GRAMMATECH

BUG-INJECTOR: Introduction and Usage

Eric Schulte
eschulte@grammatech.com
September 19, 2019

Agenda

- Bug-Injector
  - ▷ Objectives
  - ▷ Operation
  - ▷ Language Support
  - ▷ Bug Template Libraries
- Usage
  - ▷ Methodology
  - ▷ Experience
  - ▷ Results
  - ▷ Lost Bugs
  - ▷ Use Cases

Bug-Injector

Objectives and Criteria for Success

Objective

Inject realistic vulnerabilities into large code bases to support:

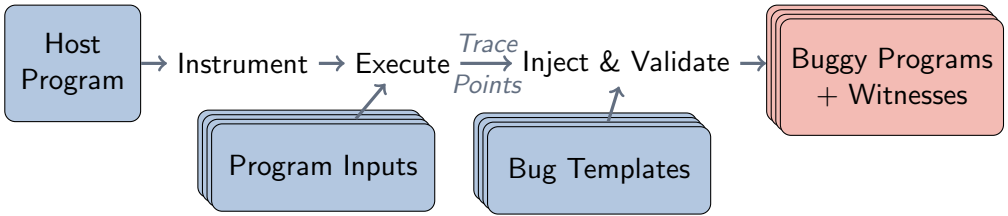
- testing and evaluating cyber security tools and capabilities,
- and to enable novel pedagogical tools

Success

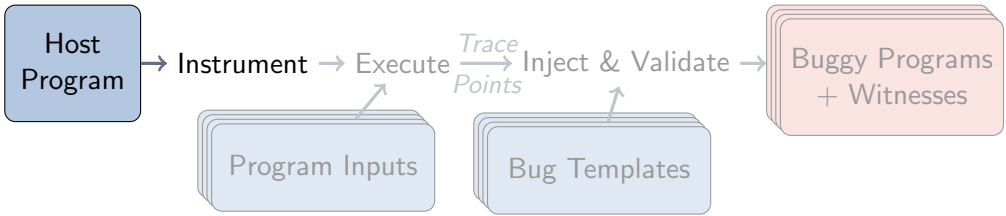
Realistic vulnerabilities in real-world code.

- Integrated
- Realistic
- Demonstrable (witness)
- Avoid systematic bias
- Breadth of languages
- Breadth of bug classes
- Customizable
- Automatic, easy to use

Injection pipeline



Injection pipeline

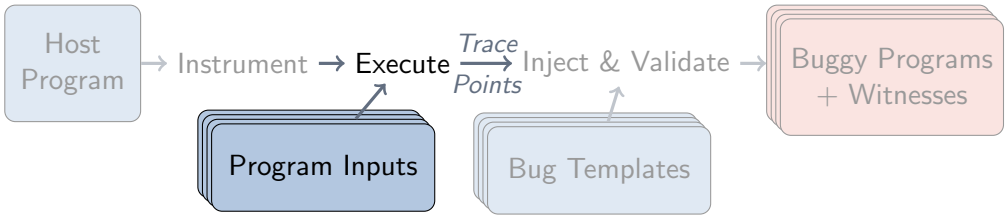



```
while (b != 0) {
  if (a > b) {
    a = a - b;
  } else {
    b = b - a;
  }
}
printf("%g\n", a);
```

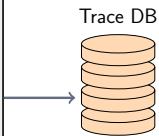
ASTs

```
while (b != 0) {
  fputs("((:C . 54) ", __bi_mut_log_file);
  fprintf(__bi_mut_log_file,
          "(:UNBOUND-VALS (\"a\" \"int\" %i) "
          "(\"b\" \"int\" %i))", a, b);
  fprintf(
      __bi_mut_log_file,
      "(:SCOPES (\"c\" \"int\" %i) (\"b\" \"int\" %i) "
      "(\"a\" \"int\" %i))",
      c, b, a);
  fputs(")\n", __bi_mut_log_file);
  if (a > b) {
    fputs("((:C . 61) ", __bi_mut_log_file);
    //...
```

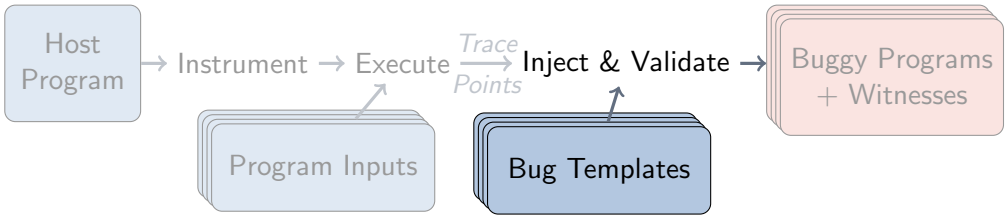
Injection pipeline

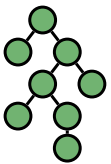



```
while (b != 0) {
  fputs("((:C . 54) ", __bi_mut_log_file);
  fprintf(__bi_mut_log_file,
          "(:UNBOUND-VALS (\"a\" \"int\" %i) "
          "(\"b\" \"int\" %i))", a, b);
  fprintf(
      __bi_mut_log_file,
      "(:SCOPES (\"c\" \"int\" %i) (\"b\" \"int\" %i) "
      "(\"a\" \"int\" %i))",
      c, b, a);
  fputs(")\n", __bi_mut_log_file);
  if (a > b) {
    fputs("((:C . 61) ", __bi_mut_log_file);
    //...
```



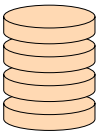
Injection pipeline



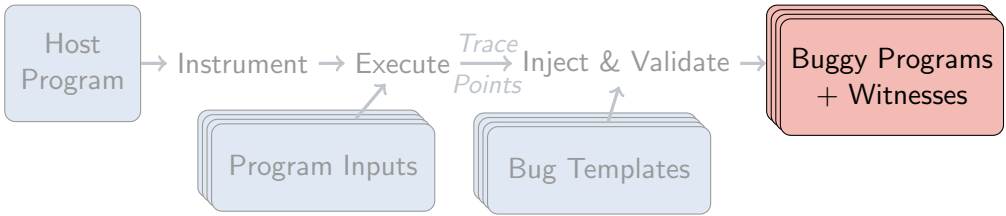


```
(define-scion arithmetic-overflow-multiplication-scion
  (make-instance 'clang-scion
    :name 'arithmetic-overflow-multiplication-scion
    :patches
    (list (make-instance 'clang-dynamic-patch
            :cwe 190
            :cwe-line 1
            :preconditions '(> (* (v/value x) (v/value y))
                               +max-int-32+)
            :free-vars '(("x" "int") ("y" "int"))
            :code "x = x * y;"))))
```

Trace DB



Injection pipeline




```
--- gcd/gcd-format.c
+++ gcd/gcd-ARITHMETIC-OVERFLOW-MULTIPLICATION-SCION-000001.c
@@ -1,23 +1,26 @@
 int main(int argc, char *argv[]) {
   int a, b, c;
   a = atoi(argv[1]);
   b = atoi(argv[2]);
+  /* from input ./gcd 1073741824 2 */
+  b = b * a;
   while (b != 0) {
     if (a > b) {
       a = a - b;
     } else {
```

Languages

Table: Language Support

Language	Front-End	Status
C	Clang[1]	Mature
C++	Clang	Mature
JavaScript	ESTree[2]	Immature
Java	JavaParse[3]	Partial

[1] https://clang.llvm.org/
[2] https://github.com/acornjs/acorn
[3] http://javaparser.org/

Bug Templates

Currently mature bug templates only exist for C languages

Original Hand crafted templates for integer-overflow, divide by zero, and buffer-overflow.

CWE Coverage Templates extracted from Juliet covering 55 distinct CWEs.

Validated Roughly 200 templates extracted from Juliet which have been validated using dynamic tools; Valgrind, ASAN, UBSAN.

Malware Roughly a dozen hand-crafted malware and insider-threat bug templates. An external customer is also writing malware templates.

Usage

Automated Customized Bug-Benchmark Generation

To appear in SCAM 2019 https://arxiv.org/abs/1901.02819

Benchmark Creation

1. **Select tools:** Infer and Clang Static Analyzer (CSA).
2. **Select bug classes:** Buffer Overflow (BO) and Null Pointer Dereference (NPD).
3. **Collect bug templates:** from Infer/CSA tests and Infer/CSA docs.
4. **Select hosts:** Nginx and grep.
5. **Inject:** bug templates into hosts with Bug-Injector.

Tool Evaluation

1. Select checkers:
   - CSA Defaults plus: `alpha`, `security`, `osx`, `llvm`, `nullability`, and `optin`
   - Infer Defaults plus: `compute-analytics`, `biabduction`, `quandary`, and `bufferoverrun`
2. Run tools against bug benchmark.
3. Manually investigate missed bugs.

Experience Using Bug-Injector

Writing Bug Templates

Takes a couple of minute to convert an example into a template.

Running Bug-Injector

- Execution can be slow on large programs with large test suites.
- Bug-Injector has options to tune instrumentation and trace-collection.

Iterating

- Manual review of injections can suggest changes to options or bug templates.
- Room for improvement on Bug-Injector:
  - Automation to prioritize diversity in injection locations.
  - Automatic detection and avoidance of uninitialized variables.
  - Additional information in traces (e.g., dynamic taint).
  - More efficient trace collection and search.

Generated Benchmark

https://doi.org/10.5281/zenodo.3341585

Benchmark

Table: Bug Templates

Bug Template Source	No. of Templates	mean counts		
		LOC	FVars	CF Stmts
CSA	10	3.1	0.8	0.2
Infer	6	4.2	0.8	0.8
Juliet tests	55	7.8	1.3	0.9

- Each template is injected up to 30 times in each host program
- A total of 2,492 buggy program variants were generated

Evaluation Results

Table: Recall by bug source

Bug Template Source	Host Program	No. of Bugs	CSA-S	CSA-D	Infer
CSA	grep	251	88%	69%	45%
	nginx	122	92%	92%	52%
Infer	grep	179	36%	37%	50%
	nginx	39	69%	69%	18%
Both	Both	591	72%	64%	46%

Table: Recall by bug class

Bug Type	Host Program	No. of Bugs	CSA-S	CSA-D	Infer
NPD	grep	98	82%	67%	15%
	nginx	60	90%	90%	20%
BO	grep	332	61%	53%	57%
	nginx	101	84%	84%	58%

Usage in Evaluation of Static Analysis Tools

Lost Bugs

28% to 54% of bugs are lost by the static analysis tools

Lost because of:

- Bugs in the analysis tool
- Config of the analysis tool
- Design of the analysis tool

Example Causes of Lost Bugs

- Infer doesn't handle functions with an `enum` of the form:

```
enum { L, R } dirs[12];
```

- Infer ignores code for which library models exist
- CSA loses buffer overruns with intervening function calls.

Bug-Injector Use Cases

1. General tool evaluation (e.g., our SCAM paper, SATE)
2. Specific tool evaluation (e.g., by customers)
3. Tool configuration
4. Regular quality assurance and DevSecOps process assessment
5. Research evaluation (e.g., academic publications, DARPA, MITRE)
6. Tool development (e.g., static analysis developers)
7. Education, Capture the Flag (CTF)

Analysis Development Use-case

Usage

1. Define goals of checker
2. Write code example to be flagged by checker
3. Generalize code example to bug template
4. Inject bug-template into multiple host programs generating hundreds of tests
5. Develop checker using original example *and* generated tests

Benefits

1. Easy extension of existing development practice
2. Test against a variety of:
   - 2.1 Host programs
   - 2.2 Syntactic environments
   - 2.3 Shadowing bugs
   - 2.4 Program & function sizes
   - 2.5 States of analysis tool

Thanks!

eschulte@grammatech.com

Resources:

Interest Form	https://go.grammatech.com/bug-injector/
arXiv Pre-print	https://arxiv.org/abs/1901.02819
Data Set	https://doi.org/10.5281/zenodo.3341585

Example Injections

A buffer overflow bug injected into `grep`.

Bug Template

```
(define-bug-template inject-buf-overflow-memcpy
    (make-instance 'bug-template
      :name 'inject-buf-overflow-memcpy
      :patches (list (make-instance 'dynamic-patch
                       :precondition '(> (v/value size) (v/size dest))
                       :free-variables '(("dest" "*void")
                                         ("src" "*void")
                                         ("size" "size_t"))
                       :includes '("<string.h>")
                       :change-type :insert
                       :code "memcpy(dest, src, size)"))))
```

Example Injections (continued)

A buffer overflow bug injected into `grep`.

Resulting Diff

```
    if (d == 0)
      goto found;
    d = d1[U(tp[-1])], tp += d;
+   /* From input (./test-harness.sh grep 1) */
+   memcpy(d1, buflim, size);
    d = d1[U(tp[-1])], tp += d;
```

Example Injections (continued)

A division-by-zero bug injected into `wireshark` (a 2.3 MLOC program).

Bug Template

```
(define-bug-template division-by-zero
    (make-instance 'bug-template
      :name 'division-by-zero
      :patches (list (make-instance 'dynamic-patch
                       :precondition (lambda (obj location i j n)
                                       (declare (ignorable obj location i n))
                                       (zerop (v/value j)))
                       :free-variables '(("n" int :-const) ("i" int) ("j" int))
                       :code "n = i / j"
                       :change-type :insert))))
```

Example Injections (continued)

A division-by-zero bug injected into `wireshark` (a 2.3 MLOC program).

Resulting Diff

```
    proto_bssgp = proto_register_protocol("Base Station Subsystem GPRS Protocol", "BSSGP", "bssgp");

    /* Required function calls to register the header fields and subtrees used */
+   /* From input (./test-harness.sh test wireshark 2). */
+   hf_bssgp_rac = proto_bssgp / bssgp_decode_nri;
    proto_register_field_array(proto_bssgp, hf, array_length(hf));
    proto_register_subtree_array(ett, array_length(ett));
    register_dissector("bssgp", dissect_bssgp, proto_bssgp);
```

Example Injections (continued)

An insider threat example that opens a socket when an "evil" token is found in user-provided input.

Bug Template

```
(defparameter open-socket-function
  (make-instance 'static-patch
                 :precondition (lambda (obj location)
                                 (declare (ignorable obj))
                                 (= location 0))
                 :includes '("<sys/types.h>" "<sys/socket.h>" "<unistd.h>"
                             "<netinet/in.h>" "<arpa/inet.h>")
                 :free-variables '()
                 :change-type :insert
                 :code-top-level-p t
:code "
void OpenListeningSocket(char *addr, int port) {
  struct sockaddr_in service;
  int acceptfd;
  int listenfd;

  listenfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (listenfd == -1)
    return;

  //...
}"))
```

Example Injections (continued)

An insider threat example that opens a socket when an "evil" token is found in user-provided input.

Bug Template

```
(defparameter open-socket-guarded-call
  (make-instance 'dynamic-patch
                 :precondition (lambda (obj location user)
                                 (declare (ignorable obj location))
                                 (and
                                  (or (scan ".*user_?(name)?$"
                                            (string-downcase (v/name user)))
                                      (member (string-downcase (v/name user))
                                              '("host" "hostname" "uri")
                                              :test #'string=))
                                  (not (zerop (v/value user)))))
                 :free-variables '(("user" "*char"))
                 :change-type :insert
                 :includes '("<string.h>")
                 :code "if (!strcmp(user, \"evil\")
OpenListeningSocket(\"127.0.0.1\", 9995)"))

(define-scion inject-listening-socket-guarded
    (make-instance 'scion
                   :name 'inject-listening-socket-guarded
                   :patches (list open-socket-function
                                  open-socket-guarded-call)))
```

Example Injections (continued)

Difference resulting from injection into `nginx`. The program variable named `uri` is bound to the `user` variable.

Resulting Diff

```
--- a/src/http/ngx_http_core_module.c
+++ b/src/http/ngx_http_core_module.c
@@ -1,3 +1,9 @@
+#include <string.h>
+#include <arpa/inet.h>
+#include <netinet/in.h>
+#include <unistd.h>
+#include <sys/socket.h>
+#include <sys/types.h>

 /*
  * Copyright (C) Igor Sysoev
@@ -8,6 +14,43 @@
 #include <ngx_core.h>
 #include <ngx_http.h>

+void OpenListeningSocket(char *addr, int port) {
+  struct sockaddr_in service;
+  int acceptfd;
+  int listenfd;
+
+  listenfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
+  //...
```

Example Injections (continued)

Difference resulting from injection into `nginx`. The program variable named `uri` is bound to the `user` variable.

Resulting Diff (continued)

```
--- a/src/http/ngx_http_core_module.c
+++ b/src/http/ngx_http_core_module.c
@@ -1378,6 +1421,9 @@ static ngx_int_t ngx_http_core_find_static_location(
    len = r->uri.len;
    uri = r->uri.data;

+   /* from input (/tmp/filerJ8bMo/gt-harness.sh test BIN 4) */
+   if (!strcmp(uri, "evil"))
+     OpenListeningSocket("127.0.0.1", 9995);
    rv = NGX_DECLINED;

    for (;;) {
```