary code.

BitBlaze is primarily intended for (or at least has been primarily used for) analyzing crash data to find vulnerabilities.
What is BitBlaze?

BitBlaze is a binary analysis tool used to aid in reverse engineering by analyzing binary code.

BitBlaze is primarily intended for (or at least has been primarily used for) analyzing crash data to find vulnerabilities.

Basically, it’s used to find out why an application crashed to see if the application can be fixed.
What’s the problem?

The difficulty/laboriousness of sifting through the binary data in crash dumps.
The difficulty/laboriousness of sifting through the binary data in crash dumps.

Searching through crashes is a time-consuming task that requires lots of people all looking for the root cause of an issue. The number of crashes a single program can generate after only a little bit of, say, fuzzing is staggering.
In 2009, Ben Nagy gave a talk called "Finding Microsoft Vulnerabilities by Fuzzing Binary Files with Ruby - A New Fuzzing Framework".

In the talk he claimed to have found 200,000 crashes by fuzzing Microsoft Word.
In 2009, Ben Nagy gave a talk called "Finding Microsoft Vulnerabilities by Fuzzing Binary Files with Ruby - A New Fuzzing Framework".

In the talk he claimed to have found 200,000 crashes by fuzzing Microsoft Word.

These 200,000 could be divided into 61 unique types (or bins)[1], of which 4 were serious security vulnerabilities.

He was only able to find out which crashes represented vulnerabilities by submitting the crashes to Microsoft so that they could help him analyze them all.
Who Would Use it and Why?

Security Researchers
- Have to find the important crashes

Developers
- Trying to find vulnerabilities in order to fix them

Security Officials
- Searching for hidden code or malware (since so few malware authors release their source code)
Who Would Use it and Why?

- **Security Researchers** - Have to find the important crashes
Who Would Use it and Why?

- **Security Researchers** - Have to find the important crashes
- **Developers** - Trying to find vulnerabilities in order to fix them
Who Would Use it and Why?

- **Security Researchers** - Have to find the important crashes
- **Developers** - Trying to find vulnerabilities in order to fix them
- **Security Officials** - Searching for hidden code or malware (since so few malware authors release their source code)
Who Would Use it and Why?

- **Security Researchers** - Have to find the important crashes
- **Developers** - Trying to find vulnerabilities in order to fix them
- **Security Officials** - Searching for hidden code or malware (since so few malware authors release their source code)
1 Introduction
   - What is BitBlaze?

2 The BitBlaze Platform
Basic Structure

BitBlaze has three major sets of components to handle:

1. **Static Analysis** - VINE
BitBlaze has three major sets of components to handle:

1. **Static Analysis** - VINE
2. **Dynamic Analysis** - TEMU
BitBlaze has three major sets of components to handle:

1. **Static Analysis** - VINE
2. **Dynamy Analysis** - TEMU
3. **Symbolic Exploration** - Rudder/BitFuzz/FuzzBALL
VINE

**Front End**
- Instruction Lifting
- Binary Format Interface

**Intermediate Language**

**Back End**
- CFG, PDG
- SSA conversion
- Dead Code Elimination
- Value Set Analysis
- Solver API
Taint Propagation and Spreading

Application taints data (flags), and within TEMU we can trace how the tainted data is propagated through the system.
Tools and Utilities - 1

Emulated System

TEMU API

Tracecap plugin
- Allocation tracing hooks

Execution trace file

Process state file

Allocation trace file

To Vine

tracealign

x86_slicer

trace_reader

tracedump

alloc_reader

Taint Analysis Engine

Semantics Extractor
trace reader: Displays information about a trace
trace reader: Displays information about a trace
x86 slicer: Displays how data propagated through the program
trace reader: Displays information about a trace
x86 slicer: Displays how data propagated through the program
align: Highlights the differences between two traces, usually one good and one bad
Tools and Utilities - 2

- trace reader: Displays information about a trace
- x86 slicer: Displays how data propagated through the program
- align: Highlights the differences between two traces, usually one good and one bad
- tracedump: Displays taint info from around registers acting as pointers
Tools and Utilities - 2

- trace reader: Displays information about a trace
- x86 slicer: Displays how data propagated through the program
- align: Highlights the differences between two traces, usually one good and one bad
- tracedump: Displays taint info from around registers acting as pointers
- alloc reader: Displays which allocation corresponds to a given address
Resources