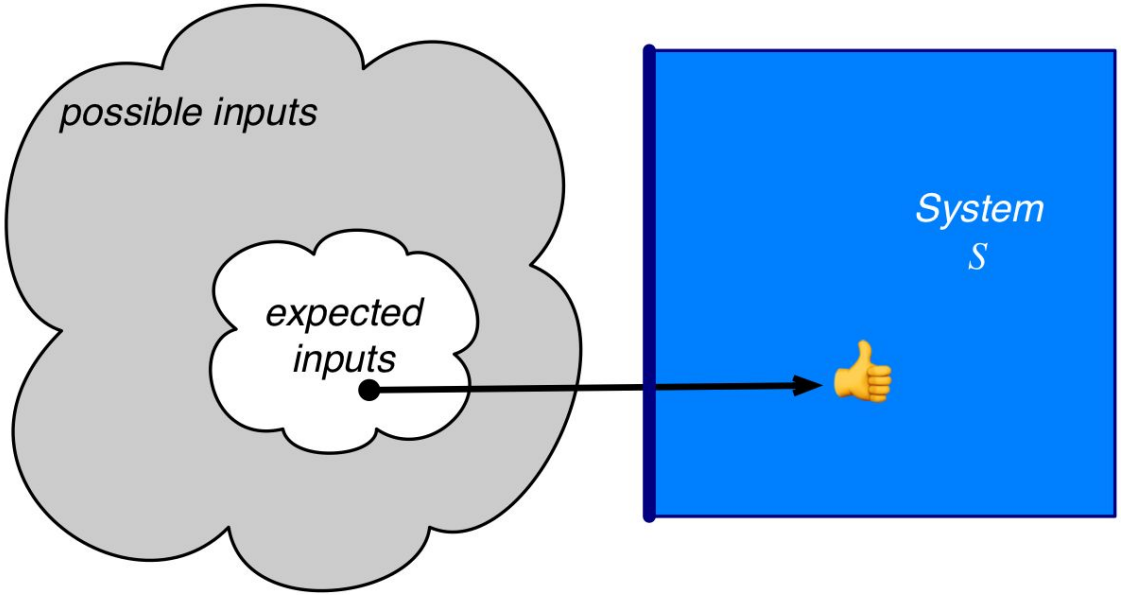
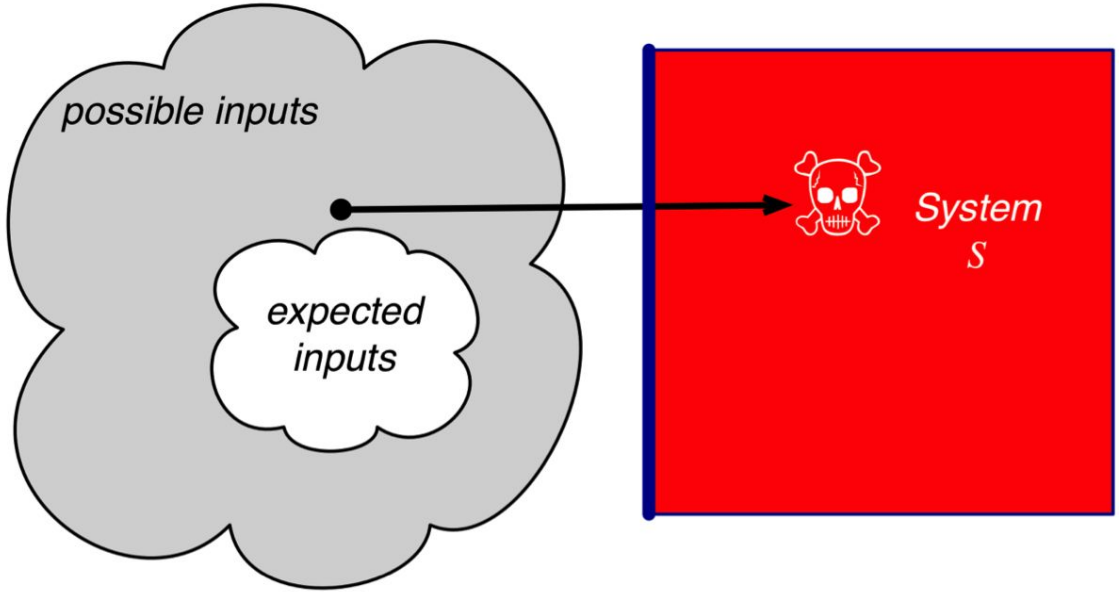
LangSec: A Principled Way to enforce Input Security

Prashant Anantharaman, PhD Student
Department of Computer Science
Dartmouth College
http://www.cs.dartmouth.edu/~pa/
pa@cs.dartmouth.edu

Problem: Zero-Days via Crafted Input



Problem: Zero-Days via Crafted Input



Endemic Everywhere

Search Results

There are **9879** CVE entries that match your search.

Name	Description
CVE-2019-9956	In ImageMagick 7.0.8-35 Q16, there is a stack-based buffer overflow in the function PopHexPixel of coders/ps.c, which allows an attacker to cause a denial of service or code execution via a crafted image file.
CVE-2019-9895	In PuTTY versions before 0.71 on Unix, a remotely triggerable buffer overflow exists in any kind of server-to-client forwarding.
CVE-2019-9773	An issue was discovered in GNU LibreDWG 0.7 and 0.7.1645. There is a heap-based buffer overflow in the function dwg_decode_eed_data at decode.c for the z dimension.
CVE-2019-9770	An issue was discovered in GNU LibreDWG 0.7 and 0.7.1645. There is a heap-based buffer overflow in the function dwg_decode_eed_data at decode.c for the y dimension.
CVE-2019-9767	Stack-based buffer overflow in Free MP3 CD Ripper 2.6, when converting a file, allows user-assisted remote attackers to execute arbitrary code via a crafted .wma file.
CVE-2019-9766	Stack-based buffer overflow in Free MP3 CD Ripper 2.6, when converting a file, allows user-assisted remote attackers to execute arbitrary code via a crafted .mp3 file.
CVE-2019-9687	PoDoFo 0.9.6 has a heap-based buffer overflow in PdfString::ConvertUTF16toUTF8 in base/PdfString.cpp.
CVE-2019-9675	** DISPUTED ** An issue was discovered in PHP 7.x before 7.1.27 and 7.3.x before 7.3.3. phar_tar_writeheaders_int in ext/phar/tar.c has a buffer overflow via a long link value. NOTE: The vendor indicates that the link value is used only when an archive contains a symlink, which currently cannot happen: "This issue allows theoretical compromise of security, but a practical attack is usually impossible."
CVE-2019-9627	A buffer overflow in the kernel driver CybKernelTracker.sys in CyberArk Endpoint Privilege Manager versions prior to 10.7 allows an attacker (without Administrator privileges) to escalate privileges or crash the machine by loading an image, such as a DLL, with a long path.
CVE-2019-9210	In AdvanceCOMP 2.1, png_compress in pngex.cc in advpng has an integer overflow upon encountering an invalid PNG size, which results in an attempted memcpy to write into a buffer that is too small. (There is also a heap-based buffer over-read.)
CVE-2019-9209	In Wireshark 2.4.0 to 2.4.12 and 2.6.0 to 2.6.6, the ASN.1 BER and related dissectors could crash. This was addressed in epan/dissectors/packet-ber.c by preventing a buffer overflow associated with excessive digits in time values.
CVE-2019-9134	Architectural Information System 1.0 and earlier versions have a Stack-based buffer overflow, allows remote attackers to execute arbitrary code.
CVE-2019-9125	An issue was discovered on D-Link DIR-878 1.12B01 devices. Because strncpy is misused, there is a stack-based buffer overflow vulnerability that does not require authentication via the HNAP_AUTH HTTP header.
CVE-2019-9077	An issue was discovered in GNU Binutils 2.32. It is a heap-based buffer overflow in process_mips_specific in readelf.c via a malformed MIPS option section.
CVE-2019-9075	An issue was discovered in the Binary File Descriptor (BFD) library (aka libbfd), as distributed in GNU Binutils 2.32. It is a heap-based buffer overflow in _bfd_archive_64_bit_slurp_armap in archive64.c.
CVE-2019-9036	An issue was discovered in libmatio.a in matio (aka MAT File I/O Library) 1.5.13. There is a heap-based buffer overflow in the function ReadNextFunctionHandle() in mat5.c.
CVE-2019-9027	An issue was discovered in libmatio.a in matio (aka MAT File I/O Library) 1.5.13. There is a heap-based buffer overflow problem in the function ReadNextCell() in mat5.c.
CVE-2019-9026	An issue was discovered in libmatio.a in matio (aka MAT File I/O Library) 1.5.13. There is a heap-based buffer overflow in the function InflateVarName() in inflate.c when called from ReadNextCell in mat5.c.
CVE-2019-9019	The British Airways Entertainment System, as installed on Boeing 777-36N(ER) and possibly other aircraft, does not prevent the USB charging/data-transfer feature from interacting with USB keyboard and mouse devices, which allows physically proximate attackers to conduct unanticipated attacks against Entertainment applications, as demonstrated by using mouse copy-and-paste actions to trigger a Chat buffer overflow or possibly have unspecified other impact.

Endemic Everywhere

Search Results

There are **873** CVE entries that match your search.

Name	Description
CVE-2019-9712	An issue was discovered in Joomla! before 3.9.4. The JSON handler in com_config lacks input validation, leading to XSS.
CVE-2019-8362	DedeCMS through V5.7SP2 allows arbitrary file upload in dede/album_edit.php or dede/album_add.php, as demonstrated by a dede/album_edit.php?dopost=save&formzip=1 request with a ZIP archive that contains a file such as "1.jpg.php" (because input validation only checks that .jpg, .png, or .gif is present as a substring, and does not otherwise check the file name or content).
CVE-2019-7385	An authenticated shell command injection issue has been discovered in Raisecom ISCOM HT803G-U, HT803G-W, HT803G-1GE, and HT803G GPON products with the firmware version ISCOMHT803G-U_2.0.0_140521_R4.1.47.002 or below, The values of the newpass and confpass parameters in /bin/WebMGR are used in a system call in the firmware. Because there is no user input validation, this leads to authenticated code execution on the device.
CVE-2019-7384	An authenticated shell command injection issue has been discovered in Raisecom ISCOM HT803G-U, HT803G-W, HT803G-1GE, and HT803G GPON products with the firmware version ISCOMHT803G-U_2.0.0_140521_R4.1.47.002 or below. The value of the fmgpon_loid parameter is used in a system call inside the boa binary. Because there is no user input validation, this leads to authenticated code execution on the device.
CVE-2019-7352	Self - Stored Cross Site Scripting (XSS) exists in ZoneMinder through 1.32.3, as the view 'state' (aka Run State) (state.php) does no input validation to the value supplied to the 'New State' (aka newState) field, allowing an attacker to execute HTML or JavaScript code.
CVE-2019-7345	Self - Stored Cross Site Scripting (XSS) exists in ZoneMinder through 1.32.3, as the view 'options' (options.php) does no input validation for the WEB_TITLE, HOME_URL, HOME_CONTENT, or WEB_CONSOLE_BANNER value, allowing an attacker to execute HTML or JavaScript code. This relates to functions.php.
CVE-2019-7331	Self - Stored Cross Site Scripting (XSS) exists in ZoneMinder through 1.32.3 while editing an existing monitor field named "signal check color" (monitor.php). There exists no input validation or output filtration, leaving it vulnerable to HTML Injection and an XSS attack.
CVE-2019-6690	python-gnupg 0.4.3 allows context-dependent attackers to trick gnupg to decrypt other ciphertext than intended. To perform the attack, the passphrase to gnupg must be controlled by the adversary and the ciphertext should be trusted. Related to a "CWE-20: Improper Input Validation" issue affecting the affect functionality component.
CVE-2019-6555	Cscape, 9.80 SP4 and prior. An improper input validation vulnerability may be exploited by processing specially crafted POC files. This may allow an attacker to read confidential information and remotely execute arbitrary code.
CVE-2019-6553	A vulnerability was found in Rockwell Automation RSLinx Classic versions 4.10.00 and prior. An input validation issue in a .dll file of RSLinx Classic where the data in a Forward Open service request is passed to a fixed size buffer, allowing an attacker to exploit a stack-based buffer overflow condition.
CVE-2019-6547	Delta Industrial Automation CNCSoft, CNCSoft ScreenEditor Version 1.00.84 and prior. An out-of-bounds read vulnerability may cause the software to crash due to lacking user input validation for processing project files.
CVE-2019-6220	An out-of-bounds read was addressed with improved input validation. This issue is fixed in macOS Mojave 10.14.3. An application may be able to read restricted memory.
CVE-2019-6218	A memory corruption issue was addressed with improved input validation. This issue is fixed in iOS 12.1.3, macOS Mojave 10.14.3, tvOS 12.1.2. A malicious application may be able to execute arbitrary code with kernel privileges.
CVE-2019-6210	A memory corruption issue was addressed with improved input validation. This issue is fixed in iOS 12.1.3, macOS Mojave 10.14.3, tvOS 12.1.2, watchOS 5.1.3. A malicious application may be able to execute arbitrary code with kernel privileges.
CVE-2019-6209	An out-of-bounds read issue existed that led to the disclosure of kernel memory. This was addressed with improved input validation. This issue is fixed in iOS 12.1.3, macOS Mojave 10.14.3, tvOS 12.1.2, watchOS 5.1.3. A malicious application may be able to determine kernel memory layout.
CVE-2019-6200	An out-of-bounds read was addressed with improved input validation. This issue is fixed in iOS 12.1.3, macOS Mojave 10.14.3. An attacker in a privileged network position may be able to execute arbitrary code.
CVE-2019-5916	Input validation issue in POWER EGG(Ver 2.0.1, Ver 2.02 Patch 3 and earlier, Ver 2.1 Patch 4 and earlier, Ver 2.2 Patch 7 and earlier, Ver 2.3 Patch 9 and earlier, Ver 2.4 Patch 13 and earlier, Ver 2.5 Patch 12 and earlier, Ver 2.6 Patch 8 and earlier, Ver 2.7 Patch 6 and earlier, Ver 2.7 Government Edition Patch 7 and

Endemic Everywhere

Search Results

There are **20797** CVE entries that match your search.

Name	Description
CVE-2019-9977	The renderer process in the entertainment system on Tesla Model 3 vehicles mishandles JIT compilation, which allows attackers to trigger firmware code execution, and display a crafted message to vehicle occupants.
CVE-2019-9969	XnView Classic 2.48 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to xnview+0x385399.
CVE-2019-9968	XnView Classic 2.48 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to ntdll!RtlQueueWorkItem.
CVE-2019-9967	XnView Classic 2.48 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to ntdll!RtlPrefixUnicodeString.
CVE-2019-9966	XnView Classic 2.48 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to xnview+0x38536c.
CVE-2019-9965	XnView MP 0.93.1 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to ntdll!RtlReAllocateHeap.
CVE-2019-9964	XnView MP 0.93.1 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to ntdll!RtlpNtMakeTemporaryKey.
CVE-2019-9963	XnView MP 0.93.1 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to ntdll!RtlFreeHeap.
CVE-2019-9962	XnView MP 0.93.1 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to VCRUNTIME140!memcpy.
CVE-2019-9956	In ImageMagick 7.0.8-35 Q16, there is a stack-based buffer overflow in the function PopHexPixel of coders/ps.c, which allows an attacker to cause a denial of service or code execution via a crafted image file.
CVE-2019-9903	PDFDoc::markObject in PDFDoc.cc in Poppler 0.74.0 mishandles dict marking, leading to stack consumption in the function Dict::find() located at Dict.cc, which can (for example) be triggered by passing a crafted pdf file to the pdfunite binary.
CVE-2019-9889	In Vanilla before 2.6.4, a flaw exists within the getSingleIndex function of the AddonManager class. The issue results in a require call using a crafted type value, leading to Directory Traversal with File Inclusion. An attacker can leverage this vulnerability to execute code under the context of the web server.
CVE-2019-9878	There is an invalid memory access in the function GfxIndexedColorSpace::mapColorToBase() located in GfxState.cc in Xpdf 4.0.0, as used in pdfalto 0.2. It can be triggered by (for example) sending a crafted pdf file to the pdftops binary. It allows an attacker to cause Denial of Service (Segmentation fault) or possibly have unspecified other impact.
CVE-2019-9877	There is an invalid memory access vulnerability in the function TextPage::findGaps() located at TextOutputDev.c in Xpdf 4.01, which can (for example) be triggered by sending a crafted pdf file to the pdftops binary. It allows an attacker to cause Denial of Service (Segmentation fault) or possibly have unspecified other impact.
CVE-2019-9785	gitnote 3.1.0 allows remote attackers to execute arbitrary code via a crafted Markdown file, as demonstrated by a javascript:window.parent.top.require('child_process').execFile substring in the onerror attribute of an IMG element.
CVE-2019-9767	Stack-based buffer overflow in Free MP3 CD Ripper 2.6, when converting a file, allows user-assisted remote attackers to execute arbitrary code via a crafted .wma file.
CVE-2019-9766	Stack-based buffer overflow in Free MP3 CD Ripper 2.6, when converting a file, allows user-assisted remote attackers to execute arbitrary code via a crafted .mp3 file.
CVE-2019-9760	FTPGetter Standard v.5.97.0.177 allows remote code execution when a user initiates an FTP connection to an attacker-controlled machine that sends crafted

Endemic Everywhere

How malformed packets caused CenturyLink's 37-hour, nationwide outage

JON BRODKIN - 8/19/2019, 4:15 PM

CenturyLink's nationwide, 37-hour outage in December 2018 disrupted 911 service for millions of Americans and prevented completion of at least 886 calls to 911, a new Federal Communications Commission report said.
… The 37-hour outage began on December 27 and "was caused by an equipment failure that was exacerbated by a network configuration error," the FCC said. CenturyLink estimates that more than 12.1 million phone calls on its network "were blocked or degraded due to the incident," the FCC said. Additionally, about 1.1 million of CenturyLink's DSL customers lost service for parts of the 37 hours. Another 2.6 million DSL customers "may have experienced degraded service," the FCC said….

Root cause

Problems began the morning of December 27 when "a switching module in CenturyLink's Denver, Colorado, node spontaneously generated four malformed management packets," the FCC report said.

https://arstechnica.com/information-technology/2019/08/centurylinks-37-hour-outage-blocked-911-service-for-17-million-people/

Vulnerabilities

```
DD 82 00 00 0C 01 00 00 01 rnd(11) rnd(11)
05 64 06 44 64 00 64 00 FF F2 C0 1D 0A
DD 82 00 00 0A 02 01 00 00 FF FF
FA 82 00 00 01 00 02 00 00 00 00 FF FF FF FF
```

Unsolicited Response

Group 1 Variation 0

Sizeless?!

4 byte start/stop

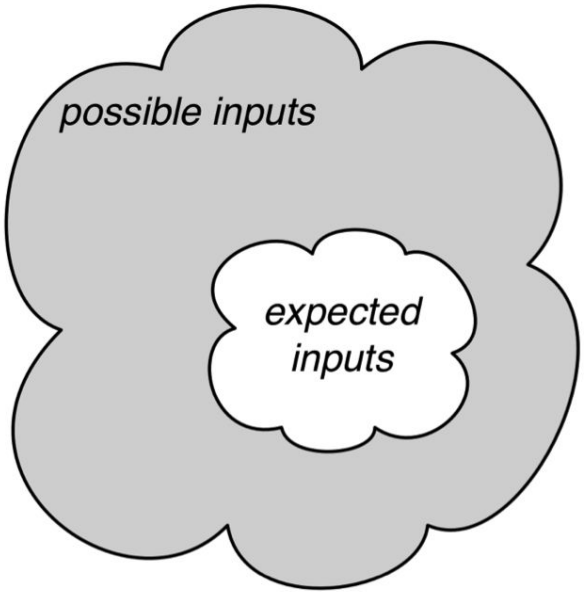
0

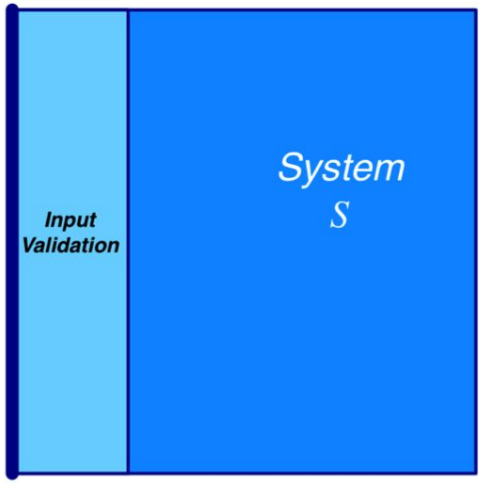
4294967295

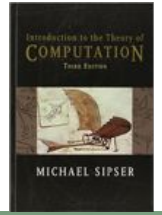
- infinite loop
- missing data
- integer overflow?
- accepts broadcast

From: Adam Crain, Chris Sistrunk “Project Robus, Master Serial Killer”, S4x14

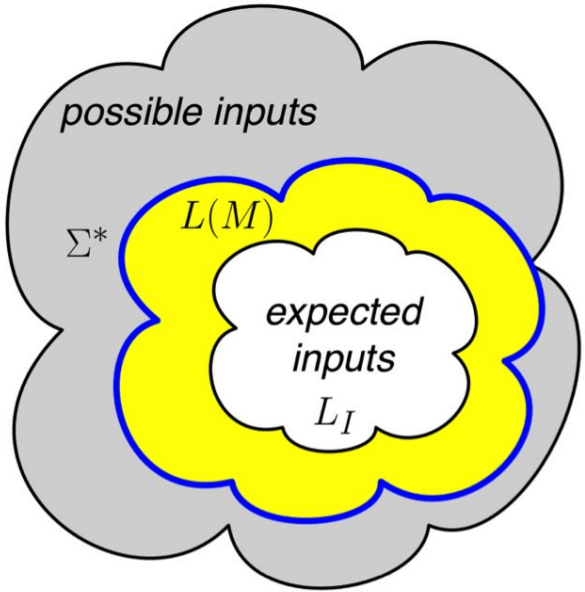
The LangSec Approach to the Problem

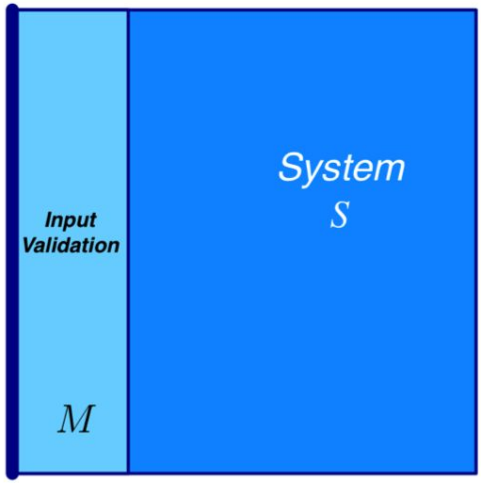




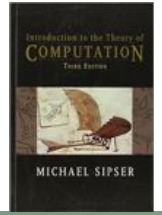


The LangSec Approach to the Problem

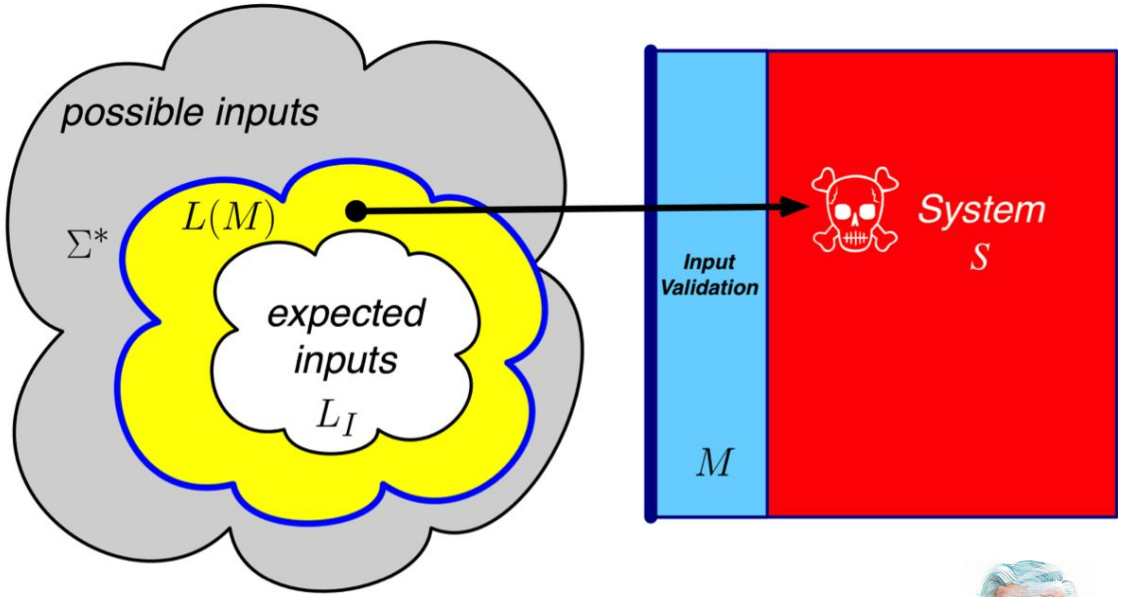




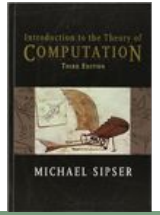
$$L_I \stackrel{?}{=} L(M)$$



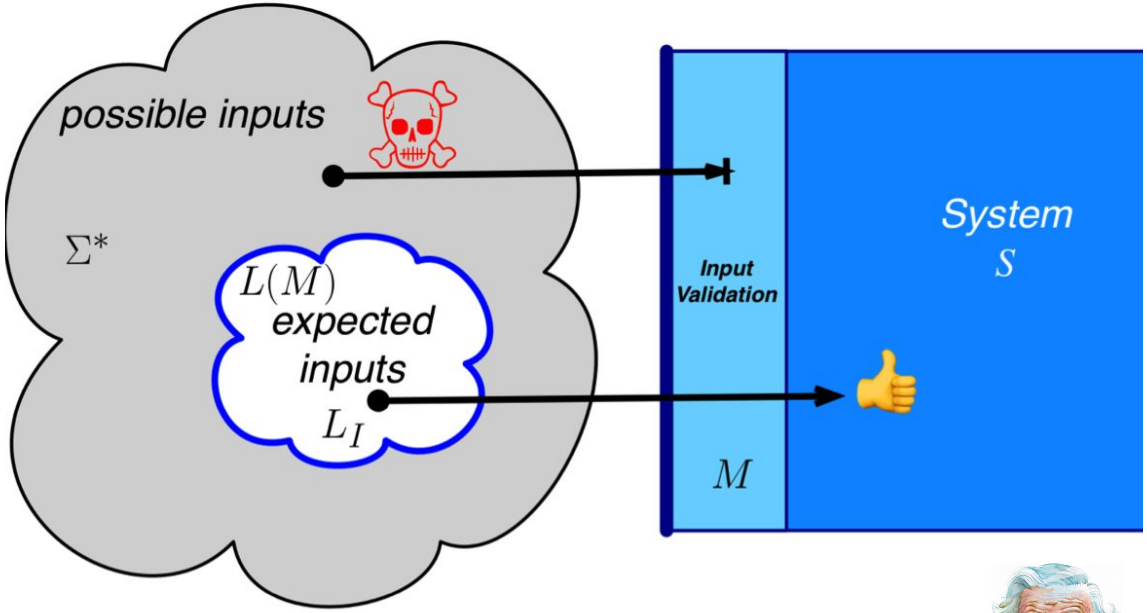
The LangSec Approach to the Problem



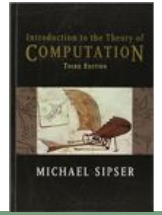
$$L_I \neq L(M)$$



The LangSec Approach to Solving It

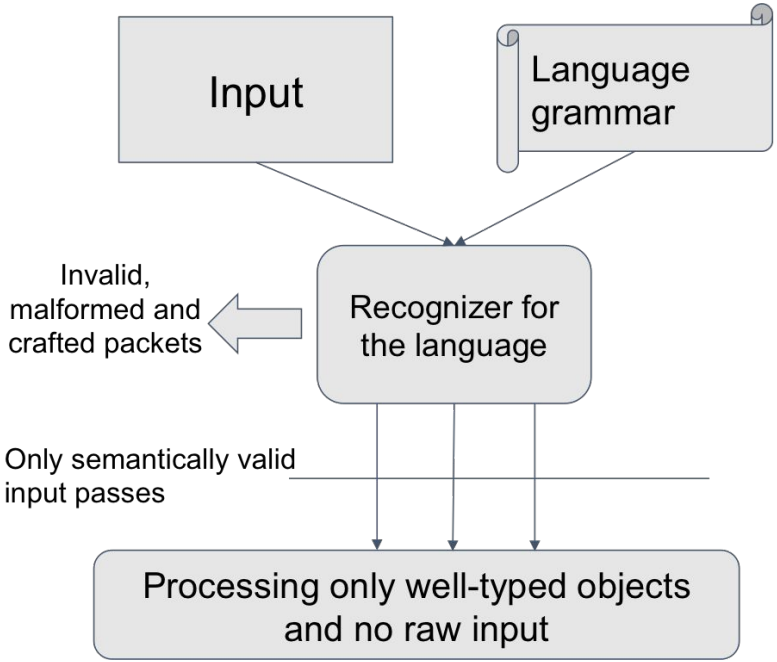


$$L_I = L(M)$$



LangSec in Practice

- Solution: LangSec Parsers
  1. Define a grammar that represents a "secure subset" of the protocol.
     - No more than context-free!
     - ...or maybe PEG
  2. Build a parser that accepts *only* this grammar.
  3. Use this parser *everywhere* this protocol is parsed.
- *And we're standardizing toolkits to make this easy for any ICS developer!*
-



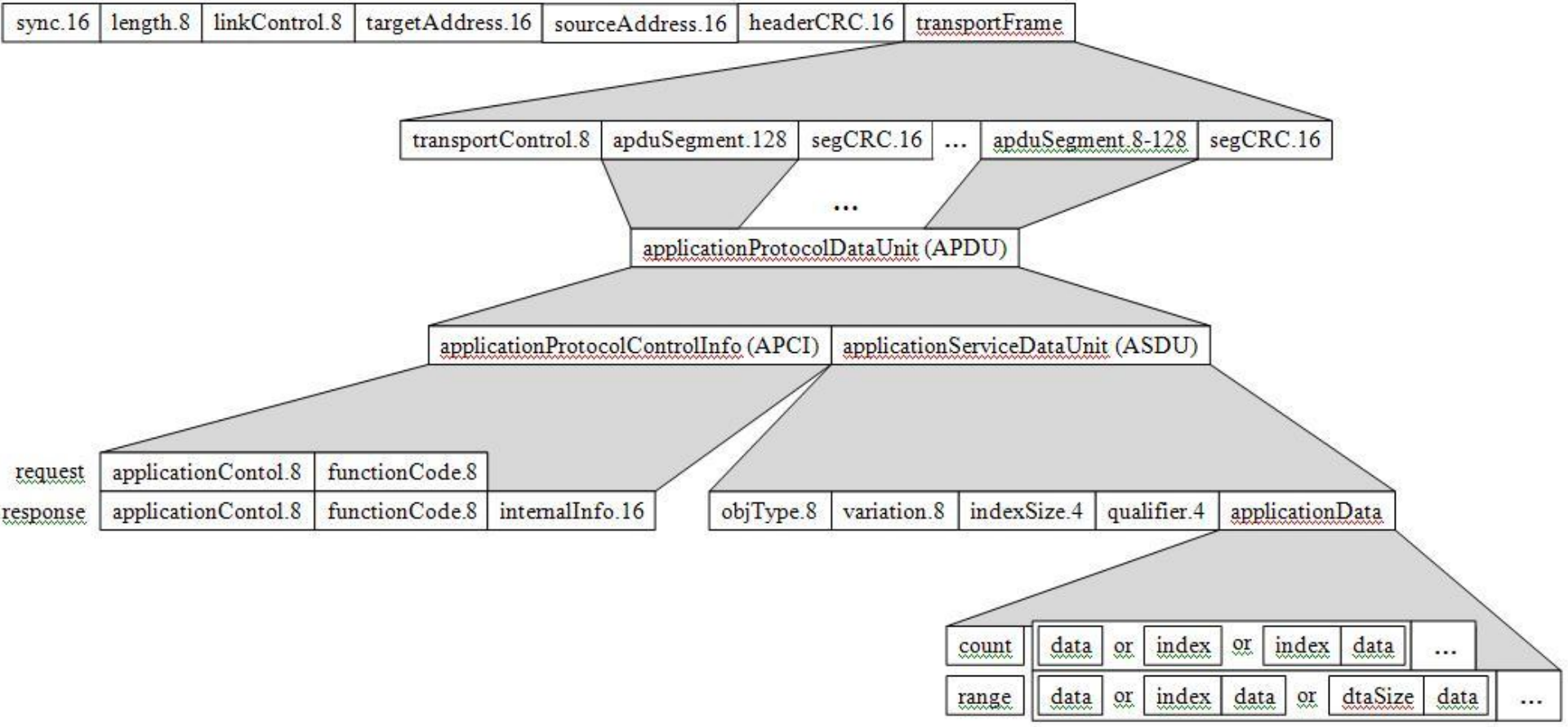
Parsing & protocol anti-patterns

- "**Shotgun parsers**": input validity checks **intermixed** with processing code; no clear separation boundary
  - OpenSSL's Heartbleed, GNU TLS Hello bug, …
- Unnecessarily complex syntax (e.g., **context-sensitive** where **context-free** or **regular** would suffice)
  - Objects' interpretation & legality depends on sibling object contents
- Parser **differentials** (parsers disagree about message contents)
  - X.509 CA vs client bugs, Android Master Key bugs, …

Parser combinators: a natural choice

- **Hammer** parser construction kit: C/C++
  - Bindings for Java, Python, Ruby, .NET, Go
  - Three algorithmic parsing back-ends
- Freely available on GitHub:
  https://github.com/UpstandingHackers/hammer

Parser combinators at a glance (1)

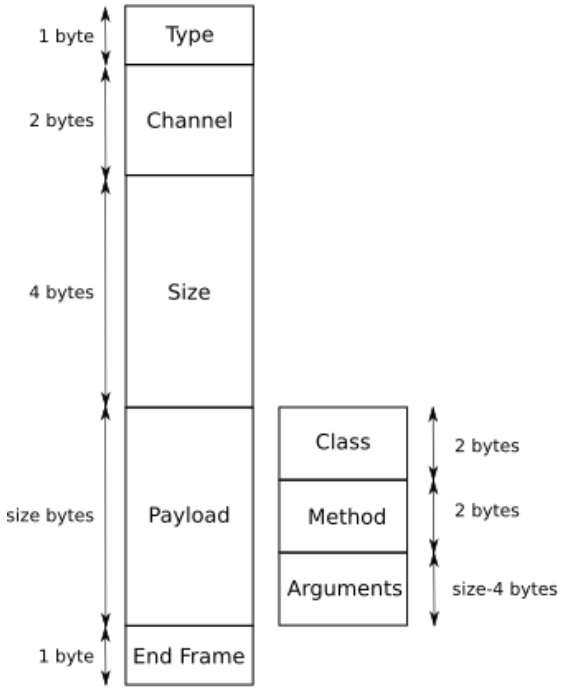


Parser combinators at a glance (2)

```
05 64 14 F3
01 00 00 04
0A 3B C0 C3
01 3C 02 06
3C 03 06 3C
04 06 3C 01
06 9A 12
```

```
start = h_token("\x05\x64");
len = h_int_range(h_uint8(), 5, 255);
ctrl = h_uint8();
dst = h_uint16();
src = h_int_range(h_uint16(), 0, 65519);
crc = h_uint16();
hdr = h_attr_bool(h_sequence(h_ignore(start),
        len, ctrl, dst, src, crc, NULL),
        validate_crc);
frame = h_attr_bool(h_sequence(hdr,
          h_optional(transport_frame),
          h_end_p(), NULL),validate_len);
```

Code that looks like grammar: AMQP



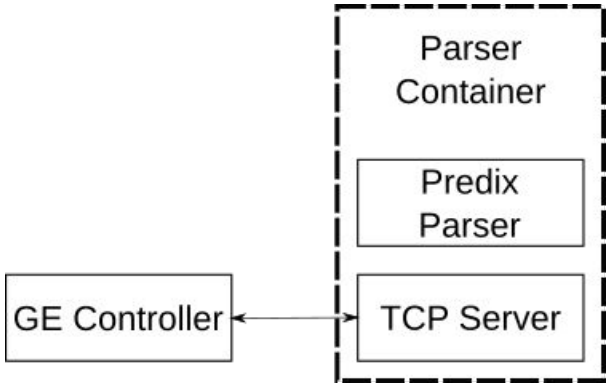

```
h_sequence(
    h_uint8(),     # Type
    h_uint16(),    # Channel

    h_and(h_length_value(
        h_uint32(),
        h_ch_range('\x00', '\xff'))),
    h_uint32(),     # Size
    h_sequence(
        h_uint16(), # Class
        h_uint16(), # Method
        h_optional(arguments)), # Arguments
    h_ch('\xce'),   # End Frame
    h_end_p(), NULL);
```

Collaborating with GE Research: *Predix Protocol Grammar*

$$
\begin{aligned}
\text{message} &\rightarrow \text{body messageId} \\
\text{whitespaces} &\rightarrow \text{” ” | \textbackslash n | \textbackslash t | \textbackslash r} \\
\text{colon} &\rightarrow \text{: | whitespaces colon | colon whitespace} \\
\text{messageId} &\rightarrow \text{”messageId” colon ”flexpipe”} \\
\text{body} &\rightarrow \text{body element} \\
\text{element} &\rightarrow \text{attribute datapoints name} \\
\text{sourcevalues} &\rightarrow \text{”CDP”} \\
\text{attribute} &\rightarrow \text{”attributes”:”source”: } sourcevalues \\
\text{datapoints} &\rightarrow \text{[[} timestamp \text{ , } double, range(0..3) \text{]]} \\
\text{name} &\rightarrow \text{”hmi.signal1” | ”hmi.signal2” | ”hmi.signal3”}
\end{aligned}
$$

Architecture

- Listens on a port for packet payloads
- It is an application layer protocol
- Server passes it on to the Predix parser which returns the parsed object or False if the parsing failed



Parser Example

{"body":[{"attributes":

{"source":"CDP"},

"datapoints":[[1519416378549,0.8717407888303543,3]],"name":"hmi.signal1"},

{"attributes":{"source":"CDP"},"datapoints":[[1519416378549,2.0,3]],"name":"hmi.signal2"},

{"attributes":{"source":"CDP"},"datapoints":[[1519416378549,-0.5333330016085733,3]],"name":"hmi.signal3"}],

"messageId":"flex-pipe"}


```
def json_input_parser():
    whitespaces = h.many(h.in_(" \r\n\t"))
    jsonopen = h.middle(whitespaces, h.ch('{'), whitespaces)
    jsonclose = h.middle(whitespaces, h.ch('}'), whitespaces)
    comma =  h.middle(whitespaces, h.ch(','), whitespaces)
    colon =  h.middle(whitespaces, h.ch(':'), whitespaces)
    arrayopen = h.sequence(whitespaces, h.ch('['), whitespaces)
    arrayclose = h.sequence(whitespaces, h.ch(']'), whitespaces)
    signal_left = h.token("\"name\"")
    signal_names = h.choice(h.token("\"hmi.signal1\""),
                            h.token("\"hmi.signal2\""),
                            h.token("\"hmi.signal3\""))
    epoch = h.many1(h.ch_range('\x30', '\x39'))
    optional_minus = h.optional(h.ch('-'))
    negative_val = h.sequence(optional_minus, epoch, h.ch('.'), epoch)
    datapoints_left = h.token("\"datapoints\"")
    datapoints = h.sequence(arrayopen, arrayopen,
                            epoch, comma, negative_val,
                            comma, h.ch_range('0', '3'),
                            arrayclose, arrayclose)
```

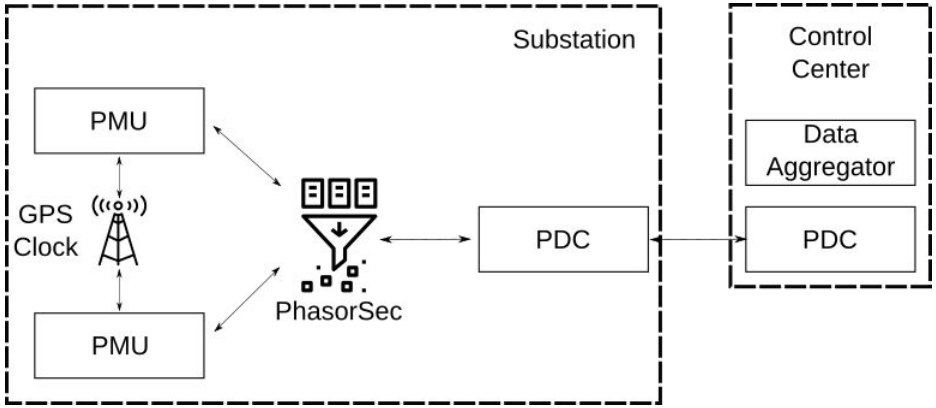
Unit testing and fuzzing

- Exhaustively validate over a corpus of over 500 valid messages
- Unit tests using python unittests library
- Fuzzing using pyjfuzz – A python JSON fuzzing library
  - Our parser showed resiliency and 0 crashes or exceptions


```
'{"body": [{"attributes": {"source": {"param": "file:///pro\\u008c\\u0001self/CDP"}}, "name": "", "datapoints": [[1520406819826, 4278190080.0
, 4278190083]]}, {"attributes": {"source": "..\\\\..\\\\..\\\\..\\\\CD\\u0014\\u0000ini"}, "name": 0, "datapoints": [[324923410, "2.0", 18961
49375]]}, {"attributes": {"source": false}, "name": ["hmi.signal3"], "datapoints": [[1519416964095, 255.0, "3"]]}], "messageId": ""}'
'{"body": [{"attributes": {"source": ""}, "name": "", "datapoints": [[1520406819826, -201730886.8, 255]]}, {"attributes": {"source": "file://
/proc/selfffffffffffffffffffffffffffffff/CDP"}, "name": false, "datapoints": [[true, 16777213.0, 16777212]]}, {"attributes": {"source": 0},
"name": "..\\\\..\\\\..\\\\..\\\\hmi.si\\u0019\\u0000al3.ini", "datapoints": [[1520406819826, true, 4278190083]]}], "messageId": 0}'
'{"body": [{"attributes": {"source": 0}, "name": false, "datapoints": [[1519416964095, 255.0, 4278190083]]}, {"attributes": {"source": ""}, "
name": {"param": "file:///procse\\u00d8f//hmi.signal2"}, "datapoints": [[-1519416964082, -2.0, 4278190083]]}, {"attributes": {"source": 0}, "
name": {"param": "..............................................................\\\\..\\\\..\\\\..\\\\hmi.signal3.ini"}, "datapoints":
 [[1519416964095, 16777214.0, 255]]}], "messageId": 0}'
'{"body": [{"attributes": {"source": ["CDP"]}, "name": {"param": "file:///proc/self/hmi\\u000b\\u0000ignal1"}, "datapoints": [[-1519416964082
, 255.0, true]]}, {"attributes": {"source": "/../../../..[/et}c/CDP"}, "name": {"param": "file:///\\u0007\\u0000\\u0000\\u0000/self/hmi.signa
l2"}, "datapoints": [[-1264447495, 16777213.0, true]]}, {"attributes": {"source": ["CDP"]}, "name": {"param": "/../../../.\\u00e0\\u0002\\u00
00\\u0000c/hmi.sig\\u001b\\u000013"}, "datapoints": [[1519416964095, "1.06666613742", 16777212]]}], "messageId": ["flex-pipe"]}'
```


Parsers as an IDS

- Within a substation, we intercept all communications through the router and parse them.
  - C37.118, DNP3, Modbus, IEC61850 (variants)
- Latencies measured on a ARM Cortex A17 with 2GB RAM.
- Our parsers survived AFL fuzzing.



Frame	CPU Time	Lines of code
Command Frame	20 μs	29
Configuration Frame	13 μs	71
Data Frame	27 μs	56
Header Frame	80 μs	30

Various Ongoing LangSec Initiatives

IDS for the Power Grid

DARPA RADICS: Threat Intelligence and Grid Recovery (TIGR) project

(With SRI International)

File Format Validators

DARPA SAFEDOCS: Parsley Project

(With SRI International)

Securing Hardware Architectures with Parsers in FPGAs

DARPA GAPS

(Extending our collaboration with GE Research: David Safford, William Smith and Kepa Krzysztof.)

Results and Next Steps

Current LangSec parsers include:

- DNP3 (Bratus et al. '16)
- C37.118 (Anantharaman et al. '18)
- IEC 61850
- GOOSE
- MMS
- HTTP HMI Parsers

Roadmap to upgrade a whole substation (Millian et al. ’19)

Future directions include tools for:

- automated discovery of legacy grammars
- automated generation of fuzz-testers
- automated discovery of differential parsing vulnerabilities
- fixing zero-days in black-box code
- discovering and fixing holes in ICS wireless interfaces
- formal verification of parsers and protocol specs

Thank you!

Questions?

https://cs.dartmouth.edu/~pa

Prashant: pa@cs.dartmouth.edu

Sean Smith: sws@cs.dartmouth.edu