



```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAB8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 4
sltu $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub 2DA48
```

# Selling LangSec

## Tales from the Alchemist's Apprentice

Felix 'FX' Lindner

*Invent & Verify*

```
move $a0, $t7
lw $a0, dword_35A6C
jal sub_2DAD4
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```



## **My Story with LangSec**

Spirits that I've cited  
My commands ignore

```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DA88
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $a0, 1
srlv $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub $t9, $t9
```

How a blind chicken found a corn

# FORMAT NORMALIZATION

```
move $a1, $v0
lw $a0, $a1
jal sub_2DA04
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($1)
sw $v0, dword_35A6C
```

*Invent & Verify*



# Blitzableiter

```

addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAEB
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1
stwu $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub $total

```

- Adobe Flash SWF Format defense layer
  - Read, Recognize, Remove, Rebuild
  - Support for AVM1 and AVM2 code
- Released under GPLv3 in 2010
  - Successfully prevents all exploits since release, without a single change, iff their class was in scope
    - Predictable and deterministic defense component
    - As a byproduct, vulnerabilities exploiting bug classes not covered are significantly harder to exploit
  - Output quality above all of Adobe's own tools, according to Adobe's own tests
  - Fully integrated for firewall and proxy use (ICAP)
  - Fully integrated in NoScript for Firefox

```

move $a1, $v0
lw $a6, dword_35A6C
jal sub_2DA04
addiu $a1, $v0, 2
beqz $v0, loc_2DA44
move $v0, $a1
la $1, dword_35A6C
lw $t1, dword_35A6C
lw $t0, 0($1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($1)
sw $v0, dword_35A6C

```



# #FAIL

```

addiu $sp, -0x18
sw    $ra, 0x18+var_4($sp)
sw    $a0, 0x18+arg_0($sp)
lui   $i, 3
jal   sub_2DAE8
lw    $a0, dword_35A6C
lui   $i, 3
lw    $t7, dword_35A6C
lw    $t6, dword_35A70
subu  $t8, $t6, $t7
addiu $t9, $t8, 1
stwu  $t1, $v0, $t9
beqz  $t1, loc_2DA24
nop
sub 70000

```

- Nobody ever uses Blitzableiter
- Papers and talks to explain the concept
  - People are nodding and don't get it
- Decades of conditioning by the detection paradigm industry
  - anti-virus, IDS, IPS and all the other expensive ways to get pwned
- Failed to overcome the "scanning" notion

*Invent & Verify*



```

move  $a1, $t7
lw    $a6, dword_35A6C
jal   sub_2DA04
addiu $a1, $v0, 0x10
beqz  $v0, loc_2DA44
move  $v0, $t9
la    $t1, dword_35A6C
lw    $t0, 0($t1)
subu  $t2, $t0, $t1
sra   $t3, $t2, 2
sll   $t4, $t3, 2
addu  $t5, $v0, $t4
sw    $t5, 0($t1)
sw    $v0, dword_35A6C

```

© 2014 seemikedraw.com.au



Somewhere in Nigeria

**How That Felt Like**

```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAE8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1
stwu $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub $total
```

And the apprentice said to the alchemist:  
"Gods creation is so rich and things interact with so much complexity, he must have used Turing-complete machines to build it."

The alchemist thought about that and answered: "No, young padawan. What he used was mathematics."

*Invent & Verify*



```
move $a1, $a0
lw $a1, dword_35A6C
jal sub_2DAE8
addiu $a1, $a1, 1
beqz $v0, loc_2DA24
move $v0, $0
la $1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

# Terminology #FAIL

## When People Hang Up On You

- **Language-theoretic Security**
  - Impossible to convey how practical LangSec is if the recipient repudiates theory
  - In theory, theory and practice are the same. In practice, they are not.
    - Let's call it Language-practical Security ;)
- **Formally verified** is tainted with #FAIL
  - Exception: verification people 😊
- **Automaton** is not a word for many people
- **Grammar** reminds people of their ugly Russian language teachers at school

```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAEE
lw $a0, dword_35A6C
lui $1, 3
sw $t7, dword_35A6C
sw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8
stwu $1, $v0, $t9
beqz $1, loc_2DA24
nop
```

*Invent & Verify*



```
move $t0, $t1
lw $t1, sub_2DA64
addiu $t1, $t0, 0x10
beqz $t0, $t1, loc_2DA70
move $v0, $t0
la $t1, dword_35A70
lw $t1, $t1
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```



# Helmbrecht's Law

- Whoever uses a car-comparison or crash testing analogy in a discussion about software security, is immediately wrong.



```

addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $i, 3
jal sub_2DAEB
lw $a0, dword_35A6C
lui $i, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1
andiu $t9, $t9, 0x1

```

```

move $a1, $t7
lw $a6, dword_35A6C
jal sub_2DA64
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $i, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($i)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($i)
sw $v0, dword_35A6C

```



```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAE8
$a0, dword_35A6C
$1, 3
$e7, dword_35A6C
$e6, dword_35A70
$e8, $e6, $e7
$e9, $e8
$1, $e9, $e9
$1, loc_2DA24
sub $total
```



## Jason Knowles

More: [Bio](#), [Facebook](#), [Twitter](#), [News Team](#)

April 29, 2014 (WLS) -- The ABC7 I-Team is learning more about the "heartbleed virus" and how you can be a victim without even knowing it.

The popular software used to protect your personal information has been compromised, but that threat is confusing some people.

This not something you should blow off.

### Related Content

MORE: [E-mail the I-Team](#)

MORE: [LIKE ABC7Chicago on Facebook](#)

MORE: [Follow ABC7Chicago on Twitter](#)

Technology experts say that the "Heartbleed Virus" is widespread and hit 66 percent of all websites over the past two years.

So the I-Team wanted to know what people are doing to stop the bleeding.

Heartbleed. The term comes from the communication between two so-called "hearts" on a server which verify your security as you shop, check e-mails and bank statements. There is now a backdoor break-in between those hearts, and it's bleeding.

*Invent & Verify*



```
move $t4, $t3, 2
addiu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

```
addiu $sp, -0x18  
sw $ra, 0x18+var_4($sp)  
sw $a0, 0x18+arg_0($sp)  
lui $1, 3  
jal sub_2DA88  
lw $a0, dword_35A6C  
lui $1, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t9, $t8, 1  
stwu $1, $v0, $t9  
beqz $1, loc_2DA24  
nop  
sub $t9, $t9
```

That sounds not very agile!

## MAKING BUSINESS CASES

```
move $a1, $v0  
lui $a1, 0x10  
jal sub_2DA04  
addiu $a1, $v0, 0x10  
beqz $v0, loc_2DA44  
move $v0, $0  
la $1, dword_35A70  
lw $t1, dword_35A6C  
lw $t0, 0($1)  
subu $t2, $t0, $t1  
sra $t3, $t2, 2  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($1)  
sw $v0, dword_35A6C
```

*Invent & Verify*



# The Sorcerer's Apprentice Develops an API

```

addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAE8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1
stru $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub $total

```

That old alchemist has vanished  
 And for once has gone away!  
 Spirits called by him, now banished,  
 My commands shall soon obey.  
 Every step and saying  
 That he used, I know,  
 And with web sockets obeying  
 My arts I will show.

Stop now, hear me!  
 Ample measure  
 Of your treasure  
 We have gotten!  
 Ah, I see it, dear me, dear me.  
 LangSec's word I have forgotten!

```

move $a1, $t7
lw $a6, dword_35A6C
jal sub_2DA04
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($1)
sw $v0, dword_35A6C

```

*Invent a verify*



# The Sorcerer's Apprentice Develops an API

```

addiu $sp, -0x18
sw    $ra, 0x18+var_4($sp)
sw    $a0, 0x18+arg_0($sp)
lui   $i, 3
jal   sub_2DAE8
lw    $a0, dword_35A6C
lui   $i, 3
lw    $t7, dword_35A6C
lw    $t6, dword_35A70
subu  $t8, $t6, $t7
addiu $t9, $t8, 1
stwu  $t1, $v0, $t9
beqz  $t1, loc_2DA24
nop
sub  Total

```

**The bug returns, more 0day dragging!  
 Now I'll throw myself upon you!  
 Soon, 0 goblin, you'll be sagging.  
 Crash! This sharp patch has undone you.  
 What a good blow, truly!  
 There, it's fixed, I see.  
 Hope now rises newly,  
 And my breathing's free.**

**Woe betide me!  
 New bugs scurry  
 In a hurry,  
 Rise like towers  
 There beside me.  
 Help me, help, oh LangSec powers!**

*Invent a verify*



```

move  $a6, $t7
lw    $a6, dword_35A6C
jal   sub_2DA64
addiu $a1, $v0, 0x10
beqz  $v0, loc_2DA44
move  $v0, $0
la    $i, dword_35A70
lw    $t1, dword_35A6C
lw    $t0, 0($i)
subu  $t2, $t0, $t1
sra   $t3, $t2, 2
sll   $t4, $t3, 2
addu  $t5, $v0, $t4
sw    $t5, 0($i)
sw    $v0, dword_35A6C

```

# The Stability Argument

- Business suffers from large costs when APIs need to be changed
  - Example: Apple iOS address book API
- API changes for security are even worse
- Stable APIs increase agility of development for the hipster apps
- Unstable APIs cause an explosion of development cost
  - Ask Fabs about the Linux Kernel



```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAE8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1
stwu $t1, $v0, $t9
loc_2DA24
sub $t10, $t10, 1
```

```
move $t0, $v0
lw $a0, dword_35A70
jal sub_2DAE8
addiu $a1, $v0, 1
beqz $v0, $t0
move $t0, $v0
la $t1, dword_35A70
lw $t1, dword_35A70
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

# The Generator Argument

- Grammars scare people
  - Generating code automatically makes them wanting grammars
- Product Houses in enterprises know the cost of not having a grammar
  - Hence XML
- Providing a process to agree on interface grammars is providing a strong business incentive to the people that matter
  - If you save someone money, they don't ask how you did that, they just do as you tell them

```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
sub_2DAE8
$a0, dword_35A6C
$1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8
stru $1, $v0, $t9
beqz $1, loc_2DA24
nop
```

```
move $a1, $v0
lw $a0, $v0
jal sub_2DAE8
addiu $a1, $v0, 0x20
beqz $v0, $v0, 0
move $a1, $v0
la $1, dword_35A6C
lw $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```



# The Agreement Machine

- Given a set of strong data types...
- For each product group...
  1. List information to handle
  2. Strongly type all fields
  3. Send to all product groups
  4. Request signoff
  5. If not signed off, goto 1
- Generate code for all product groups: Nail it

1	Name	var String
---	------	------------

2	SpargelSize	var int
---	-------------	---------

1	Name	var UTF8
---	------	----------

2	SpargelSize	var unsigned int
---	-------------	------------------

1	Name	var UTF8
---	------	----------

2	SpargelSize	unsigned int32
---	-------------	----------------

1	Name	UTF8[200]
---	------	-----------

2	SpargelSize	unsigned int32
---	-------------	----------------

Unfortunately, MLP pointed out NP.





# The Agreement Machine Scales

- Standardization Committees have the same problem:  $n^2$
- The standard solution is to agree on disagreement, also known as Vendor Extension
  - IPv6 is a prime example

```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
and $t8, $1, 0x00000008
lwr $t7, dword_35A6C
lwr $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1
stru $1, $v0, $t9
loc_2DA24
sub $t9, $t9
```

```
move $a0, $t7
lwr $a6, dword_35A6C
jal sub_2DA04
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $1, dword_35A70
lwr $t1, dword_35A6C
lwr $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C
```

*Invent & Verify*



# The Liability Argument

```
addiu $sp, -0x18  
sw $ra, 0x18+var_4($sp)  
sw $a0, 0x18+arg_0($sp)  
lui $i, 3  
jal sub_2DAE8  
lw $a0, dword_35A6C  
lui $i, 3  
lw $t7, dword_35A6C  
lw $t6, dword_35A70  
subu $t8, $t6, $t7  
addiu $t9, $t8, 1  
addiu $t9, $t9, 1
```

- Lack of Product Liability is the root cause
  - The business case is in selling broken software
- Liability will come back in one form or another
  - Regulation is liability
  - Lean back and watch me go
- Liability flips the business case towards selling working software
- LangSec allows small components to be composed safely
  - Agile trust architectures deal with the authorization



*Invent & Verify*

```
move $a1, $t7  
lw $t6, 0($t1)  
jal sub_2DAE8  
addiu $t1, $t1, 1  
beqz $t1, $t0, $t1  
move $t1, $t0  
lw $t1, dword_35A70  
lw $t1, dword_35A70  
lw $t0, 0($t1)  
subu $t2, $t0, $t2, 2  
sll $t4, $t3, 2  
addu $t5, $v0, $t4  
sw $t5, 0($t1)  
sw $v0, dword_35A6C
```

```
addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAE8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $a0, 1
srlv $1, $v0, $t9
beqz $1, loc_2DA24
nop
sub $t9, $t9
```

Dear Academia, I have some wishes for Xmas ...

## WHAT WE NEED NOW

```
move $a1, $v0
lw $a6, sub_2DA04
jal sub_2DA04
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la $1, dword_35A70
lw $t1, dword_35A6C
lw $t0, 0($1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($1)
sw $v0, dword_35A6C
```

*Invent & Verify*



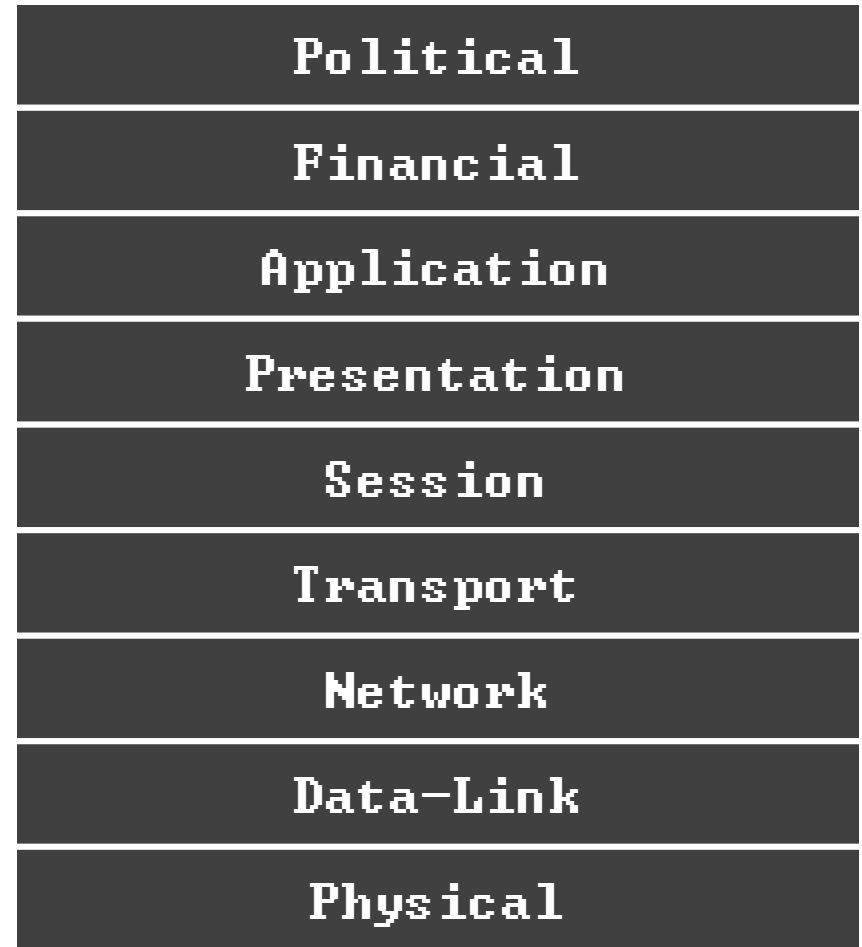
# The Stack Is A Lie

```

addiu $sp, -0x18
sw $ra, 0x18+var_4($sp)
sw $a0, 0x18+arg_0($sp)
lui $1, 3
jal sub_2DAE8
lw $a0, dword_35A6C
lui $1, 3
lw $t7, dword_35A6C
lw $t6, dword_35A70
subu $t8, $t6, $t7
addiu $t9, $t8, 1

```

- Systems are composed, not stacked
- Composition works best when the properties of the instruments are known
  - Sound properties
  - Physical properties
- We need a standard reference model:



## The LangSec Orchestra

*Invent & Verify*



```

move $t0, $a0
lw $a1, 0($t0)
jal sub_2DAE8
addiu $a1, $t0, 1
beqz $v0, $t0, $t0
move $v0, $t0
la $t1, dword_35A6C
lw $t0, 0($t1)
subu $t2, $t0, $t1
sra $t3, $t2, 2
sll $t4, $t3, 2
addu $t5, $v0, $t4
sw $t5, 0($t1)
sw $v0, dword_35A6C

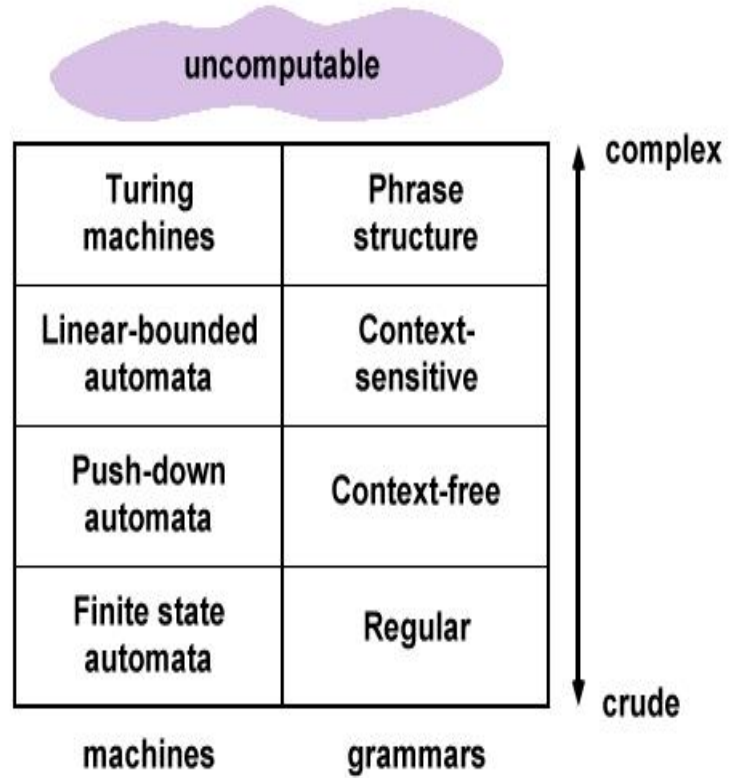
```

# Language Complexity Proof

```

addiu $sp, -0x18
sw    $ra, 0x18+var_4($sp)
sw    $a0, 0x18+arg_0($sp)
lui   $i, 3
      sub_2DAE8
      $a0, dword_35A6C
      $i, 3
lw    $t7, dword_35A6C
lw    $t6, dword_35A70
subu  $t8, $t6, $t7
addiu $t9, $t8, 1
      $t9, $t9, $t9
      24
    
```

- Andreas Bogk's Proof Engineering request seconded
- The Agreement Machine only works if we can automatically and reliably tell language complexity



```

move $a1, $t7
lw   $a0, $t6
jal  sub_2DA44
addiu $a1, $v0, 0x10
beqz $v0, loc_2DA44
move $v0, $0
la   $i, dword_35A70
lw   $t1, dword_35A6C
lw   $t0, 0($t1)
subu $t2, $t0, $t1
sra  $t3, $t2, 2
sll  $t4, $t3, 2
addu $t5, $v0, $t4
sw   $t5, 0($t1)
sw   $v0, dword_35A6C
    
```





**To all you war waging retirees  
All you bot herders and runners  
All you Comment Group soldiers  
And FinFishing Simon coward liars**

**To all you listening Stasi dealers  
All you forum egos, trolls and leakers  
All you Data Retention law abiders  
And you C-board data protection liars**

**Form Up and Brace Yourself!**

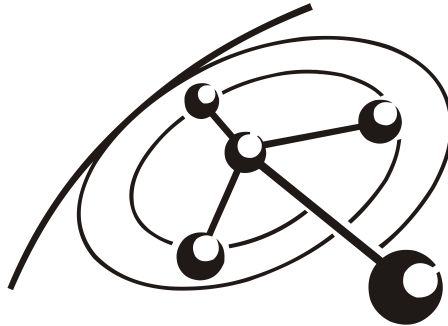
**Then Watch LangSec Ruining Your Day!**

# Thank You!

```

addiu $sp, -0x18
sw    $ra, 0x18+var_4($sp)
sw    $a0, 0x18+arg_0($sp)
lui   $i, 3
jal   sub_2DAE8
lw    $a0, dword_35A6C
lui   $i, 3
lw    $t7, dword_35A6C
lw    $t6, dword_35A70
subu  $t8, $t6, $t7
addiu $t9, $t8, 1
stwu  $t1, $v0, $t9
beqz  $t1, loc_2DA24
nop
sub  $t9, $t9, 1

```



**Recurity Labs**

Felix 'FX' Lindner  
Head

fx@recurity-labs.com

Recurity Labs GmbH, Berlin, Germany  
<http://www.recurity-labs.com>

*Invent & Verify*



```

move  $a1, $t7
lw    $a6, dword_35A6C
jal   sub_2DA04
addiu $a1, $v0, 0x10
beqz  $v0, loc_2DA44
move  $v0, $0
la    $i, dword_35A70
lw    $t1, dword_35A6C
lw    $t0, 0($i)
subu  $t2, $t0, $t1
sra   $t3, $t2, 2
sll   $t4, $t3, 2
addu  $t5, $v0, $t4
sw    $t5, 0($i)
sw    $v0, dword_35A6C

```