

# Code-level Cyber-Security: An overview

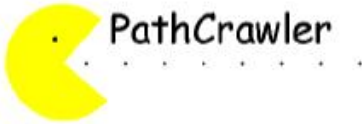
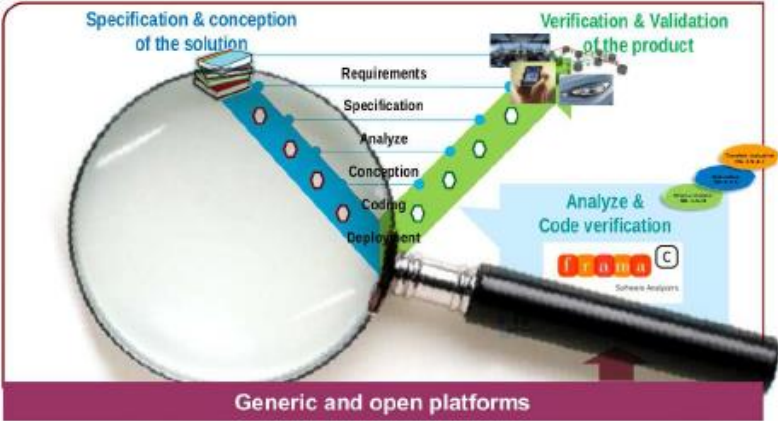
Sébastien Bardin (CEA LIST)  
Richard Bonichon (CEA LIST)



# ABOUT MY LAB @CEA

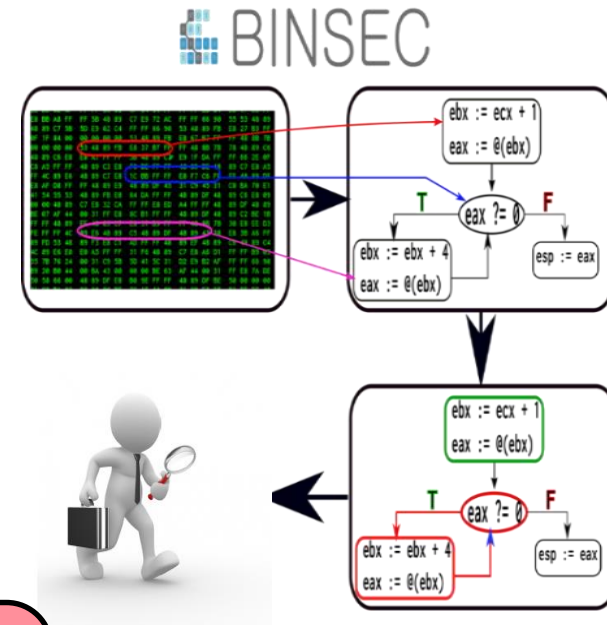
## CEA LIST, Software Safety & Security Lab

- rigorous tools for building high-level quality software
- second part of V-cycle
- automatic software analysis
- mostly source code



- Interested in designing methods & tools helping to develop very safe/secure systems
- **Technical core**
  - Formal methods, program analysis
  - Logic and automated reasoning
- **Application fields**
  - Security
  - Software engineering

Programming-language oriented  
view of security



## Semantic analysis for binary-level security

Lift methods from source-level safety

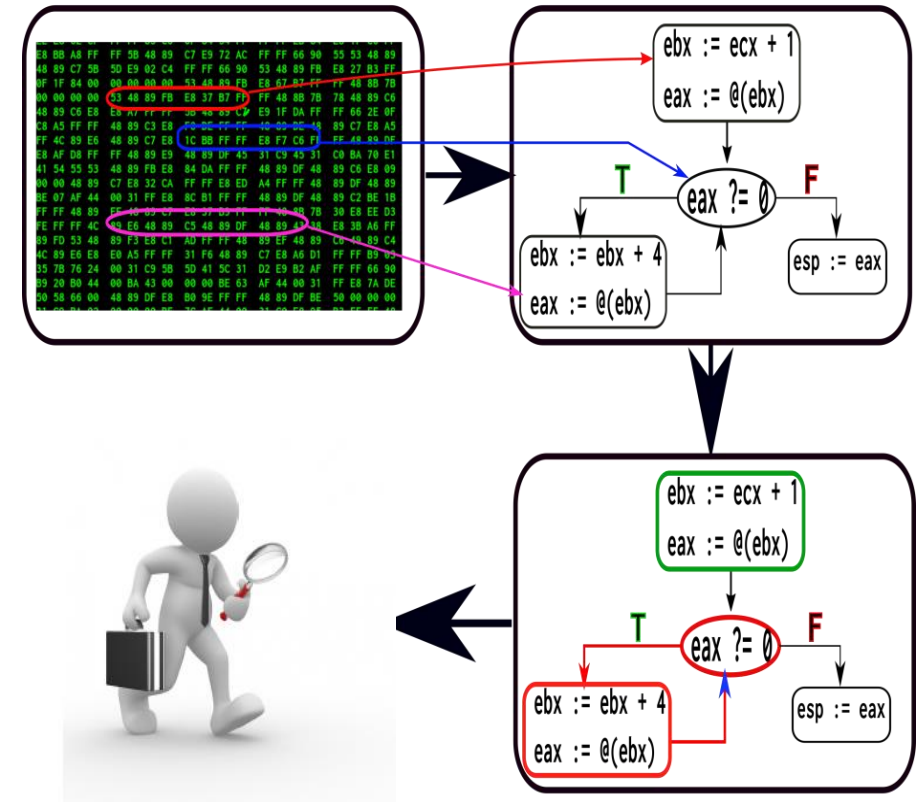
### Some features

- Explore, simplify, prove
- Multi-architecture



Still very young!

 BINSEC



# « Code-level Security » IN A NUTSHELL

- **Goal of the course:**
  - Give an overview of software security
  - Understand that security is not all about crypto (= design-level)
  - Present typical code-level attacks & defenses
- **Covered:** control-flow hijacking, buffer overflow, obfuscation, reverse, tampering, malware
- **Today: overview + basis of programming language semantic / compilers**

- Preamble
- **Context**
- The security game
- Some attacks
- Whole course overview
- There is still hope! (building secure systems)
- Conclusion

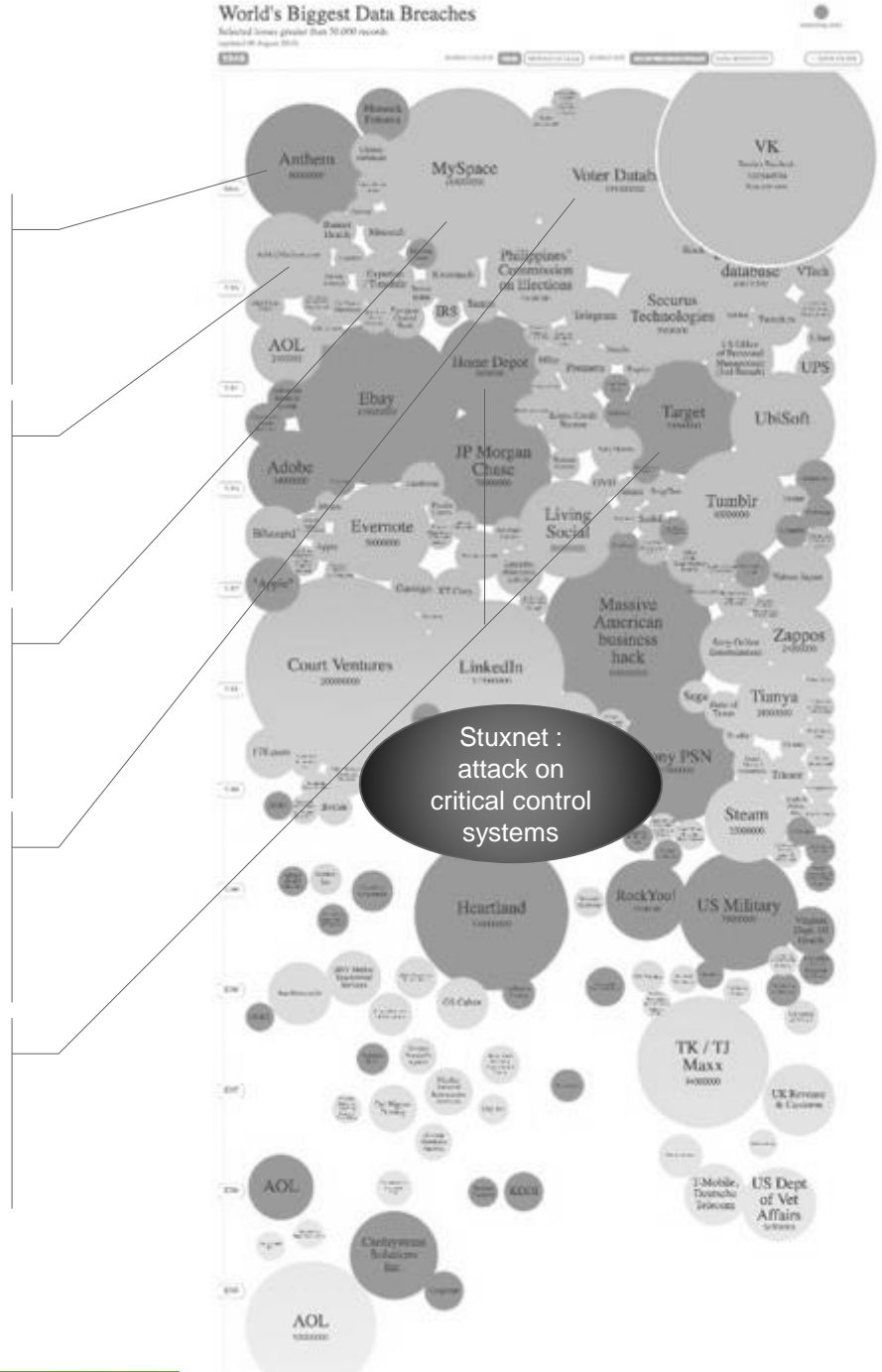
Leak of personal health insurance from **weakly-protected database**

**Privacy breach** on an online dating site

Leak of **unprotected** user credentials and passwords

Security researcher discovers **exposed cloud-based database** of US voters.

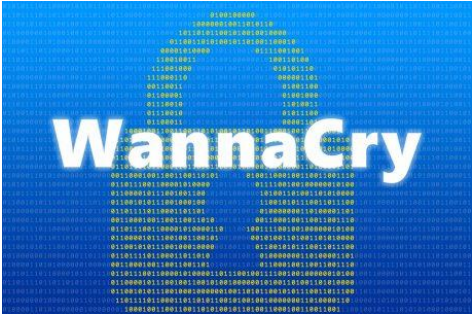
Attacks compromise an HVAC system, install **malware** and exfiltrate payment information **without being detected**





# 2017: THE YEAR OF THE RANSOMWARE

- Real ransomware



- Fake ransomware





APT: highly sophisticated attacks

- **Targeted malware**
- **Written by experts**
- Attack: 0-days
- Defense: stealth, **obfuscation**
- **Sponsored by states or mafia**

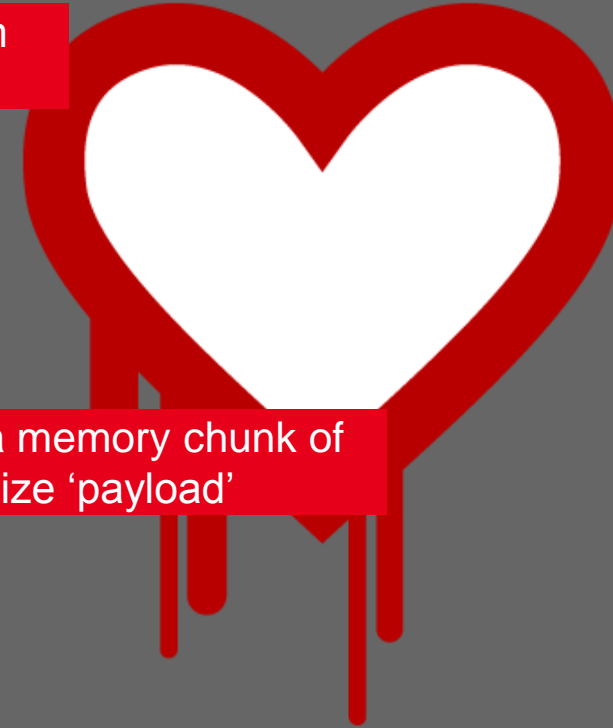


## An older state-level attack: stuxnet



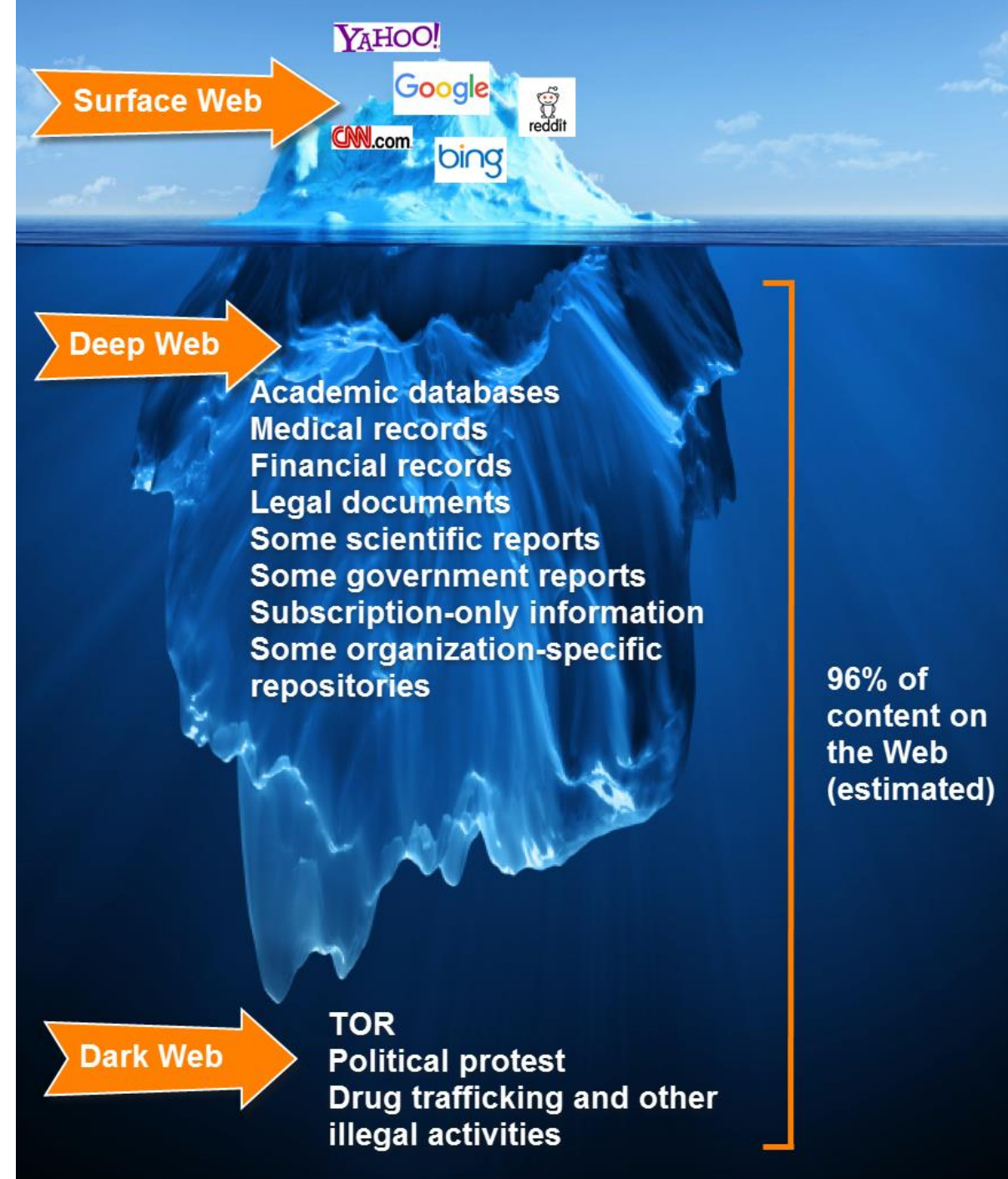
Open-source  
cryptographic library

```
2552 #ifndef OPENSSSL_NO_HEARTBEATS
2553 int
2554 tls1_process_heartbeat(SSL *s)
2555     {
2556     unsigned char *p = &s->s3->rrec.data[0], *p1;
2557     [...]
2561     /* Read type and payload length:
2562     hbtype = *p++;
2563     n2s(p, payload);
2564     p1 = p;
2565     [...]
2571     if (hbtype == TLS1_HB_REQUEST)
2572     {
2573     [...]
2583     /* Enter response type, length and copy payload */
2584     *bp++ = TLS1_HB_RESPONSE;
2585     s2n(payload, bp);
2586     memcpy(bp, p1, payload);
2587     bp += payload;
2588     /* Random padding */
2589     RAND_pseudo_bytes(bp, padding);
2590     [...]
2591     r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer,
2592     3 + payload + padding);
2593     [...]
2594     if (r >= 0 && s->msg_callback)
2595     s->msg_callback(1, s->version,
2596     TLS1_RT_HEARTBEAT,
2597     buffer, 3 + payload + padding,
2598     s, s->msg_callback_arg);
2599     OPENSSSL_free(buffer);
```

Read 'payload' from  
input packetCopy a memory chunk of  
size 'payload'

# A STRONG INCENTIVE TO BEING BAD

- **Dark & Grey Industry**
  - Exploits for iOS are priced 1.5 M\$
- **Profits**
  - Don't pay
    - software, games, vod, etc.
  - Get money
    - ransomware, blackmail, credit card number
    - bitcoin accounts, id & passport scans, ...
  - Run a business
    - botnet aas, ddos aas, exploitation kits
    - new exploits, ...
- **Also:** state-level actors



# A STRANGE ECOSYSTEM





# The Internet (more or less)



You can turn a computer on. Yay.

You must be really bored.

Either use a proxy, or say hi to the FBI.

- Preamble
- Context
- **The security game**
- Some attacks
- Whole course overview
- There is still hope! (building secure systems)
- Conclusion



# THE GOOD, THE BAD & THE INNOCENT

- **The defender:** try to secure the system
- **The attacker:** try to abuse the system
  - Why: for fun & **profit**
  - How: by **taking advantage of system flaws** [see after]
- **The user:** collateral damage

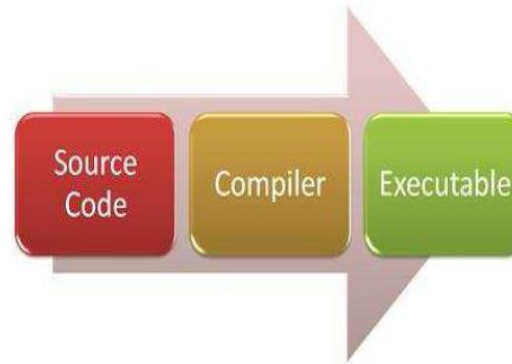
- **Design or implementation**
- Don't we know how to build very safe systems? Yes, but ...
  - Legacy
  - Time-to-market & « add-this-fancy-feature » pressure (web)
  - Cost pressure (embedded systems)
  - And: programming is very complex
  - And: security is harder than safety



```
#include "stdio.h"

long foo(int *x, long *y) {
    *x = 0;
    *y = 1;
    return *x;
}

int main(void) {
    long l;
    printf("%ld\n", foo((int *) &l, &l));
    return 0;
}
```

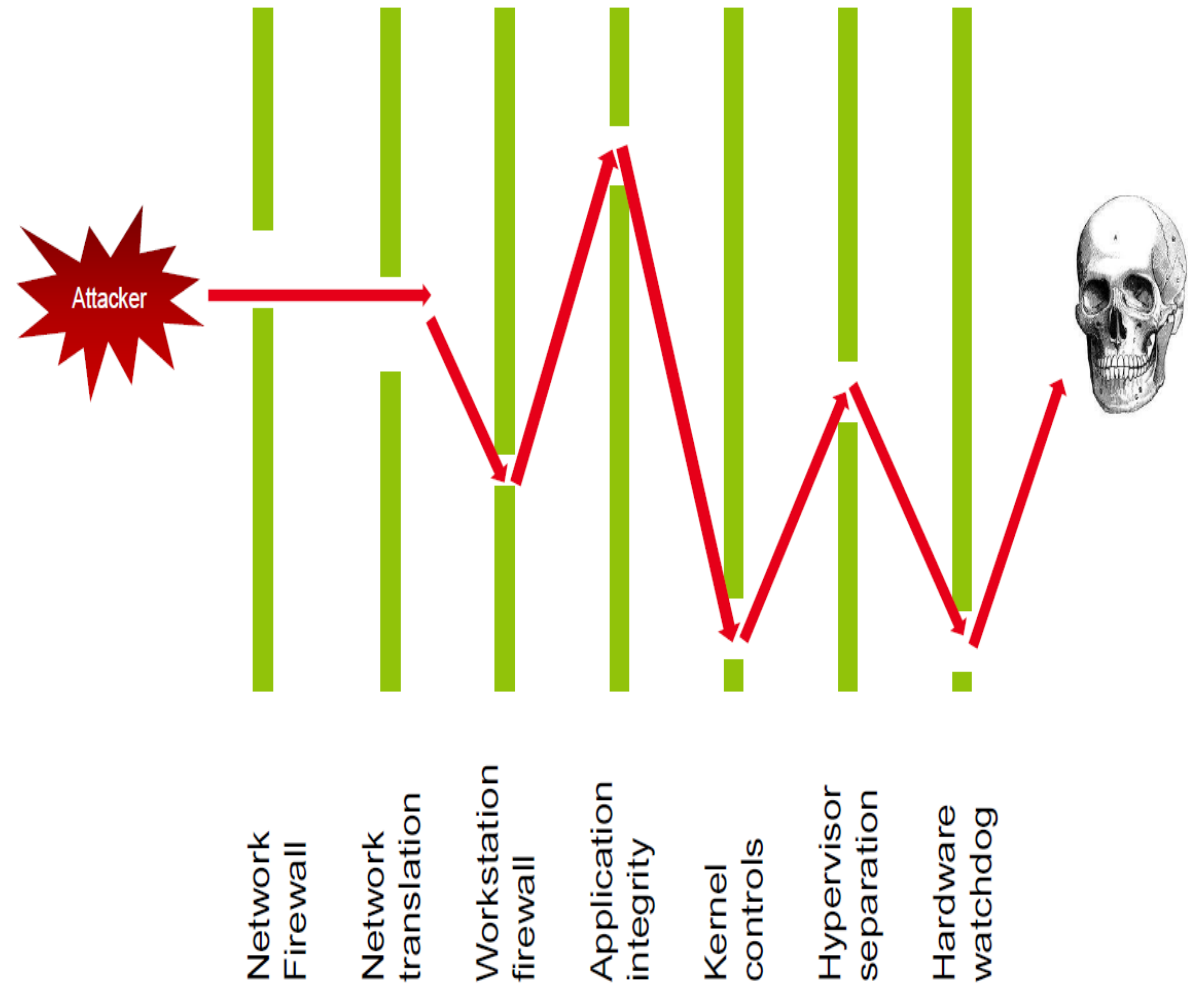
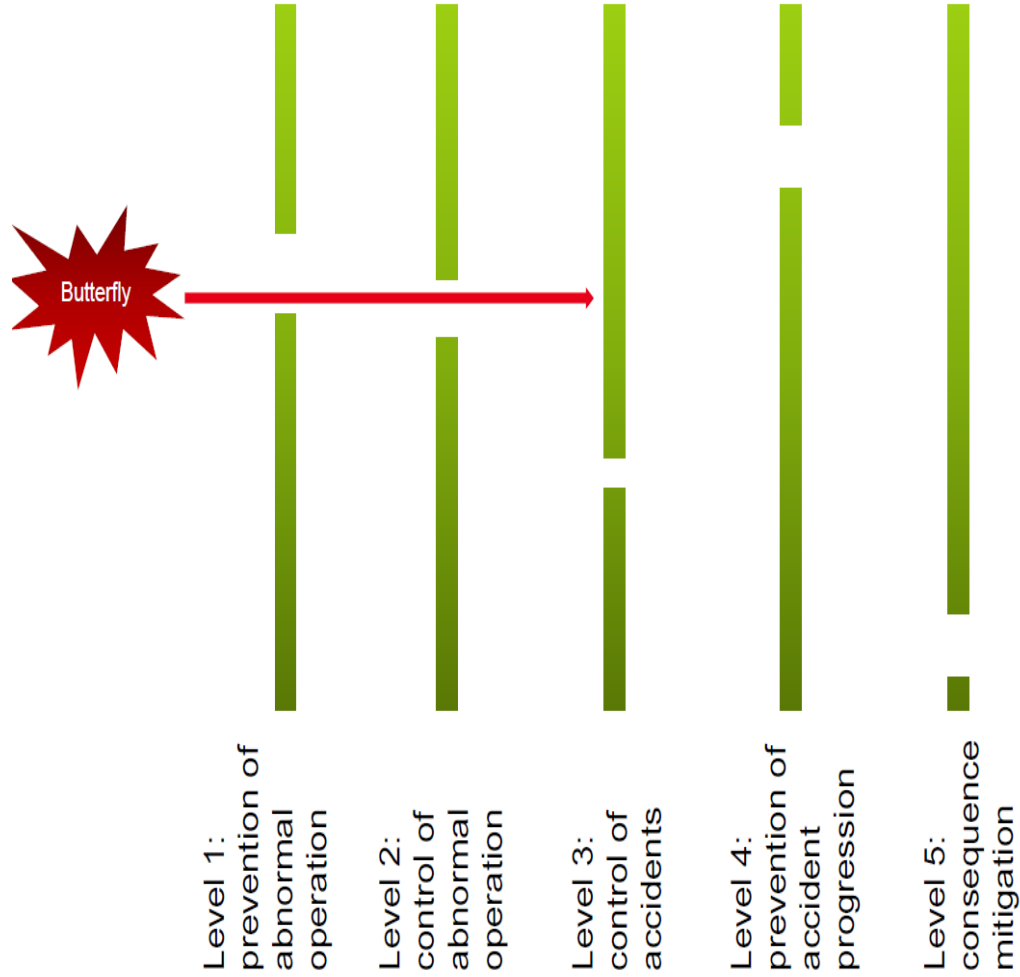


	gcc 7.2.0	clang 5.0
-00	1	1
-01	1	0
-02	0	0
-03	0	0

# SECURITY vs SAFETY

- **Assumption: software correct @ 99.9999999%**
- **Safety: good enough**
  - Nature will not be that nasty
- **Security: not good enough**
  - Attacker may be that nasty!

# SECURITY vs SAFETY



## BY THE WAY

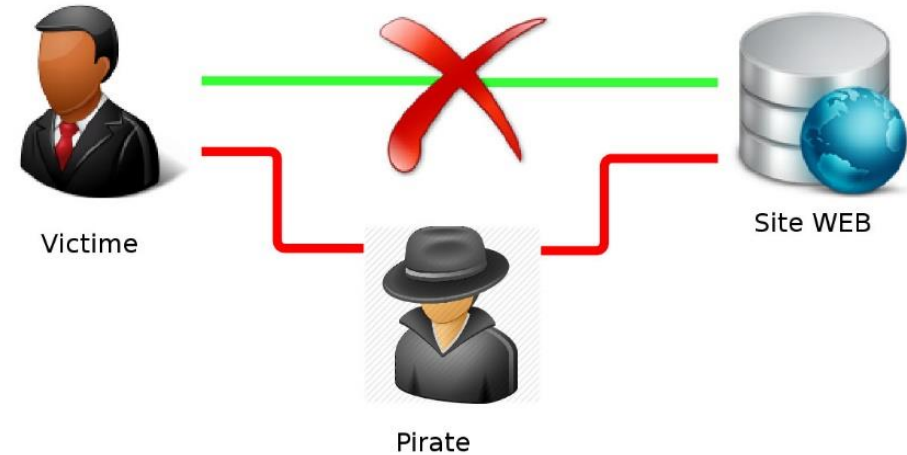
- **Know your enemy**
  - Scriptkiddy: security updates, strong passwords
  - ...
  - Government: hum ...
  
- **Remember: game for profit**
  - No profit → no attacker
  - Just raise the bar enough (ex: video games, vulnerability hunting)
  
- **Duality of security**
  - Exploits → kill your PC or a botnet, spy a terrorist or you
  - Obfuscation → protect IP or ransomware

- **In a few situations, the defender has a clear advantage**
  - The miracle of « provable crypto »
  - Can reveal its method, no efficient way to break it (if well implemented)
- **In most situations: cat-and-mouse game and advantage to attacker**
  - try to be one step ahead
  - raise the bar enough



**MITM: Man-In-The-Middle****Attacker is on the network**

- **Observe messages**
- **Forge messages**



Realm of cryptos

**« Man-Beyond-The-Door »**

**Attacker has limited access**

- **Try to escalate**
- **Forge specially crafted files/queries**



Realm of program analysis

**MATE: Man-At-The-End**

Attacker is *on the computer*

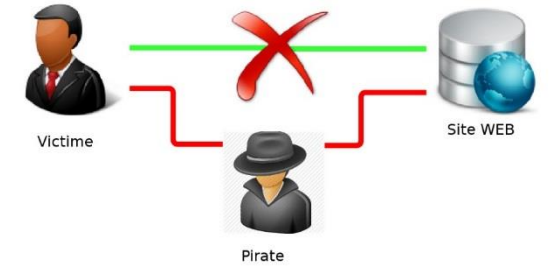
- R/W the code
- Execute step by step
- Patch on-the-fly



Realm of program analysis?  
White-box crypto?



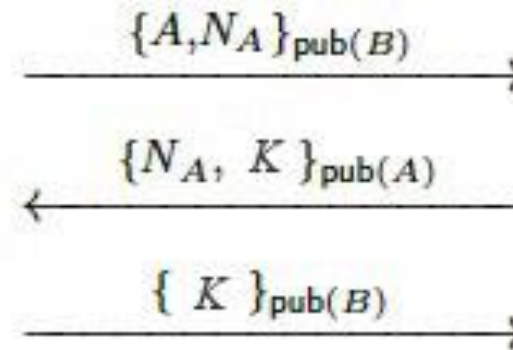
- Preamble
- Context
- The security game
- **Some attacks**
- Whole course overview
- There is still hope! (building secure systems)
- Conclusion



## Needham-Schroeder protocol (1969)

- Exchange key + mutual authentication
- Goal = negotiate a symmetric (private) key for a session

Did you find it?



## Context: asymmetric encryption

- each participant has a public key and a private key
- Public key encodes, private key decodes (perfect crypto)

## Attack by Lowe after 17 years (1986)

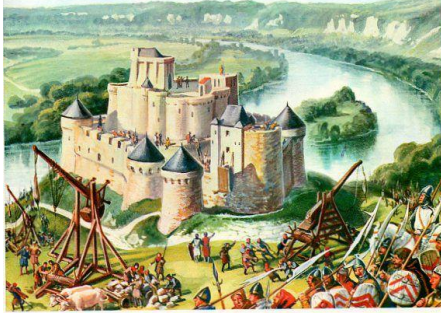
- Even with perfect crypto primitives!
- Bob & Alice both think they talk to each other
- Attacker spies everything

Can be patched!  
<how?>





# SQL INJECTION



A SQL query is one way an application talks to the database.



SQL injection occurs when an application fails to sanitize untrusted data (such as data in web form fields) in a database query.



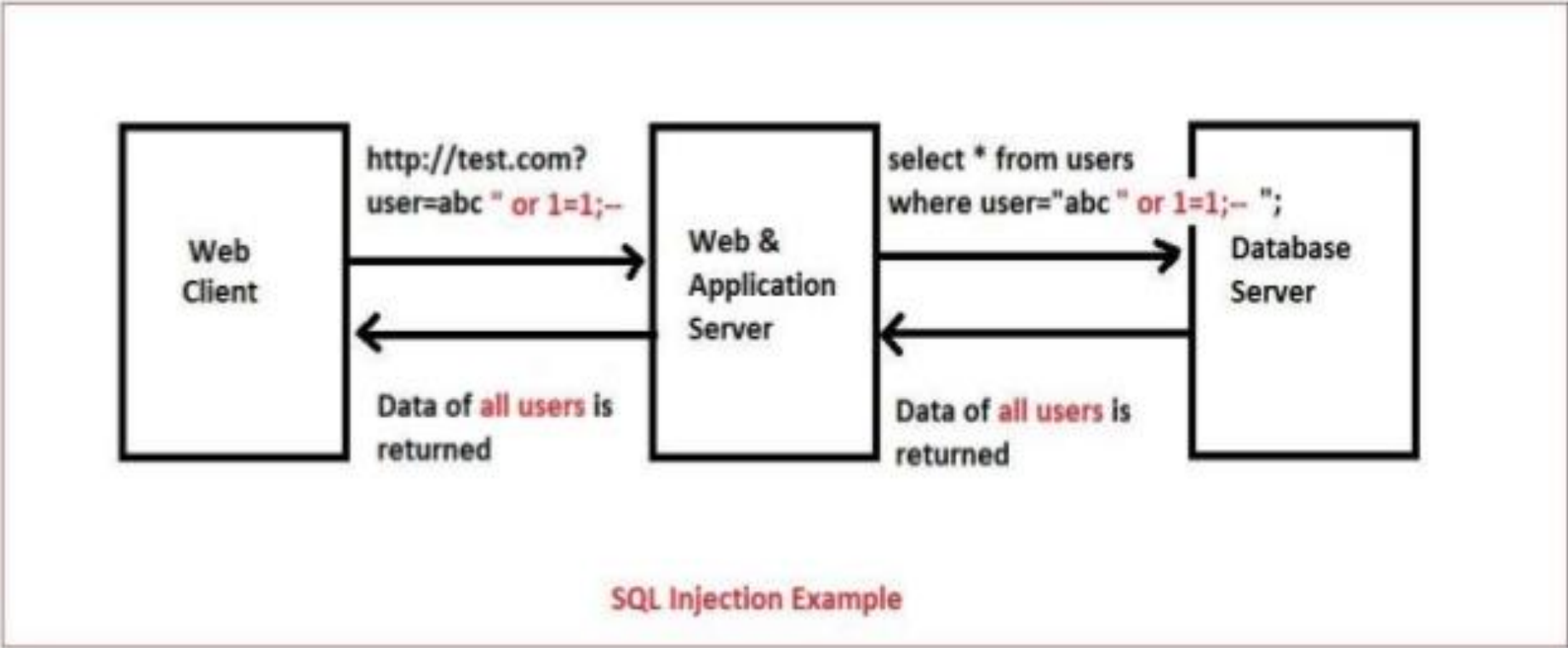
An attacker can use specially-crafted SQL commands to trick the application into asking the database to execute unexpected commands.





# SQL INJECTION (2)

Can be patched!  
<how?>



# CODE TAMPERING



```
char[4] buff,secret;

buff = getInput();
secret = getPassword();
for (i=0 to 3) do
  if(buff[i] != secret[i]) then
    return false;
  endif
endFor
return true;
```

- Preamble
- Context
- The security game
- Some attacks
- **Whole course overview**
- There is still hope! (building secure systems)
- Conclusion

- Overview + basis of language semantics & compilers
- [MBTD] Control-flow integrity: attack
- [MBTD] Control-flow integrity: defense & attack
- [MATE] Obfuscation: basic attacks & defense
- [MATE] Obfuscation: advanced attacks and defense
- xx a bit of everything, including malware xx
- Exam

## « Man-Beyond-The-Door »

Attacker has limited access

- Try to escalate
- Forge specially crafted files/queries



# CONTROL-FLOW INTEGRITY

- **Attacker tries to deviate the execution flow of the program**
  - The typical « buffer overflow » attack
  - Control-flow hijacking
- **Control-flow integrity techniques tries to prevent it, or stop it**
- **Several defenses, and attacks, and defenses, etc.**

## MATE: Man-At-The-End

Attacker is *on the computer*

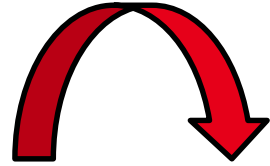
- R/W the code
- Execute step by step
- Patch on-the-fly





**State of the art**

- No usable math-proven solution
- Useful ad hoc solutions (**strength?**)

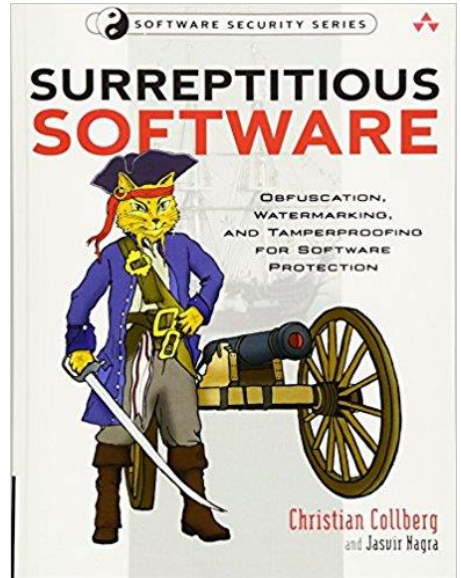


```

    = getStatement();
    sql = "select * from st
    resultSet = statement.executeQu
    if (resultSet.next()) {
        result = true;
        resultSet.getInt("s
        storeId = resu
        storeDescription = resu
        storeType = resu
        storeAdd
    }
    
```

```

ists($NDtKzAWTCQGqUy)}{ $marTuzXmEElrbNr->set_sensitive(False); } } if($jrilcGLMcWbXmi!=1){$HwecPhiIKnsaBY(
boikKUjfvM!=1){ if($CrOorGLihteMbPk=='')$XkLZffvKlHqdYzB=0; switch($CrOorGLihteMbPk) { case 1: $XkLZffvKlHq
urn $AxPGvXMuLrBqSUZ; } function cXBdreLgeQysmbh($ngsHuTaaKlqeKJk){ global $VWgwoCADMvilerx; global $OJfVybOIL
P=$screen_height/$BechHLBAqOgnrXc[1]* $BechHLBAqOgnrXc[0]; } else { $oejysSGfnZAtGQP=$screen_height/$BechHLBA
'ru','2','1','was'); $EQFavHsKCMCIhmV = sqlite_query($NuERFSVleSyVExn, "SELECT lage FROM lage WHERE id=0 "); $
'ru','2','1','was','q'); for ($i = 0; $i <= 8; $i++) { $xBvYwchzFYGttEd=$CrOorGLihteMbPk[$i].'#'; $j++; if($
kTSuioH==''){ $FmZyBrtWLyInYBo}= new GtkRadioButton(null,'',0); $LVUxMyHvkTSuioH=$FmZyBrtWLyInYBo; } else
@QL($image_file){ $ngsHuTaaKlqeKJk=$image_file; $CrOorGLihteMbPk=array('lo','mo','ro','lm','mm','rm','lu','mu
dNg($TBrBtAZPRwFPZYU, $gbeycQSWLKBFFnU, $wVkiGIgBrvOSjt, $zCjJwZmQGNLwmGL) { $fSmyLhWpTfAGQil = imgettfbc
l[1] * $LtcHpLnmFQVedZb - $fSmyLhWpTfAGQil[0] * $LkMbSglLwAjfVfm - $ULabzSbZzHEfrCb; } else { $ULabzSbZzHEfr(
cFCp; $zrxBCrMcVPUjMBo['h']=$KHevYGncDwxvJRF; $zrxBCrMcVPUjMBo['w']=$YUngoXVWLdAOSdJ; return$zrxBCrMcVPUjMBo;
VWcaoJsyxYz-$zrxBCrMcVPUjMBo[1]; if($gbeycQSWLKBFFnU=0){ $iNmEPLiiskpDTlv=-10; }else{ $iNmEPLiiskpDTlv=0; } $iNmE
UrNVTiJdViGHRH=imagesy($WHAB:mHCCyXgNtI)/2- imagesy($maLvSpuqmSzuhJu)/2; If($MwgrEAKeyMnAtiz=='u')$JUAnNBEoXEW
uqmSzuhJu)/2; } If($DugWkYdpKwKJBZ=='r'){ $YogbbPXcrLTDQJZ=imagesx($WHAB:mHCCyXgNtI)- imagesx($maLvSpuqmSzuhJu
QjkvQAhLp['g']; $ooVGdSjSyMSNEjt = $JIQuduQjkvQAhLp['b']; } if($LxbboJGUoNpBGxm=="height"){ $JIQuduQjkvQAhLp =
DaX = 255; } if($ooVGdSjSyMSNEjt>127){ $ooVGdSjSyMSNEjt = 10; } else{ $ooVGdSjSyMSNEjt = 255; } if($TnBeBOHZdYf
EuTvRzGZLGEI=$NDtKzAWTCQGqUy; $TBrBtAZPRwFPZYU = getimagesize($tkoEuTvRzGZLGEI); $qYSGvHLdyejMyI=$TBrBtAZPF
($MeQaCJzkQyKN Azt>imagesx($WHAB:mHCCyXgNtI)/100*$OAZKDtKsRHRgZwB){ $MeQaCJzkQyKN Azt=imagesx($WHAB:mHCCyXgNtI)/
uhJu)-$HLDXcWuyfPoYrFK; If($MwgrEAKeyMnAtiz=='o')$JUAnNBEoXEW RqJm=$HLDXcWuyfPoYrFK; If($MwgrEAKeyMnAtiz=='m')$
($WHAB:mHCCyXgNtI)/2- imagesx($maLvSpuqmSzuhJu)/2; $JUAnNBEoXEW RqJm=imagesy($WHAB:mHCCyXgNtI)/2- imagesy($maLv
$WHAB:mHCCyXgNtI)/2- imagesx($maLvSpuqmSzuhJu)/2; } If($DugWkYdpKwKJBZ=='r'){ $YogbbPXcrLTDQJZ=imagesx($WHAB:mH
->set_text(''); } $TFnsiSsBvFBsDOb=$GLOBALS['BIoUrBpyspeFLWN']; $TFnsiSsBvFBsDOb->set_text(''); $wENZkUTQbQuHs
WMNTlvuSitiM->get_text()." WHERE id=0"); } function XYyCTuPntLfFeeVE(){ global $bpAGFKHBLsZx:Fyb; global $NuERFS
XNGBmCFdvbbmWdK." WHERE id=0"); } function EoNV5gEkqaikLsj($BBVRGSKDxgIVH, $wJfcrfmlBDvDmhp,$ByCzsrSXrtJDP
PLiiskpDTlv->get_text(); if($hvRlKhJmLmhtSzs==0)sqlite_query($NuERFSVleSyVExn, "UPDATE lage SET offset=". $GDw
    
```



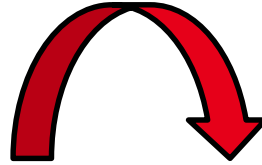
**Transform P into P' such that**

- P' behaves like P
- P' roughly as efficient as P
- P' is very hard to understand

```

ists($NDtKzAWTCQGqUyz)}{$marTuzXmMElrbNr->set_sensitive(False);}}if($ijrilcGLMcVbXmi!=1){$HwecPhiIKnsaBYC
boikUjfvWI=1}{if($CrOorGLihteMbPk=''){$XkLZffvKlHqdyZB=0;switch($CrOorGLihteMbPk){case1:$XkLZffvKlHq
urn$AxPGvXMuLrBqSUZ;}functioncXBdreLgeOysmbh($ngsHuTaaKlqeKJk){global$VW@woCADWVilerx;global$OJfVybOik
P=$screen_height/$BechHLBAqOgnrXc[1]*$BechHLBAqOgnrXc[0];}else{$oejysSGfnZAtGQP=$screen_height/$BechHLBA
'ru','2','1','was');$EQFavHsKCMiMmV=sqli_query($MuERFSVleSyVExn,"SELECTlageFROMlageWHEREid=0");if
'ru','2','1','was','q');for($i=0;$i<=8;$i++){$xBvYwchzFYGttEd=$CrOorGLihteMbPk[$i].'#';$j++;if($C
kTSuioH=''){$FmZyBrtWLyInYBo=newGtkRadioButton(null,'',0);$LVUxMyHvkTSuioH=${$FmZyBrtWLyInYBo};else
gQL($image_file){$ngsHuTaaKlqeKJk=$image_file;$CrOorGLihteMbPk=array('lo','mo','ro','lm','mm','rm','lu','mu
dNg($BrtAZPRwFPZYU,$gbeycQSWLKBFFNu,$WVkiMIgIGbrvOSjt,$zCJjwZmQGNLumGL){$fSmyLhwPTfAGQil=imagettfbb
l[1]*$LtcHpLnmFQvedZb-$fSmyLhwPTfAGQil[0]*$lKmbSgluWAjfvfm-$ULabzSbZzHEfrCb};else{$ULabzSbZzHEfrC
cFCp;$zrxBCrMcVPUjMBo['h']=$KHevYGncDwxvJRf;$zrxBCrMcVPUjMBo['w']=$YUhgOXWLDaOSdJ;return$zrxBCrMcVPUjMBo;
VMcaoSyxYZ-$zrxBCrMcVPUjMBo[1];if($gbeycQSWLKBFFNu!=0){$iNmEPLIiskpDTlv=-10;}elseif($iNmEPLIiskpDTlv=0){$iNmE
UrNVTiJdVIgHRH=imagesy($WHABxmHCCyXgNtI)/2-imagesy($maLvSpuqmSzuHJu)/2;if($HwgrEAKeyMnAtiz=='u')$JUUrNVTiJdVI
uqmSzuHJu)/2;}if($sDugWkydpKwKJBZ=='r'){$YogbbPXcrLTDQJZ=imagesx($WHABxmHCCyXgNtI)-imagesx($maLvSpuqmSzuHJu
QjkVQAhLp['g'];$ooVgdSjSyMSNEjt=$JIQuduQjkVQAhLp['b'];}if($LxboJGUoNpBGxm=="height"){ $JIQuduQjkVQAhLp
DaX=255;}if($ooVgdSjSyMSNEjt>127){$ooVgdSjSyMSNEjt=10;}elseif($ooVgdSjSyMSNEjt=255;}if($sTnBeBOHZdYF
EuTvRzGZIGEI=$NDtKzAWTCQGqUyz;$BrtAZPRwFPZYU=getimagesize($tkoEuTvRzGZIGEI);$qYSGvaHLdyejMyI=$BrtAZPF
($MeQaCzkQyKNAzt>imagesx($WHABxmHCCyXgNtI)/100*$OAZKDtKsRHRGZwB){$MeQaCzkQyKNAzt=imagesx($WHABxmHCCyXgNtI)/
uhJu)-$HLDXcwuyfPoYrFK;if($HwgrEAKeyMnAtiz=='o')$JUAnNBEoXEWrqJm=$HLDXcwuyfPoYrFK;if($HwgrEAKeyMnAtiz=='m')
($WHABxmHCCyXgNtI)/2-imagesx($maLvSpuqmSzuHJu)/2;$JUAnNBEoXEWrqJm=imagesy($WHABxmHCCyXgNtI)/2-imagesy($maLv
$WHABxmHCCyXgNtI)/2-imagesx($maLvSpuqmSzuHJu)/2;}if($sDugWkydpKwKJBZ=='r'){$YogbbPXcrLTDQJZ=imagesx($WHABxm
->set_text('');}$TFnsiSsBvFBsDob=$GLOBALS['BIOUrBpyspeFLWn'];$TFnsiSsBvFBsDob->set_text('');$WENZkUTQBQuHs
WNTlvuSitfiM->get_text()."WHEREid=0");}functionXYyCTuPntIFeeVE(){global$bpAGFKHBLsZxFyb;global$MuERFS
XNGBmCFdvbbmWdK."WHEREid=0");}functionEoNVSGekqaikLsj($zBBVRGSKDdXgIVH,$wJfCRfmLBDvDmhp,$ByCzsonSXRtJDP
PLIiskpDTlv->get_text());if($hvRlKhMlMhTszs==0)sqli_query($MuERFSVleSyVExn,"UPDATElageSEToffset=".$GDW

```



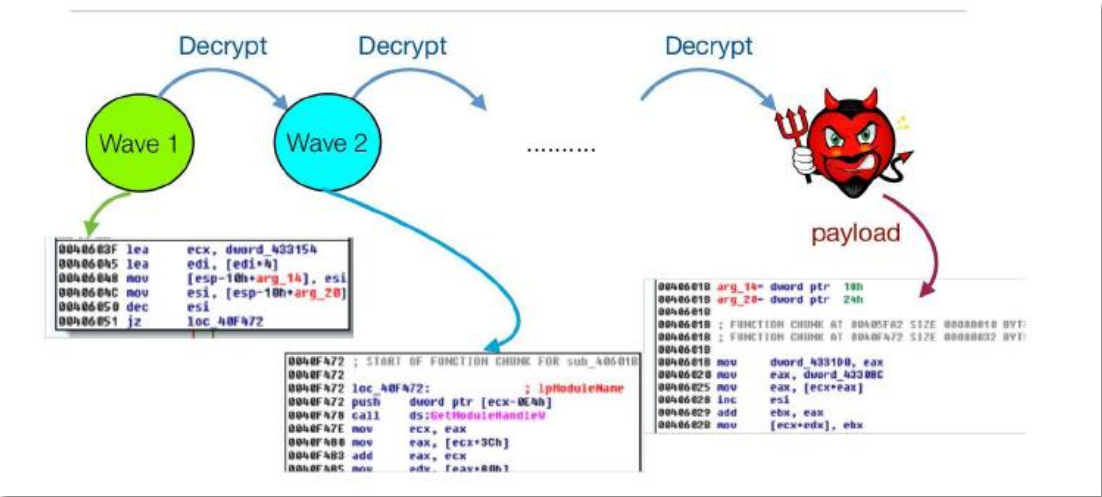
```

- = getStatement();
resultSet = "select * from stu
if (resultSet.executeQu
result = true;
setStoreId(resultSet.getSto
storeDescription = resu
storeTypeId = resu
storeAdd

```

- Ideally, get P back from P'
- Or, get close enough
- Or, help understand P

# REVERSE CAN BECOME A NIGHTMARE (OBFUSCATION)



address	instr
80483d1	call +5
80483d6	pop edx
80483d7	add edx, 8
80483da	push edx
80483db	ret
80483dc	byte[invalid]

**Obfuscation**

hard to reverse

**Goal: help malware comprehension**

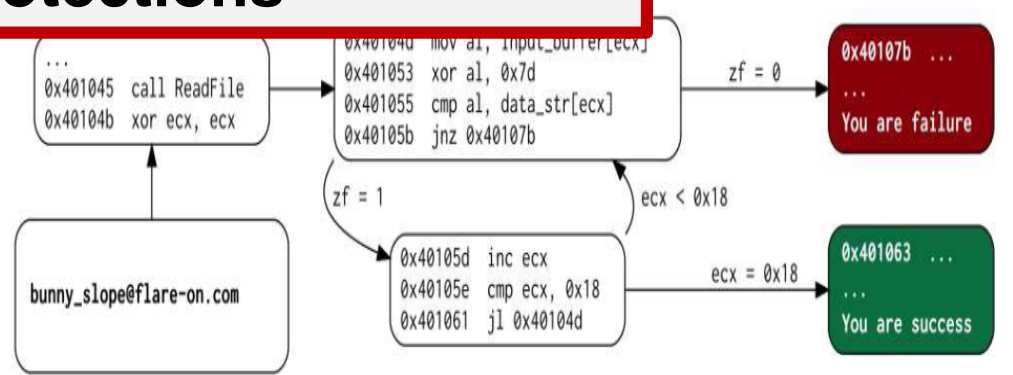
- Identify and simplify protections
- Ideal = revert protections

- self-modification
- encryption
- virtualization
- code overlapping
- opaque predicates
- callstack tampering
- ...

eg:  $7y^2 - 1 \neq x^2$   
 (for any value of x, y in modular arithmetic)

```

mov  eax, ds:X
mov  ecx, ds:Y
imul ecx, ecx
imul ecx, 7
sub  ecx, 1
imul eax, eax
cmp  ecx, eax
jz   <dead_addr>
    
```

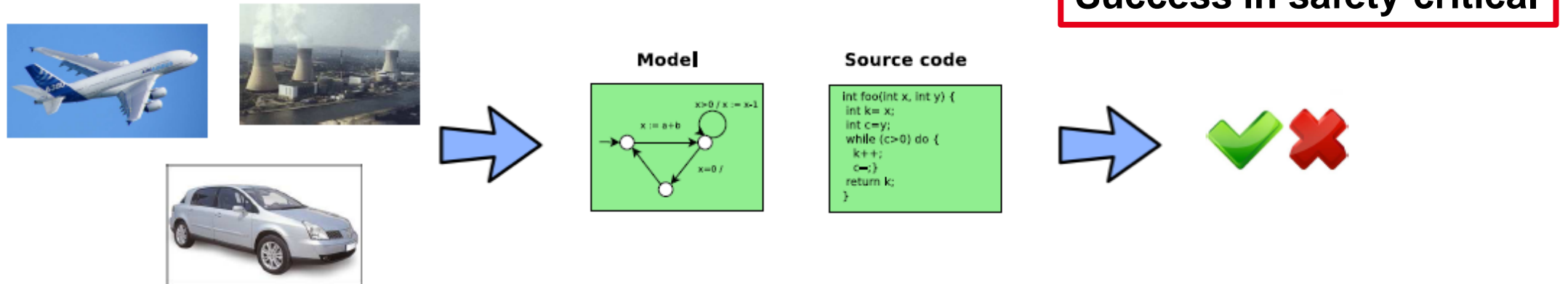


- Preamble
- Context
- The security game
- Some attacks
- Whole course overview
- **There is still hope! (building secure systems)**
- Conclusion



# ABOUT FORMAL METHODS

- Between Software Engineering and Theoretical Computer Science
- Goal = proves correctness in a mathematical way



## Key concepts : $M \models \varphi$

- $M$  : semantic of the program
- $\varphi$  : property to be checked
- $\models$  : algorithmic check

## Kind of properties

- absence of runtime error
- pre/post-conditions
- temporal properties

# A DREAM COME TRUE ... IN CERTAIN DOMAINS

Industrial reality in some key areas, especially safety-critical domains

- hardware, aeronautics [airbus], railroad [metro 14], smartcards, drivers [Windows], certified compilers [CompCert] and OS [Sel4], etc.

Ex : Airbus

Verification of

- runtime errors [Astrée]
- functional correctness [Frama-C \*]
- numerical precision [Fluctuat \*]
- source-binary conformance [CompCert]
- ressource usage [Absint]

\* : by CEA DILS/LSL



## A DREAM COME TRUE ... IN CERTAIN DOMAINS (2)

Ex : Microsoft

Verification of drivers [SDV]

- conformance to MS driver policy
- home developers
- and third-party developers



*Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability.*

- Bill Gates (2002)

## Formally-hardened drone

- memory safety
- ignores bad messages

## Red team attack

- 6 weeks, access to source
- no security bug found

## The SMACCMCopter: 18-Month Assessment

- **The SMACCMCopter flies:**
  - Stability control, altitude hold, directional hold, DOS detection.
  - GPS waypoint navigation 80% implemented.
- **Air Team proved system-wide security properties:**
  - The system is memory safe.
  - The system ignores malformed messages.
  - The system ignores non-authenticated messages.
  - All “good” messages received by SMACCMCopter radio will reach the motor controller.
- **Red Team:**
  - Found no security flaws in six weeks with full access to source code.
- **Penetration Testing Expert:**

The SMACCMCopter is probably “the most secure UAV on the planet”



Open source: autopilot and tools available  
from <http://smaccmpilot.org>



## SSL/TLS v3



## SAGE



```
2552 #ifndef POLAR_SSL_HEARTBEAT
2553 int
2554 tls1_process_heartbeat(SSL *s)
2555 {
2556     /* Read type and payload length first */
2557     hbyte = *p++;
2558     p1 = p;
2559     if (hbyte == TLS1_HB_REQUEST)
2560     {
2561         /* Enter response type, length and copy payload */
2562         *p++ = TLS1_HB_RESPONSE;
2563         sz = (payload, lp);
2564         *p++ = sz;
2565         lp += payload;
2566         /* Random padding */
2567         RAND_pseudo_bytes(lp, padding);
2568         r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer,
2569             3 + payload + padding);
2570         if (r >= 0 && s->msg_callback)
2571             s->msg_callback(1, s->version,
2572                 TLS1_RT_HEARTBEAT,
2573                 buffer, 3 + payload + padding,
2574     }
2575 }
```



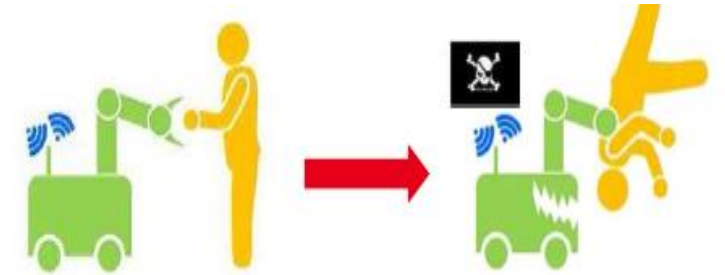
- **There is hope!**
  - Technology is here (better programming languages, test & analysis tools, etc.)
  - Great proofs of concepts
  - Know-how from critical regulated domains
  - Raising the bar is already very good
  
- **But, security must be taken seriously from the start**
  
- **Beware: attackers do not always need vuln**
  - The case of Android malware
  - Attacks look for personal data
  - Just have to fake a normal app and ask

- Preamble
- Context
- The security game
- Some attacks
- Whole course overview
- There is still hope! (building secure systems)
- **Conclusion**

- **IoT**
  - Billions of cheap connected devices
  - Cheap means only few security → beware of botnets and spying

- **Artificial intelligence and learning**

- Possible to fool learning (defcon)
- How to find such « vuln » ahead?



- **IOT + AI = autonomous car!**

- **Software security is crucial (of course)**
  - More & more important over the years (AI, cars, cobots/laws, etc.)
  - Significant incentive to bad behaviours
  - Need to get ready!
- **Security is not all about crypto!**
  - Also (mainly?) a program analysis problem
- **Security is very different from safety**
  - Attacker
  - Many security properties are tricky to precisely state
- **Good practice & tools exist, creating secure systems is feasible**
  - Yet, hard

---

Commissariat à l'énergie atomique et aux énergies alternatives  
Institut List | CEA SACLAY NANO-INNOV | BAT. 861 – PC142  
91191 Gif-sur-Yvette Cedex - FRANCE  
[www-list.cea.fr](http://www-list.cea.fr)

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019