Parsing Real-World Data Formats

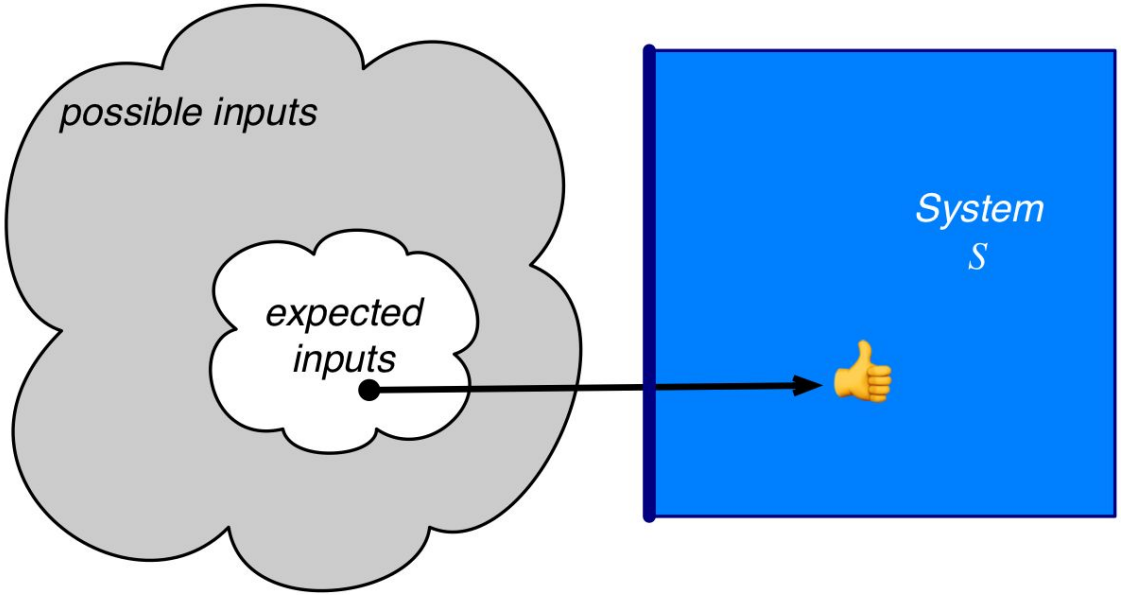
Prashant Anantharaman, PhD Student
Department of Computer Science
Dartmouth College
https://prashant.at
pa@cs.dartmouth.edu

Boston Computing Club: 29th November 2020

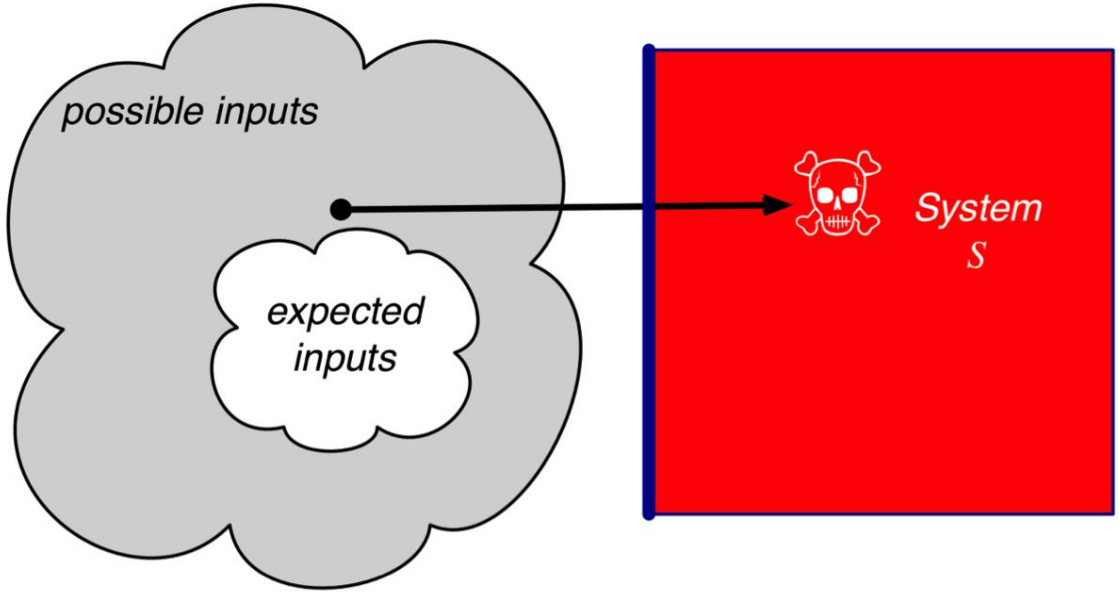
About me

- Master’s at Dartmouth in 2017. Ugrad: College of Engineering Guindy in 2015 in Computer Science and Engineering.
- Originally focussed on building parsers for IoT and SCADA protocols.
- Recently shifted to building a Data Description Language for File Formats and Network Protocols.

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

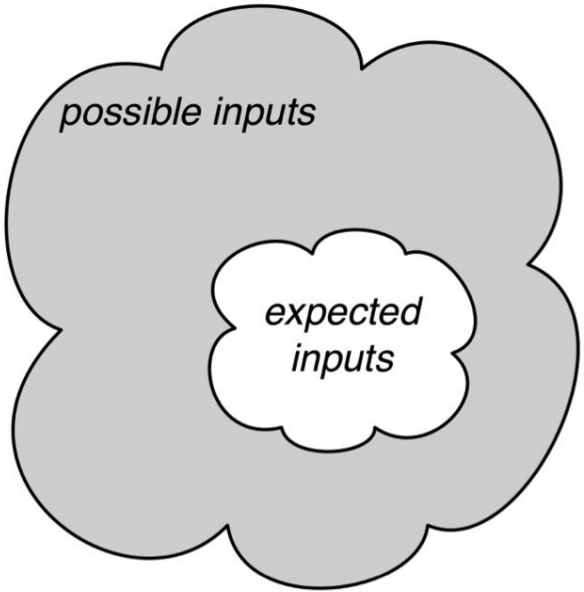
Endemic Everywhere

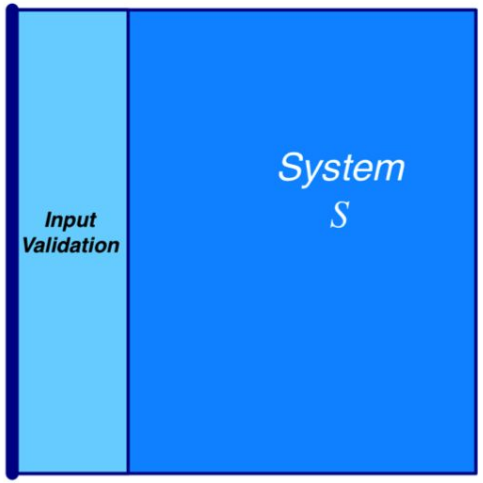
Search Results

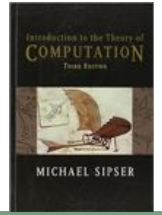
There are **20797** CVE entries that match your search.

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

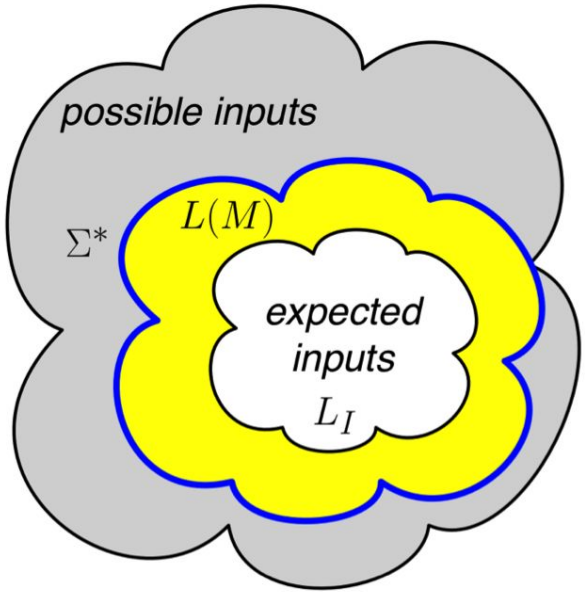
The LangSec Approach to the Problem

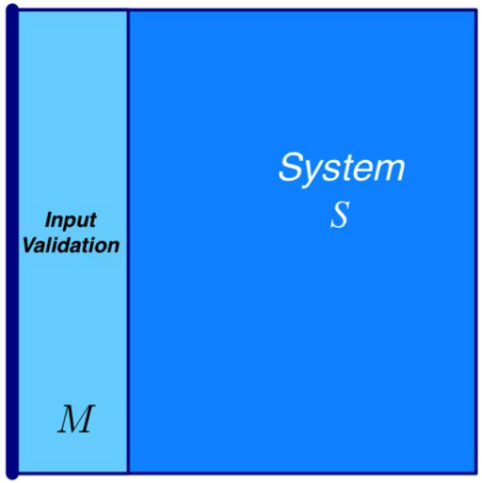




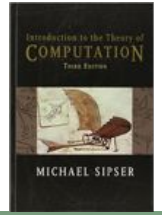


The LangSec Approach to the Problem

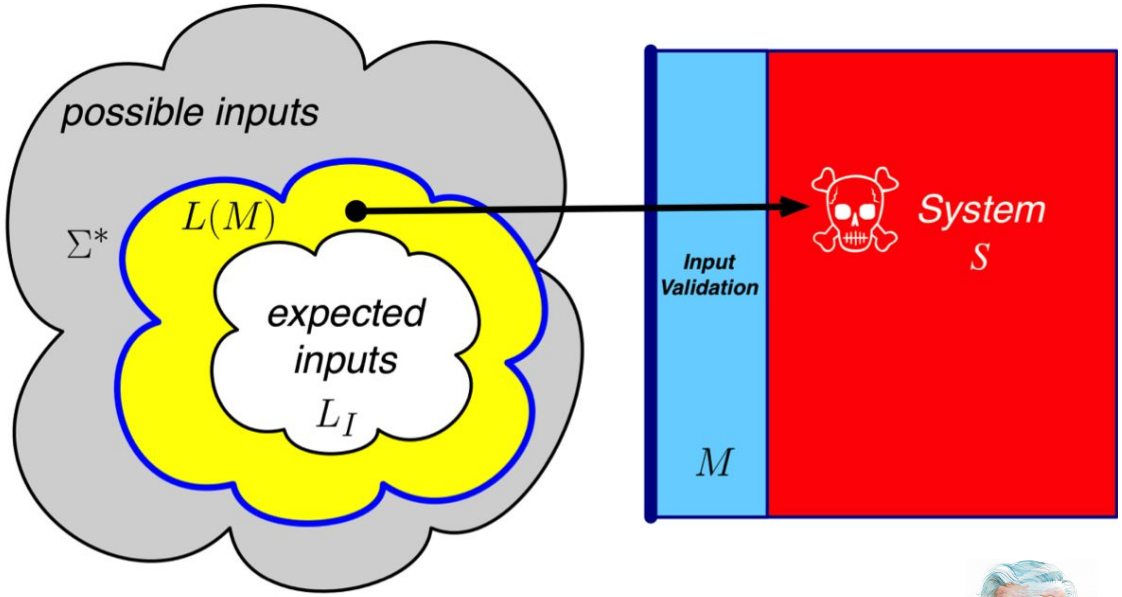




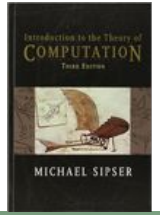
$$L_I \stackrel{?}{=} L(M)$$



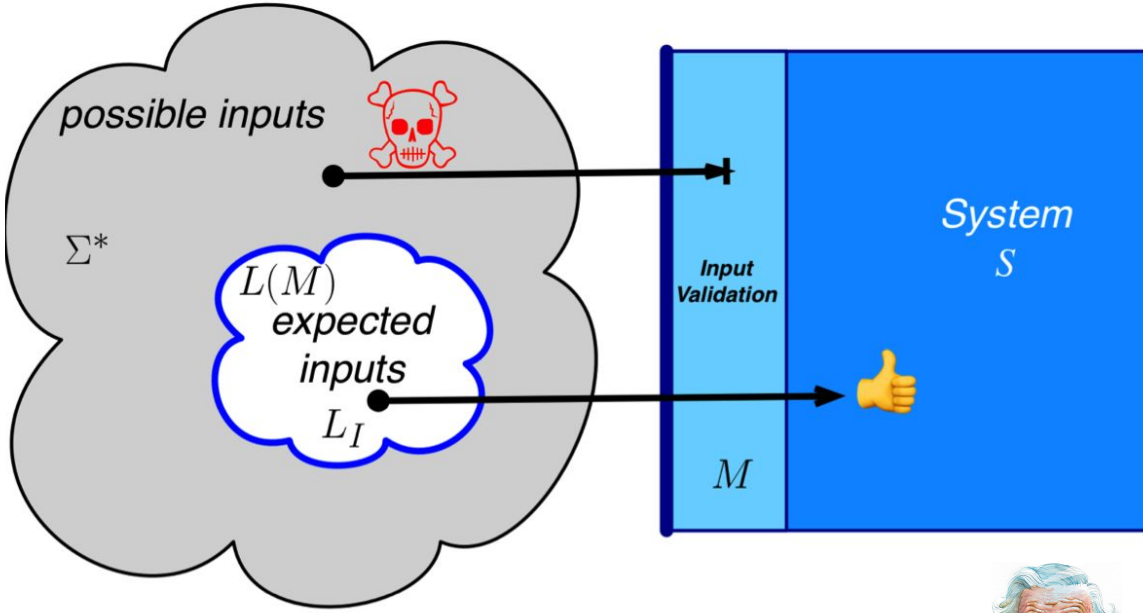
The LangSec Approach to the Problem



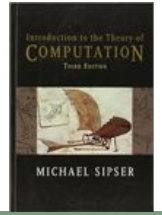
$$L_I \neq L(M)$$



The LangSec Approach to Solving It

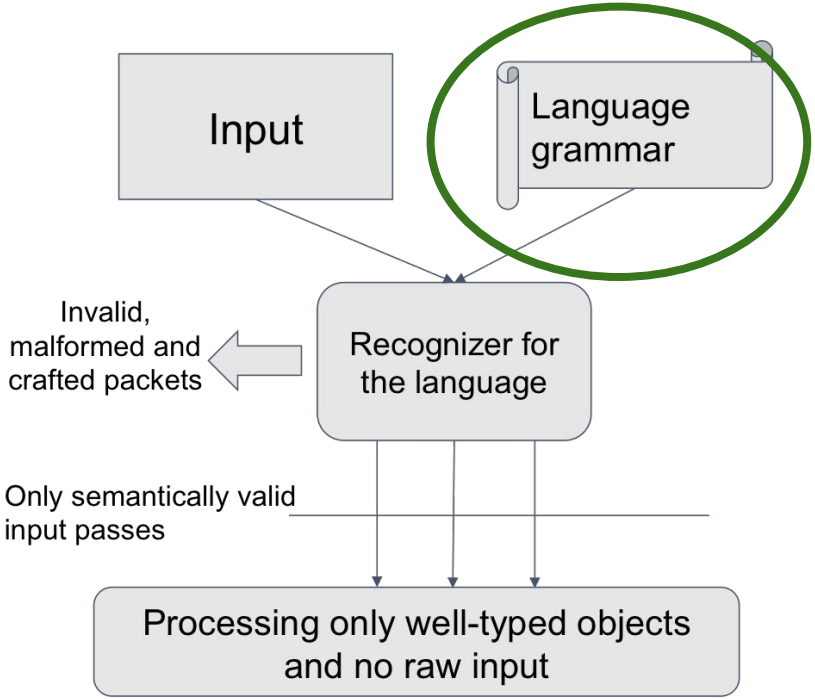


$$L_I = L(M)$$



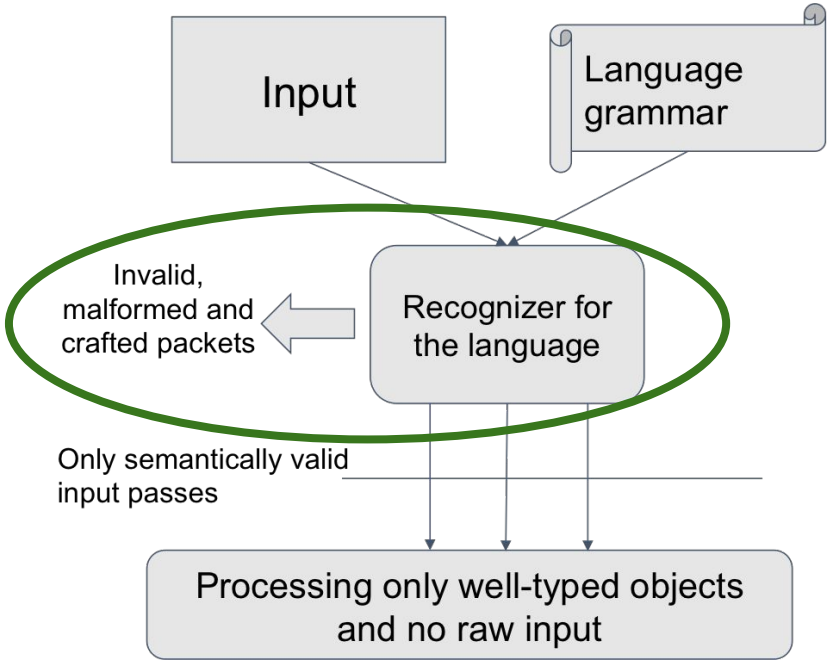
Preventing Zero-days via Crafted Input

- *Solution: LangSec Parsers*
  1. Define a grammar that represents a "secure subset" of the protocol.
     - No more than context-free!
     - ...or maybe PEG



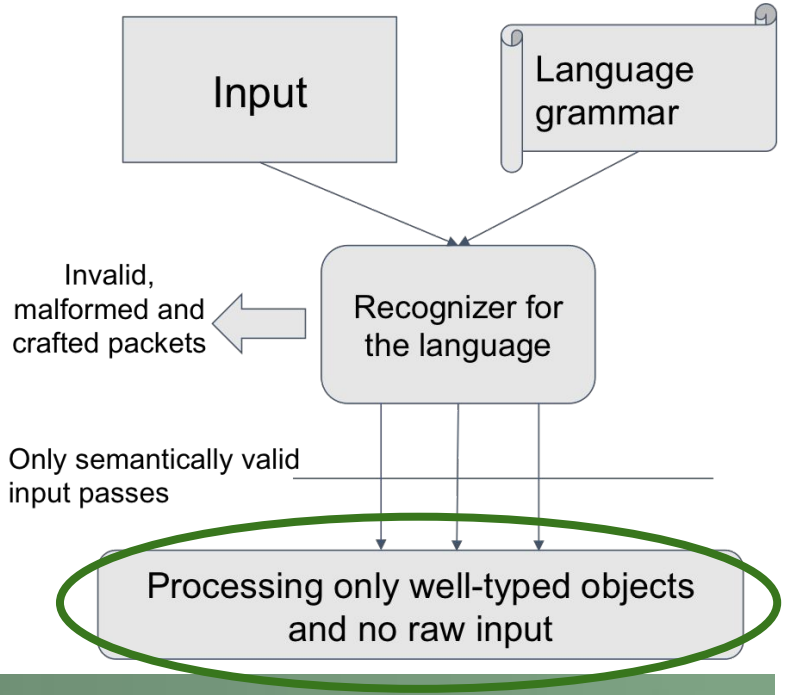
Preventing Zero-days via Crafted Input

- *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.
     - Invocations match the grammar!



Preventing Zero-days via Crafted Input

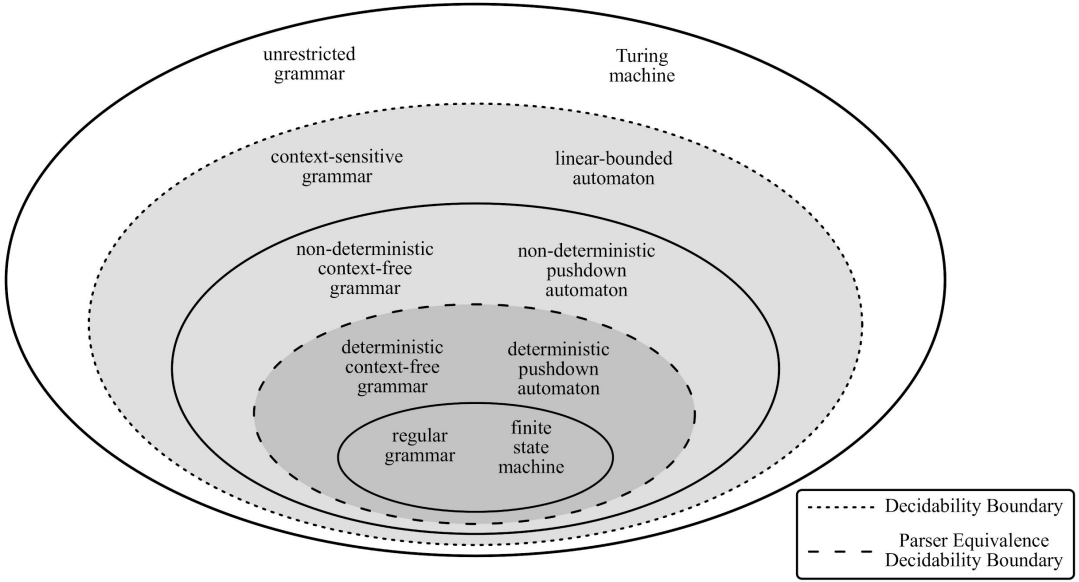
- *Solution: LangSec Parsers*
  1. Define a grammar that represents a "secure subset" of the protocol.
     - No more than context-free!
     - ...or maybe PEG
  2. Build a parser that accepts *only* this grammar.
     - Invocations match the grammar!
  3. Use this parser *everywhere* this protocol is parsed.



Context-Free Grammars

Defined as a 4-tuple: $G = (V, \Sigma, R, S)$

- V: Finite set of non-terminals
- Σ: Finite set of terminals (disjoint from V)
- R: Set of productions defined by
  $V \times (V \cup \Sigma)^*$
  * is the kleene-star here.



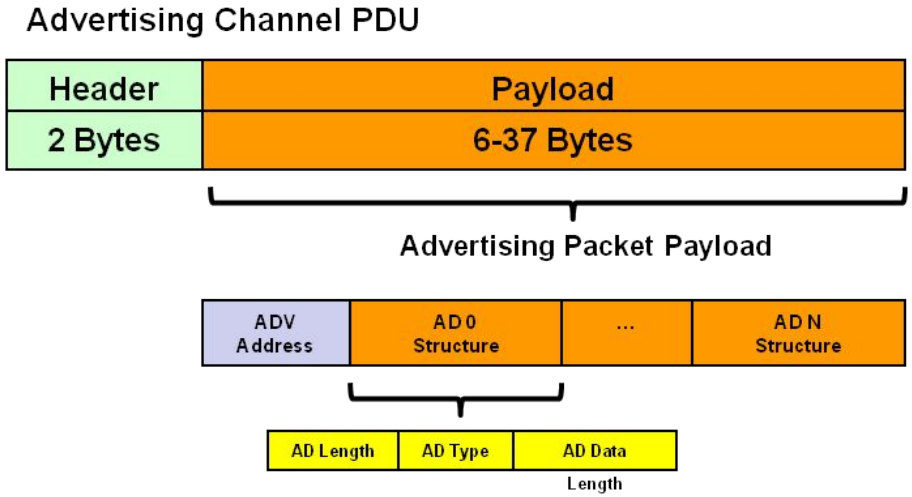
An introduction to PEGs

- No ordered choices like CFGs.
- *&* and *!* are additional operations allowed on PEGs. These operations do not consume input.
- PEGs are not ambiguous like CFGs.
- You can express languages such as $a^n b^n c^n$ using PEGs. How?
  - Hint: use the *&*

Empty	ϵ
Failure	f
Any	$any(p)$
Terminal	c
Concatenation	$n_j n_k$
Ordered Choice	n_j / n_k
Check	$\&n_j$
Negation	$!n_j$

None of the traditional formal-language semantics are enough for real world formats

pos	size	type	id
0	2	u2→ObjType	e_type
2	2	u2→Machine	machine
4	4	u4	e_version
8	...	switch (_root.bits)	entry_point
...	...	switch (_root.bits)	program_header_offset
...	...	switch (_root.bits)	section_header_offset
...	4		flags
...	2	u2	e_ehsize
...	2	u2	program_header_entry_size
...	2	u2	qty_program_header
...	2	u2	section_header_entry_size
...	2	u2	qty_section_header
...	2	u2	section_names_idx



Bluetooth

Supporting Real-World Formats

- Predicates
- Length Fields
- Repeat Fields
- Jump

Approaches:

- Symbolic Register Automata
- **Data Dependent Grammars**

Finite-cost Symbolic Register Automata

- DFA + finite registers + transitions on predicates instead of symbols.
- Counters can be incremented only once.
  - Decremented afterwards.
  - Test for counter == 0
- Unidirectional head movement.
- Allows us to prove equivalence!
- Have not been used for parsing before.

Definition 1 *A DooSRA automata is an extended Symbolic Register Automata R that can be defined as an 6-tuple $(R, Q, q_0, v_0, F, \Delta)$, where R are the finite set of registers, Q is the finite set of states, q_0 is the state state, v_0 is the initial cofiguration of registers, F is the final state, Δ is the set of transitions, where each transition can be defined as:*

$$p \xrightarrow{\varphi, R_{op}/E, I, U, R} q$$

Finite-cost Symbolic Register Automata

- Uses Frama-C to implement invariants, preconditions and postconditions.
- Suited for streaming protocols as well, we only do *O(1)* work when every byte is received.
- Runs in *O(n)* time

```
/*@ requires TAPE_LEN >= 0;
 *  requires \valid(TAPE+(0..(TAPE_LEN-1)));
 *  requires \valid(p.fsm_table+(0..(p.table_lengt
 *  requires cur_byte_pos < TAPE_LEN;
 *  assigns cur_byte_pos, cur_bit_pos;
 */
bool get_next_bit(const char* TAPE, const int TAPE
    bool cur_symbol;

    assert(cur_byte_pos < TAPE_LEN);
    char cur_char = TAPE[cur_byte_pos];
    cur_symbol = 1 & (cur_char >> cur_bit_pos);
    printf("Read %d from tape at loc %d\n",
           cur_symbol,
           cur_byte_pos*8+cur_bit_pos);
    cur_bit_pos++;
    if (cur_bit_pos >= 8) {
        cur_bit_pos = cur_bit_pos % 8;
        cur_byte_pos++;
    }
    return cur_symbol;
}
```

Do we need a new Data Description Language?

- Kaitai (kaitai.io) does implement most of these constructs, but:
  - It does not implement any particular parsing algorithm.
  - Uses a non-traditional way of specifying choices
  - Does not use any module system

```
curve_type:
  seq:
    - id: reserved
      contents: [0x00, 0x00, 0x00, 0x00]
    - id: number_of_entries
      type: u4
    - id: curve_values
      type: u4
      repeat: expr
      repeat-expr: number_of_entries
      if: number_of_entries > 1
    - id: curve_value
      type: u1
      if: number_of_entries == 1
```

Parsley Language

- We build on Attribute Grammars and Data-Dependent Grammars

- If we wanted to accept $a^n b^n c^n$

S -> APlus BPlus CPlus

APlus -> a APlus / ε

BPlus -> b BPlus / ε

CPlus -> b CPlus / ε

Parsley Language

- Adding attributes:
  - These names cannot overlap due to the way our type system is setup.
  - We support int, string, byte strings, floats.

```
S       -> APlus BPlus CPlus

APlus ap {size_a: int} -> a APlus / ε

BPlus bp {size_b: int} -> b BPlus / ε

CPlus cp {size_c: int} -> b CPlus / ε
```

Parsley Language

- Dependent variables
  - A variable is created with scope only within this production. The variable takes the type of the nonterminal.
  - Syntax useful in cases such as S -> AA
- Assigning Attributes
  - We need to assign all the attributes initialized.
  - In case of ε,we assign size to be 0. In all other cases we add the previous size.

```
S       -> APlus BPlus CPlus
APlus ap {size_a: int} -> a ap1=APlus
                {ap.size_a := ap1.size_a + 1}
                / ε  {ap.size_a := 0}

BPlus bp {size_b: int} -> b bp1=BPlus
                {bp.size_b := bp1.size_b + 1}
                / ε {bp.size_b := 0}

CPlus cp {size_c: int} -> b cp1=CPlus
                {cp.size_c := cp1.size_c + 1}
                / ε {cp.size_c := 0}
```

Parsley Language

- Adding constraints
  - These are boolean functions.
  - We can move them to a user-defined function.
  - S does not need any assignments here.

```
S       -> a=APlus b=BPlus c=CPlus
             [a.size_a = b.size_b &&
                 a.size_a = c.size_c] ;;
APlus ap {size_a: int} -> a ap1=APlus
                 {ap.size_a := ap1.size_a + 1}
                 / ε  {ap.size_a := 0} ;;

BPlus bp {size_b: int} -> b bp1=BPlus
                 {bp.size_b := bp1.size_b + 1}
                 / ε {bp.size_b := 0} ;;

CPlus cp {size_c: int} -> b cp1=CPlus
                 {cp.size_c := cp1.size_c + 1}
                 / ε {cp.size_c := 0}
```

Let us try to parse “aabbcc” using our algorithm

	a	**a**	**b**	**b**	**c**	**c**	**ε**
S -> APlus BPlus CPlus							
APlus -> A APlus / **ε**							
BPlus ->B BPlus / **ε**							
CPlus ->C CPlus / **ε**							
A -> a							f
B -> b							f
C -> c							f

Let us try to parse “aabbcc” using our algorithm

	a	**a**	**b**	**b**	**c**	**c**	**ε**
S -> APlus BPlus CPlus							
APlus -> A APlus / ε							0
BPlus ->B BPlus / ε							0
CPlus ->C CPlus / ε							0
A -> a							f
B -> b							f
C -> c							f

Let us try to parse “aabbcc” using our algorithm

	a	**a**	**b**	**b**	**c**	**c**	**ε**
S -> APlus BPlus CPlus							0
APlus -> A APlus / ε							0
BPlus ->B BPlus / ε							0
CPlus ->C CPlus / ε							0
A -> a							f
B -> b							f
C -> c							f

Let us try to parse “aabbcc” using our algorithm

	a	a	b	b	c	c	ε
S -> APlus BPlus CPlus							0
APlus -> A APlus / ε							0
BPlus ->B BPlus / ε							0
CPlus ->C CPlus / ε						1	0
A -> a						f	f
B -> b						f	f
C -> c						1	f

Let us try to parse “aabbcc” using our algorithm

	a	a	b	b	c	c	ε
S -> APlus BPlus CPlus						f	0
APlus -> A APlus / ε						0	0
BPlus ->B BPlus / ε						0	0
CPlus ->C CPlus / ε						1	0
A -> a						f	f
B -> b						f	f
C -> c						1	f

Let us try to parse “aabbcc” using our algorithm

	a	a	b	b	c	c	ε
S -> APlus BPlus CPlus					f	f	0
APlus -> A APlus / ε					0	0	0
BPlus ->B BPlus / ε					0	0	0
CPlus ->C CPlus / ε					2	1	0
A -> a					f	f	f
B -> b					f	f	f
C -> c					1	1	f

Let us try to parse “aabbcc” using our algorithm

	a	**a**	**b**	**b**	**c**	**c**	**ε**
S -> APlus BPlus CPlus			f	f	f	f	0
APlus -> A APlus / ε			0	0	0	0	0
BPlus ->B BPlus / ε			2	1	0	0	0
CPlus ->C CPlus / ε			0	0	2	1	0
A -> a			f	f	f	f	f
B -> b			1	1	f	f	f
C -> c			f	f	1	1	f

Let us try to parse “aabbcc” using our algorithm

	a	**a**	**b**	**b**	**c**	**c**	**ε**
S -> APlus BPlus CPlus			f	f	f	f	0
APlus -> A APlus / ε			0	0	0	0	0
BPlus ->B BPlus / ε			2	1	0	0	0
CPlus ->C CPlus / ε			0	0	2	1	0
A -> a			f	f	f	f	f
B -> b			1	1	f	f	f
C -> c			f	f	1	1	f

Let us try to parse “aabbcc” using our algorithm

	a	a	b	b	c	c	ε
S -> APlus BPlus CPlus	6	f	f	f	f	f	0
APlus -> A APlus / ε	2	1	0	0	0	0	0
BPlus ->B BPlus / ε	0	0	2	1	0	0	0
CPlus ->C CPlus / ε	0	0	0	0	2	1	0
A -> a	1	1	f	f	f	f	f
B -> b	f	f	1	1	f	f	f
C -> c	f	f	f	f	1	1	f

Parsley on MavLink

```
fun convert(b : [byte]) -> int = {
  (case Int.of_bytes(b) of
   | option::Some(i) -> i
   | option::None()  -> 0)
}

type headertype = {magic: int,
     payload_length:int,
     incompatibility_flag: int,
     compatibility_flag: int,
     packet_sequence: int,
     system_id: int,
     component_id: int,
     message_id: int}
```

```
MavlinkHeader header {headertype} :=
     magic=Byte
     payloadlength=Byte
     incompatibility=Byte
     compatibility=Byte
     packet_sequence=Byte
     system_id=Byte
     component_id=Byte
     message_id=(Byte ^ 3)
     [Int.of_byte(magic) = 253 ||
          Int.of_byte(magic) = 254]
     {
     header.payload_length  :=
          Int.of_byte(payloadlength);
     …
     Header.message_id := convert(message_id)
     }
}
```

Complexity Analysis

- If number of productions is *k* and the input size is *n,* our Parsley parser runs in *O(n.k)* time.
- Jump instruction is still a work in progress. We need to create another chart at the location where we need to start parsing.

```
S -> Jump(offset, size, Production)
```

- There seem to be situations (like in the PDF format) where you do not know the size. How would we deal with such a construct?

Further Reading

- Information about LangSec: http://langsec.org/

- Research Report: The Parsley Data Format Definition Language https://www.cs.dartmouth.edu/~sws/pubs/parsley-langsec2020.pdf

- Papers on SRAs and the Parsley Parser are work-in-progress.

Thank you!

Questions?

https://prashant.at

pa@cs.dartmouth.edu