GETTING IT RIGHT: SYSTEM TESTING OF SCIENTIFIC SOFTWARE

Myra Cohen

https://www.cs.iastate.edu/~mcohen
mcohen@iastate.edu
Scientific Software is Everywhere

EXAFEL AND COPA: RAPID IMAGING OF MOLECULAR SYSTEMS
Using CuPA tools, ExaFEL prevents computational data throughput from bottlenecking experimental progress in x-ray free electron laser facilities.
Source: ECP

EXAALT AND KOKKOS: MAKING EXASCALE SIMULATIONS OF MATERIAL BEHAVIOR A "SNAP"
Molecular dynamics has become a cornerstone of computational science and is a key component of developing materials with enhanced properties.
Source: ECP

EXASCALE COMPUTING PROJECT SOFTWARE HELPS LAUNCH A NEW ERA FOR NASA
ECP Software Helps Launch a New Era for NASA
Source: ECP

EXASCALE COMPUTING PROJECT CONtributes to ACCELERATING CANcer RESEARCH
The Exascale Computing Project’s CANDLE application will improve cancer research techniques and clinical outcomes
Source: ECP
Yet it sometimes Fails..

Error: NCBI C++ Exception:
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/objjstrasnb.cpp"
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/member.cpp", l:
Yet it sometimes Fails..

Error: NCBI C++ Exception:
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/objjstrasnb.cpp"
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/member.cpp", l:

ial/objjstrasnb.cpp", line 499: Error: (CSerialException::eOverflow) byte 132: overf
ial/member.cpp", line 767: Error: (CSerialException::eOverflow) ncbi::CMemberInfoFur
Yet it sometimes Fails..

NCBI blastp bug - changing max_target_seqs returns incorrect top hits

NCBI blastp seems to have a bug where it reports different top hits when -max_target_seqs is changed. This is a serious problem because the first 20 hits (for example) should be the same whether -max_target_seqs 100 or -max_target_seqs 500 is used.

The bug is reproducible on the command line when searching NCBI's nr blast database (dated 25-Nov-2015) using NCBI 2.2.28+, 2.2.30+ and 2.2.31+.
Yet it sometimes Fails..

Sequence analysis

**Misunderstood parameter of NCBI BLAST impacts the correctness of bioinformatics workflows**

Nidhi Shah¹, Michael G. Nute², Tandy Warnow ³ and Mihai Pop ¹,∗

¹Department of Computer Science, University of Maryland College Park, MD 20742, USA, ²Department of Statistics and ³Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61820, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Contact: mpop@umd.edu

Received and revised on August 13, 2018; editorial decision on September 19, 2018; accepted on September 21, 2018
Sometimes it seems to fail..

Sequence analysis

Reply to the paper: Misunderstood parameters of NCBI BLAST impacts the correctness of bioinformatics workflows

Thomas L. Madden*, Ben Busby and Jian Ye

National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA

*To whom correspondence should be addressed.
Associate Editor: John Hancock
Contact: madden@ncbi.nlm.nih.gov

Received and revised on November 30, 2018; editorial decision on December 10, 2018; accepted on December 19, 2018
Sometimes it seems to fail..

Dear Editor,

A recent letter by Shah et al. (2018) addressed the use of a command-line parameter in BLAST (Altschul et al., 1997; Camacho et al., 2009). BLAST is a very popular tool, so it is not surprising that this topic has provoked a great deal of interest. The authors have, however, conflated three different issues. One is a bug that will be fixed in the BLAST+ 2.8.1 release due out in December 2018, another is simply how BLAST works and the third might be viewed as a shortcoming of our implementation of composition-based statistics (CBS). Here, we address these issues and describe some new documentation about the BLAST process.

Shah et al. (2018) did not provide their own example in the letter, but later provided one at https://github.com/shahnidhi/BLAST_maxtargetseq_analysis. At the NCBI, we examined the new example and it became clear that the demonstrated behavior was a bug, resulting from an overly aggressive optimization, introduced in 2012 for BLASTN and MegaBLAST (DNA–DNA alignments). This bug has been fixed in the BLAST+ 2.8.1 release, due out in December 2018. The aberrant behavior seems to occur only in alignments with an extremely large number of gaps, which is the case in the example provided by Shah and collaborators.
Cost of Poor Quality Software

- CISQ Consortium for Information & Software Quality 2020 report
  $2.08 trillion cost in United States
- 2002 National Institutes of Standards and Technology
  Up to $59 Billion per year in United States
- Scientific Software
  ?
Why Test Software?

**contributed articles**

**Dynamic analysis techniques help programmers find the root cause of bugs in large-scale parallel applications.**

By Ignacio Laguna, Dong H. Ahn, Bronis R. De Supinski, Todd Gamblin, Gregory L. Lee, Martin Schulz, Saurabh Bagchi, Milind Kulkarni, Bowen Zhou, Zhezhe Chen, and Feng Qin

**Debugging High-Performance Computing Applications at Massive Scales**
Why Test Software?

contributed articles

An approach to reproducibility problems related to porting software across machines and compilers.

BY DONG H. AHN, ALLISON H. BAKER, MICHAEL BENTLEY, JAN BRIGGS, GANESH GOPALAKRISHNAN, DORIT M. HAMMERLING, IGNACIO LAGUNA, GREGORY L. LEE, DANIEL J. MILROY, AND MARIANA VERTENSTEIN

Keeping Science on Keel When Software Moves

the machine instructions that actually get executed. Unfortunately, such changes do affect the computed results to a significant (and often worrisome) extent. In a majority of cases, there are not easily definable a priori answers one can check against. A programmer ends up comparing the new answer against a trusted baseline previously established or checks for indirect confirmations such as whether physical properties such as energy are conserved. However, such non-systematic efforts might miss underlying issues, and the code may keep misbehaving until these are fixed.

In this article, we present real-world evidence to show that ignoring numerical result changes can lead to misleading scientific conclusions. We present techniques and tools that can help computational scientists understand and analyze compiler effects on their scientific code. These techniques are applicable across a wide range of examples to narrow down the root causes to single files, functions within files, and even computational expressions that affect specific variables.

The developer may then rewrite the code selectively and/or suppress the application of certain optimizations to regain more predictable behavior.

Going forward, the frequency of required ports of computational software will increase, given that performance gains can no longer be obtained by mere-
Why Test Software?

Overview

- Types of Testing
- Challenges
- Models
- Coverage
- Oracles
- Configurability
What is Testing

Input/data → Computer → Result

Oracle
What Should We Test?

Correctness  Performance  Security  Interoperability  Usability  Other...
What Should We Test?

Unit tests are important, but there is more....
Limitations

Testing can only show the presence of faults. It cannot determine their absence.

*Edsger W. Dijkstra*
Challenge 1

- To detect a program FAILURE we need to:
  - Reach a FAULT in the code
  - Infect the code (change to incorrect state) - ERROR
  - Propagate the error out of program
  - Reveal (detect) the error - (ORACLE)

RIPR model Ammann, Offutt (Introduction to Software Testing, 2016)
Challenge 2

- **Covering code** during testing, only tests the logic that is there!
  - *Also need to test from the system* – does the software meet the specifications
Tests Can Miss Faults

```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp  # fault should be a=b
        b = tmp
    if a > c:
        tmp = a
        a = c
        c = tmp
    if b > c:
        tmp = b
        b = c
        c = tmp
    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```
Tests Can Miss Faults

1. Test Case 3, 4, 5 (scalene)
   - doesn’t reach fault

```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp  # fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```
Tests Can Miss Faults

1. Test Case 3, 4, 5 (scalene)
   • doesn’t reach fault

2. Test Case 5, 1, 1 (invalid)
   • reaches fault and infects
   • reveals (returns isosceles)
Tests Can Miss Faults

1. Test Case 3, 4, 5 (scalene)
   - doesn’t reach fault

2. Test Case 5, 1, 1 (invalid)
   - reaches fault and infects
   - reveals (returns isosceles)

3. Test Case 2, 1, -1 (invalid)
   - reaches fault and infects
   - Doesn’t propagate (2, 2,-1) is still INVALID
<table>
<thead>
<tr>
<th>ID</th>
<th>myracohen:narrative_1501886992034/kbase-all-hands-on-different-media</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object type</td>
<td>KBaseFBA.FBA-13.2</td>
</tr>
<tr>
<td>Owner</td>
<td>myracohen</td>
</tr>
<tr>
<td>Version</td>
<td>1</td>
</tr>
<tr>
<td>Mod-date</td>
<td>2024-02-26T06:15:44+00:00</td>
</tr>
<tr>
<td>Objective value</td>
<td>0.507834</td>
</tr>
<tr>
<td>Model</td>
<td>E-Coli_k12_MG1</td>
</tr>
<tr>
<td>Media</td>
<td>Carbon-D-Glucose</td>
</tr>
<tr>
<td>Single KO</td>
<td>0</td>
</tr>
<tr>
<td>Number reactions</td>
<td>857</td>
</tr>
<tr>
<td>Number compounds</td>
<td>29</td>
</tr>
<tr>
<td>Gene KO</td>
<td>0</td>
</tr>
<tr>
<td>Reaction KO</td>
<td>0</td>
</tr>
<tr>
<td>Custom bounds</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>myracohen:narrative_1501886992034/kbase-all-hands-on-different-media</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object type</td>
<td>KBaseFBA.FBA-13.2</td>
</tr>
<tr>
<td>Owner</td>
<td>myracohen</td>
</tr>
<tr>
<td>Version</td>
<td>2</td>
</tr>
<tr>
<td>Mod-date</td>
<td>2024-02-26T06:36:45+0000</td>
</tr>
<tr>
<td>Objective value</td>
<td>13.4555</td>
</tr>
<tr>
<td>Model</td>
<td>E-Coli_k12_MG1</td>
</tr>
<tr>
<td>Media</td>
<td>Complete</td>
</tr>
<tr>
<td>Single KO</td>
<td>0</td>
</tr>
<tr>
<td>Number reactions</td>
<td>857</td>
</tr>
<tr>
<td>Number compounds</td>
<td>29</td>
</tr>
<tr>
<td>Gene KO</td>
<td>0</td>
</tr>
<tr>
<td>Reaction KO</td>
<td>0</td>
</tr>
<tr>
<td>Custom bounds</td>
<td>0</td>
</tr>
</tbody>
</table>
The Future?

, or alternatively, Machine learning (ML) classifiers will select the templates for model reconstruction

Build and gap-fill genome-scale metabolic models with ModelSEED v2 (MS2)

ModelSEED 2 genome-scale metabolic reconstruction pipeline enabling quantitative prediction of ATP production

A genome annotated with RAST is provided as input. Users can select a template for reconstruction, or alternatively, Machine learning (ML) classifiers will select the templates for model reconstruction. ATP production is tested across 54 media formulations representing many diverse energy biosynthesis strategies, with gap-filling performed as necessary to ensure ATP is produced in at least one condition. Next, we expand the core metabolism model to the genome-scale. The genome-scale model’s gap-filling is then performed using our default auxotrophic medium or an optional user-specified custom media formulation.
Overview

Types of Testing  Challenges  Models  Coverage  Oracles  Configurability
Models

- Provide an *abstraction* of the software we are testing
- Can be *for different dimensions* of the software (specifications, interface, code)
- Allow us to reason about *how much* we have tested
- The foundation for *automated test generation*
Example Models

- Graphs
- Tabular
- Relational
- Grammar based
- Logic based
Graph Models

- Program control flow graph
- User interface
- Program state machine
Types of Graph Coverage

- All nodes
- All edges (pairs of nodes)
- All length $N$ paths
- $M$ random length $N$ paths
Program Code Coverage

```java
if (x < y)
{
    y = 0;
    x = x + 1;
} else
{
    x = y;
}
```

Control flow graph
Program Code Coverage

Cog coverage: 38.75%

Example tools: jacoco, coverage.py, gcov
Interface (graph) Coverage

GUI

Web
Other Coverage

Specification coverage

• Cover the system requirements

Interaction coverage

• Measure interactions between components
  • Pairs, n-way coverage
Overview

Types of Testing
Challenges
Models
Coverage
Oracles
Configurability
What is the Correct Answer?
Trivial Oracles

- Program crashes
- Core dump
- Segmentation error
- Overflow
- Program hangs
Trivial Oracles

- Good when we don’t have a known result
- **Weakest oracle** since it only shows that the program fails/not that the result is incorrect
- Exact oracles are easy to compute in some programs
Harder Oracles
Making Oracles Hard

- Results may differ by small epsilons (due to rounding)
- Expected result may not be computable without program
- May have time series results
- Takes a long time to manually compute each oracle (even when we can)
- Programs may be stochastic (or non-deterministic)
Examples

Python docs

Note: The behavior of `round()` for floats can be surprising: for example, `round(2.675, 2)` gives 2.67 instead of the expected 2.68. This is not a bug: it's a result of the fact that most decimal fractions can't be represented exactly as a float. See Floating Point Arithmetic: Issues and Limitations for more information.

Same growth values?

Expected: 0.35695124
Observed: 0.35695122

Correct hits?

<table>
<thead>
<tr>
<th>Descriptions</th>
<th>Genetic Summary</th>
<th>Alignments</th>
<th>Taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select all 100 sequences selected</td>
<td>GenBank</td>
<td>Graphics</td>
<td>Distance tree of results</td>
</tr>
<tr>
<td>Saccharomyces pastorianus strain CBS 1483 chromosome SoXII</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saccharomyces cerevisiae strain CEN.PK113-7D chromosome XII</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saccharomyces cerevisiae strain YDR128 chromosome XI, complete sequence</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saccharomyces cerevisiae strain YMR269 chromosome 45</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

41
Some Techniques

Differential testing  Metamorphic testing
Differential Testing

- Run same tests using different programs that have the same functionality
  - run tests with BLAST
  - run tests with HPC-BLAST

Differential testing

Challenge is determining equivalency
Metamorphic Testing

Define relations on sets of tests:

e.g. (subtraction)

(1) \( A - B = C \)

Create \( A' \) (greater than \( A \))

(2) \( A' - B = C' \) means \( C' \) is greater than \( C \)
Metamorphic Testing

Use Domain Knowledge
  e.g. Ocean temperature modeling
A. Compute predicted temperature
B. Modify to increase expected temperature

Confirm relation holds
Carbon-D-Glucose <= Complete
Overview

Types of Testing  Challenges  Models  Coverage  Oracles  Configurability
Configurability
gactacgatcgggc
Configurability

- Open new windows in a new tab instead
- Warn me when closing multiple tabs
- Warn me when opening multiple tabs might slow down Firefox
- Don’t load tabs until selected
- When I open a link in a new tab, switch to it immediately
MEGAHIT (DNA Assembler)

- DNA sequenced into small segments (reads)
- Assembly combines reads into longer continuous sequences
- Result is a certain number of continuous sequences

Credit to Mikaela Cashman

Assemble Reads with MEGAHIT v1.11
Assemble metagenomic reads using the MEGAHIT assembler.

Input Objects
Read Library
rhodo.art.q20.PE.reads

Parameters (5 advanced parameters showing) hide advanced
Parameter preset

--min-count
--k-min
1 ≤ ≤127
--k-max
1 ≤ ≤255
--k-step
1 ≤ ≤28

--k-list

--min-contig-len
300 ≤ 2000

Output Objects
Output Assembly name
Number of continuous sequences: 284
Configuration
Default

Job Status
56  [utils.h : 126] Real: 0.4913 user: 0.3962 sys: 0.0720 maxrss: 24688
7 --- [Sat Mar 24 04:35:57 2018] k-max reset to: 119 ---
59 --- [Sat Mar 24 04:35:57 2018] Extracting solid (k+1)-mers for k = 21 ---

Output Objects
Number of continuous sequences 284

Run
Configure
Job Status
Result
Assemble Reads with MEGAHIT v1.11
Assemble metagenomic reads using the MEGAHIT assembler.

Input Objects
Read Library: rhodo.art.q20.PE.reads

Parameters (5 advanced parameters showing)
Parameter preset: 

--min-count
--k-min
1 ≤ k ≤ 127
--k-max
1 ≤ k ≤ 255
--k-step
1 ≤ k ≤ 28

--k-list

--min-contig-len
300 ≤ contig-len ≤ 2000

Output Objects
Number of continuous sequences: 284

Configuration
k-max=119

Result
### Assemble Reads with MEGAHIT v1.1.1

**Assemble metagenomic reads using the MEGAHIT assembler.**

**Input Objects**

- **Read Library**: rhodo.art.q20.PE.reads

**Parameters (5 advanced parameters showing)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--min-count</td>
<td>2</td>
</tr>
<tr>
<td>--k-min</td>
<td>21</td>
</tr>
<tr>
<td>--k-max</td>
<td>99</td>
</tr>
<tr>
<td>--k-step</td>
<td>12</td>
</tr>
</tbody>
</table>

**Output Objects**

<table>
<thead>
<tr>
<th>Number of continuous sequences</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>289</td>
</tr>
</tbody>
</table>

**Configuration**

- Default
- k-max=119
- Default-Manual
- k-max=99
### Input Objects

**Read Library**
- `rhodo.art.q20.PE.reads`

### Parameters (5 advanced parameters showing)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>--min-count</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>--k-min</td>
<td>1≤</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>--k-max</td>
<td>1≤</td>
<td>99</td>
<td>≤127</td>
</tr>
<tr>
<td>--k-min</td>
<td>1≤</td>
<td>119</td>
<td>141</td>
</tr>
<tr>
<td>--k-max</td>
<td>1≤</td>
<td>≤255</td>
<td></td>
</tr>
<tr>
<td>--k-min</td>
<td>1≤</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>--k-max</td>
<td></td>
<td>≤28</td>
<td></td>
</tr>
</tbody>
</table>

### Output Objects

**Number of continuous sequences**
- 284
- 285
- 285
- 289
## Challenge for Testing

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Cache Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Media Player

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Cache Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Media Player

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Cache Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Media Player

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Cache Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Testing the Player

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Cache Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Test Case:** Open video and play to completion
# Interaction Fault

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Cache Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Test Case:** Open video and play to completion
Configuration-Dependent Security Bugs (CERT)

Overview
Apache Struts 2 framework, versions 2.5 to 2.5.12, with REST plugin insecurely deserializes untrusted XML data. A remote, unauthenticated attacker can leverage this vulnerability to execute arbitrary code in the context of the Struts application.

Description
CWE-692: Deserialization of Untrusted Data - CVE-2017-9805
In Apache Struts 2 framework, versions 2.5 to 2.5.12, the REST plugin uses XStreamHandler with an instance of XStream to deserialize XML data. Because there is no type filtering, a remote, unauthenticated attacker may send a specially crafted XML payload to execute arbitrary code in the context of the Struts application.

Refer to the researcher’s blog post for more information about this vulnerability. A Metasploit module with exploit code is publicly available.

Remove or limit the REST plugin
If it is not used, consider removing the REST plugin. Per the vendor, it is also possible to limit its functionality to normal server pages or JSON with the following configuration change in struts.xml:

```xml
<constant name="struts.action.extension" value="xhtml,,json"/>
```
Also Impacts Program Performance

Same functionality can be achieved in different performance
Configuration Dependence

Figure credited to Xiao Qu
# Real Configuration Spaces

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Caching Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Real Configuration Spaces

- $3 \times 3 \times 3 \times 2 \times 2 = 108$ configurations
- 10 features with 5 options = 9,765,625 configs
- 4 hours to run test suite $\rightarrow$ 4,459 years to run
- gcc optimizer (199 options) $\rightarrow$ $10^{61}$
- Linux > 10,000 features
Combinatorial Interaction Testing (CIT)

- Sample the space so that all \( t \)-way combinations of values occur AT LEAST once

- \( t \) is defined as strength of testing
## Combinatorial Interaction Testing (CIT)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Caching Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MPEG</td>
<td>Stream</td>
<td>Medium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2 RAW</td>
<td>Video</td>
<td>High</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3 MPEG</td>
<td>Video</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>4 WAV</td>
<td>Stream</td>
<td>High</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>5 RAW</td>
<td>Stream</td>
<td>Low</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>6 MPEG</td>
<td>Audio</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>7 WAV</td>
<td>Video</td>
<td>Medium</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>8 RAW</td>
<td>Audio</td>
<td>Medium</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>9 WAV</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Combinatorial Interaction Testing (CIT)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Caching Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MPEG</td>
<td>Stream</td>
<td>Medium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2 RAW</td>
<td>Video</td>
<td>High</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3 MPEG</td>
<td>Video</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>4 WAV</td>
<td>Stream</td>
<td>High</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>5 RAW</td>
<td>Stream</td>
<td>Low</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>6 MPEG</td>
<td>Audio</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>7 WAV</td>
<td>Video</td>
<td>Medium</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>8 RAW</td>
<td>Audio</td>
<td>Medium</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>9 WAV</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Combinatorial Interaction Testing (CIT)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Caching Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td>Stream</td>
<td>Medium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Video</td>
<td>High</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MPEG</td>
<td>Video</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>WAV</td>
<td>Stream</td>
<td>High</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RAW</td>
<td>Stream</td>
<td>Low</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MPEG</td>
<td>Audio</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WAV</td>
<td>Video</td>
<td>Medium</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RAW</td>
<td>Audio</td>
<td>Medium</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>WAV</td>
<td>Audio</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Combinatorial Interaction Testing (CIT)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Format</th>
<th>Caching Level</th>
<th>Closed-Captioning</th>
<th>Network Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPEG Stream Medium</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RAW Video High</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MPEG Video Low</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>WAV Stream High</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RAW Stream Low</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MPEG Audio High</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>WAV Video Medium</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RAW Audio Medium</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>WAV Audio Low</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Some Combinatorial Testing Tools

ACTS

National Institutes of Standards

Combinatorial Methods for Trust and Assurance

Overview

Combinatorial methods reduce costs for testing, and have important applications in software engineering:

- Combinatorial or t-way testing is a proven method for better testing at lower cost. The key insight underlying its effectiveness resulted from a series of studies by NIST from 1999 to 2004. NIST research showed that most software bugs and failures are caused by one or two parameters, with progressively fewer by three or more, which means that combinatorial testing can provide more efficient fault detection than conventional methods. Multiple studies have shown fault detection equal to...
Characterizing Test Failures

- Use machine learning techniques (classification trees) to model option setting patterns that explain test failures.
Summary

Overview

Types of Testing

Challenges

Models

Coverage

Oracles

Configurability

Some Techniques

Differential testing

Metamorphic testing

Configurability

This work was supported in part by the Better Scientific Software Fellowship Program, funded by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy (DOE) Office of Science and the National Nuclear Security Administration; and by the National Science Foundation (NSF) under Grant Nos. 2154495, 2234908, 1909688.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DOE or NSF.