An Excessively Short Introduction to LangSec

Explaining Very Little in Even Less Time

Perry E. Metzger

University of Pennsylvania
Department of Computer and Information Science

DSSS'17
July 20, 2017

Not My Work

This is work by Meredith Patterson, Sergey Bratus, et al.

Why am I discussing it?

Because it's important, and you should know about it.

Motivation

Fig. 1: The State of Modern Security

What's LangSec...

Observation:
Programs that do no I/O are generally secure.

What's LangSec...

Observation:
Programs that do no I/O are generally secure.

Related Observation:
Programs that do I/O usually do it wrong!

Programs as Language Recognizers

Programs that take input are *language recognizers!*

Computer Science has excellent theory for parsing.
Sadly, that excellent theory is rarely used.

This is where a large fraction of security holes come from.

What's a Shotgun Parsers?

A *shotgun parser* is the gordion knot of nested conditionals and loops that makes up the average ad hoc parser.

What happens if there's a discrepancy between the language accepted and the one that should be accepted?

VERY BAD THINGS!

Shotgun Parsers Wreck Security!

The average image library, network service, format converter, etc. uses a shotgun parser.

Parsers grounded in parsing theory rarely fail.
Shotgun parsers rarely succeed.

There are numerous CVEs attributable to shotgun parsers *every single week.*

Unparsers are a problem too!

When you're turning data into a string in a formal language, you're writing an unparser.

What if you make a mistake? What if what you generate wouldn't parse back the same way?

Almost all cross-site scripting, code injection, etc. is caused by unparsing problems.

Little Bobby Tables

Remember Little Bobby Tables from XKCD?

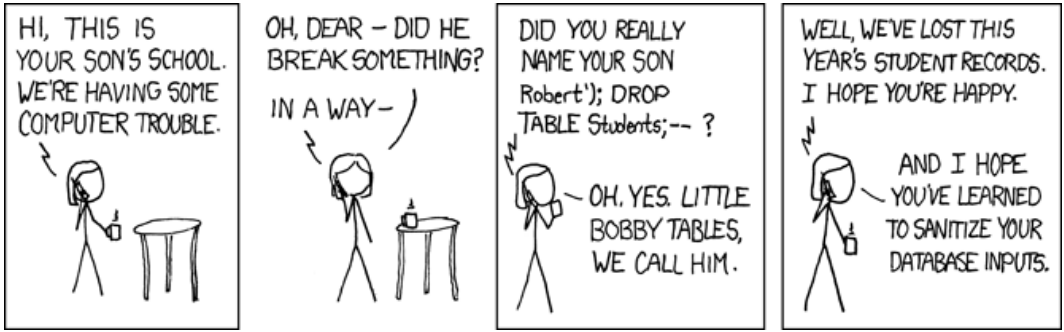

Fig. 2: An Unparser Failure

Little Bobby Tables is an Unparser Problem

Why? A variable containing the "first name":

```
Robert'); DROP TABLE students;--
```

is not a single SQL lexical token, so with a naive unparser,

```
Parse(Unparse(AST)) ≠ AST
```

and this violates a necessary security invariant.

How can *you* help?

Don't write shotgun parsers and unparsers!

Help construct sound and practical tools for generating parsers and unparsers — there are surprisingly few suitable for protocols, binary files, etc.

Formally verified protocol implementations, image libraries, etc. etc. need formally verified parser/unparser generators!

Learn more: `http://www.langsec.org/`