AI FOR HACKER

Automatic Exploit Generation for Application Source Code Analysis

THE TEAM

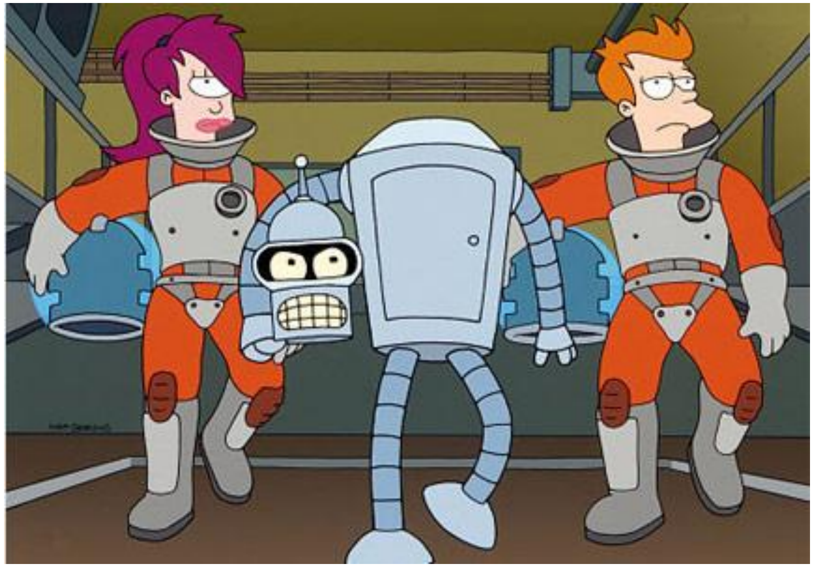
Alexey Moskvin, Head of Research

Sergey Plekhov, Head of Development

Vladimir Kochetkov, Hacking, PoC

Denis Baranov, Project Manager

Sergey Gordeychik, Business Development, Marketing



TO ANALYZE ~ 400 APPLICATIONS…

WE NEED THE AI

THERE ARE DIFFERENT KINDS OF ROBOTS

Marketing approach

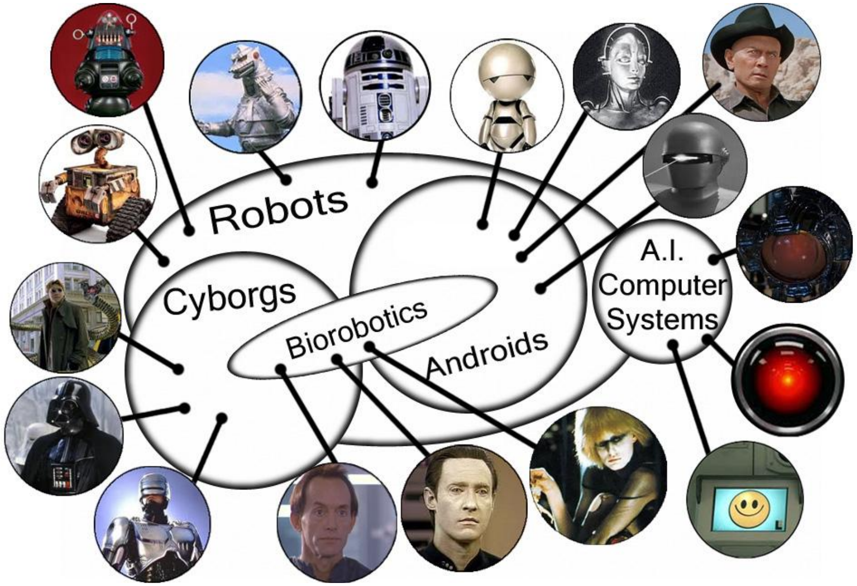
- Interactive Application Security Testing (IAST)
- Dynamic Application Security Testing (DAST)
- Static Application Security Testing (SAST)

Technical approach

- Black Box/White Box
- Static/Dynamic

Scientific approach

- It's all relative



DAST

We don’t have access to [server] application

Fuzzing/Fault injection

Pro

- Easy to implement/Easy to verify results/Low level of false positives
- Language/Framework/Backend independent

Cons

- Weak API coverage/Auth/Web 2.0
- Application should be deployed/Can terminate app*
- $(O(c^n), c > 1)$ **

*And admins will terminate you

**Never stops

SAST

We have access full access to application [source code]

Model checking/correctness properties of finite-state systems

Pro

- [possible] Good coverage/Don’t need to deploy app
- [possible] Good performance*

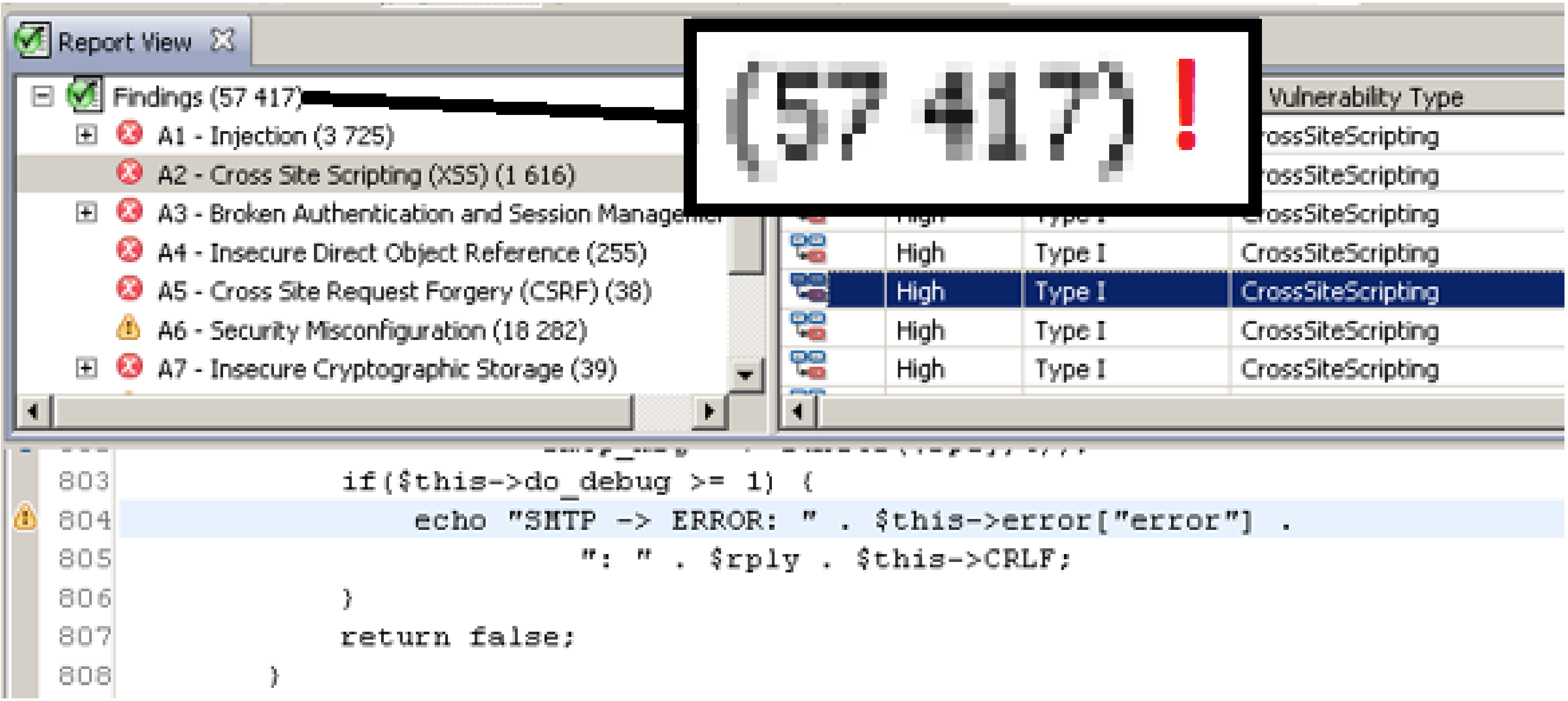
Cons

- Hard to implement/Hard to verify results
- [can generate]a lot of of false positives/Language dependent
- K := { (i, x) | program i will eventually halt if run with input x} *

*Because of computation timeouts

**The halting problem

SAST



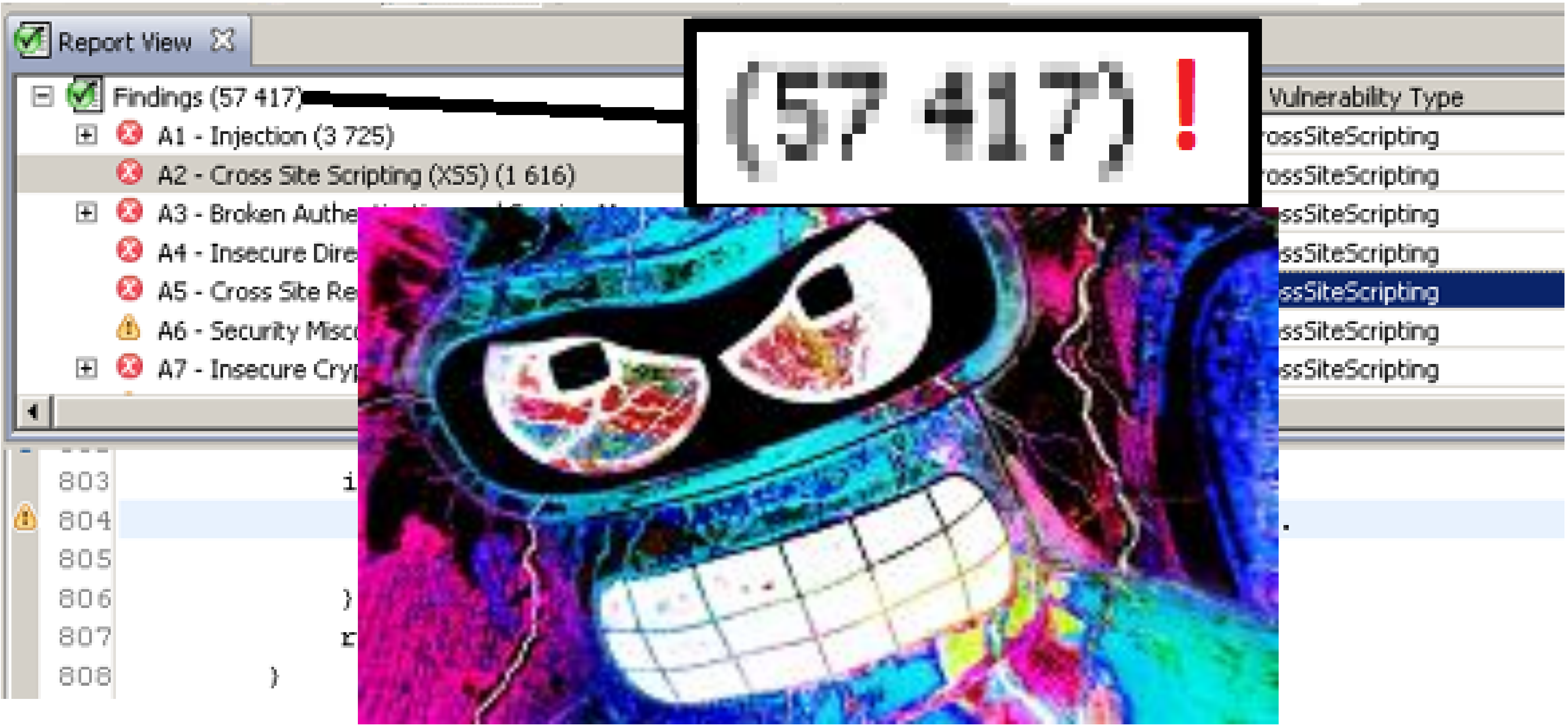
- Findings (57 417)
  - A1 - Injection (3 725)
  - A2 - Cross Site Scripting (XSS) (1 616)
  - A3 - Broken Authentication and Session Manageme
  - A4 - Insecure Direct Object Reference (255)
  - A5 - Cross Site Request Forgery (CSRF) (38)
  - A6 - Security Misconfiguration (18 282)
  - A7 - Insecure Cryptographic Storage (39)

(57 417) !

		Vulnerability Type
		rossSiteScripting
		rossSiteScripting
High	Type I	CrossSiteScripting
High	Type I	CrossSiteScripting
High	Type I	CrossSiteScripting
High	Type I	CrossSiteScripting
High	Type I	CrossSiteScripting

```
            if($this->do_debug >= 1) {
                echo "SMTP -> ERROR: " . $this->error["error"] .
                         ": " . $rply . $this->CRLF;
            }
            return false;
        }
```

SAST



Report View

Findings (57 417)
- A1 - Injection (3 725)
- A2 - Cross Site Scripting (XSS) (1 616)
- A3 - Broken Authe
- A4 - Insecure Dire
- A5 - Cross Site Re
- A6 - Security Misc
- A7 - Insecure Cryp

(57 417) !

Vulnerability Type
rossSiteScripting
rossSiteScripting
ssSiteScripting
ssSiteScripting
ssSiteScripting
ssSiteScripting
ssSiteScripting

```
                i

                }
                r
        }
```

IAST

Have full access to application [source code]/system and can patch it

Fuzzing/Instrumentation/Data [control] flow tracing

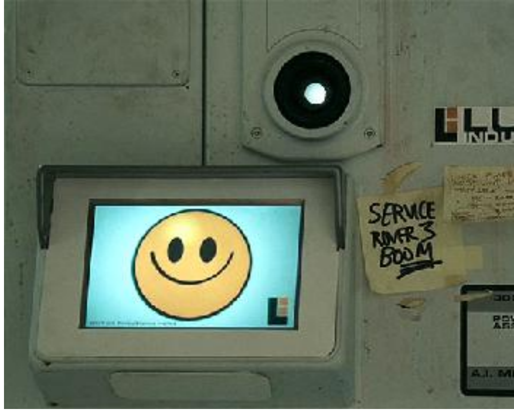
Pro

- Can combine strengths of SAST and DAST
- Control of dataflow/Second chance vulns/binary analysi

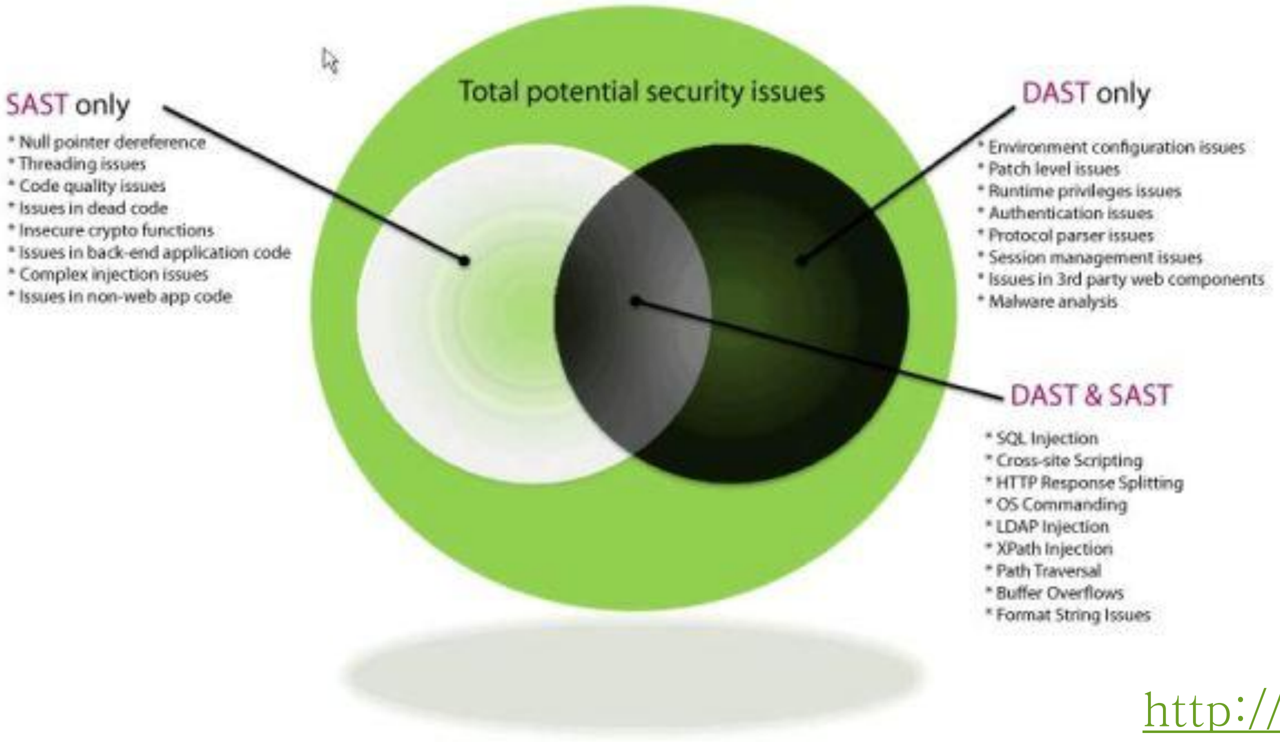
Cons

- Can combine weaknesses of SAST and DAST
- Need fuzzer/Need to patch server
- Generates tons of results (execution trace)
- Need to have/patch "live" system

CAN WE USE (.AST)



Dynamic Application Security Testing (DAST) and Static Application Security Testing (SAST) -- Issue Type Coverage



Total potential security issues

SAST only

- Null pointer dereference
- Threading issues
- Code quality issues
- Issues in dead code
- Insecure crypto functions
- Issues in back-end application code
- Complex injection issues
- Issues in non-web app code

DAST only

- Environment configuration issues
- Patch level issues
- Runtime privileges issues
- Authentication issues
- Protocol parser issues
- Session management issues
- Issues in 3rd party web components
- Malware analysis

DAST & SAST

- SQL Injection
- Cross-site Scripting
- HTTP Response Splitting
- OS Commanding
- LDAP Injection
- XPath Injection
- Path Traversal
- Buffer Overflows
- Format String Issues

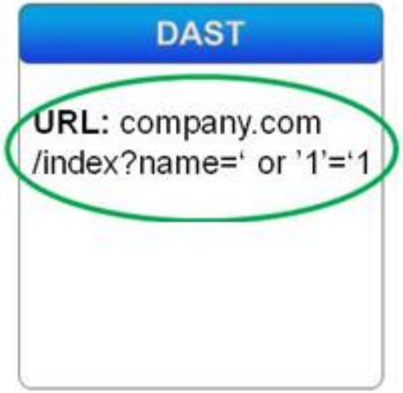
http://ibm.co/HeDsGw

URL-TO-SOURCE MAPPINGS

SAST and DAST have produces incompatible output

SAST: line of code, CFG

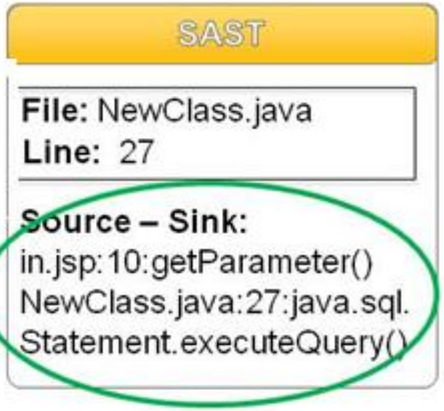
DAST: Input data (HTTP Request)



DAST

URL: company.com /index?name=' or '1'='1

?

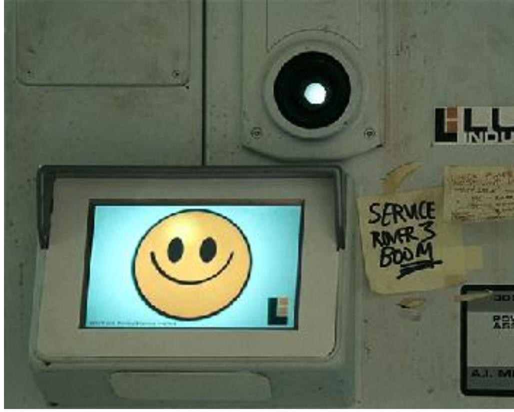


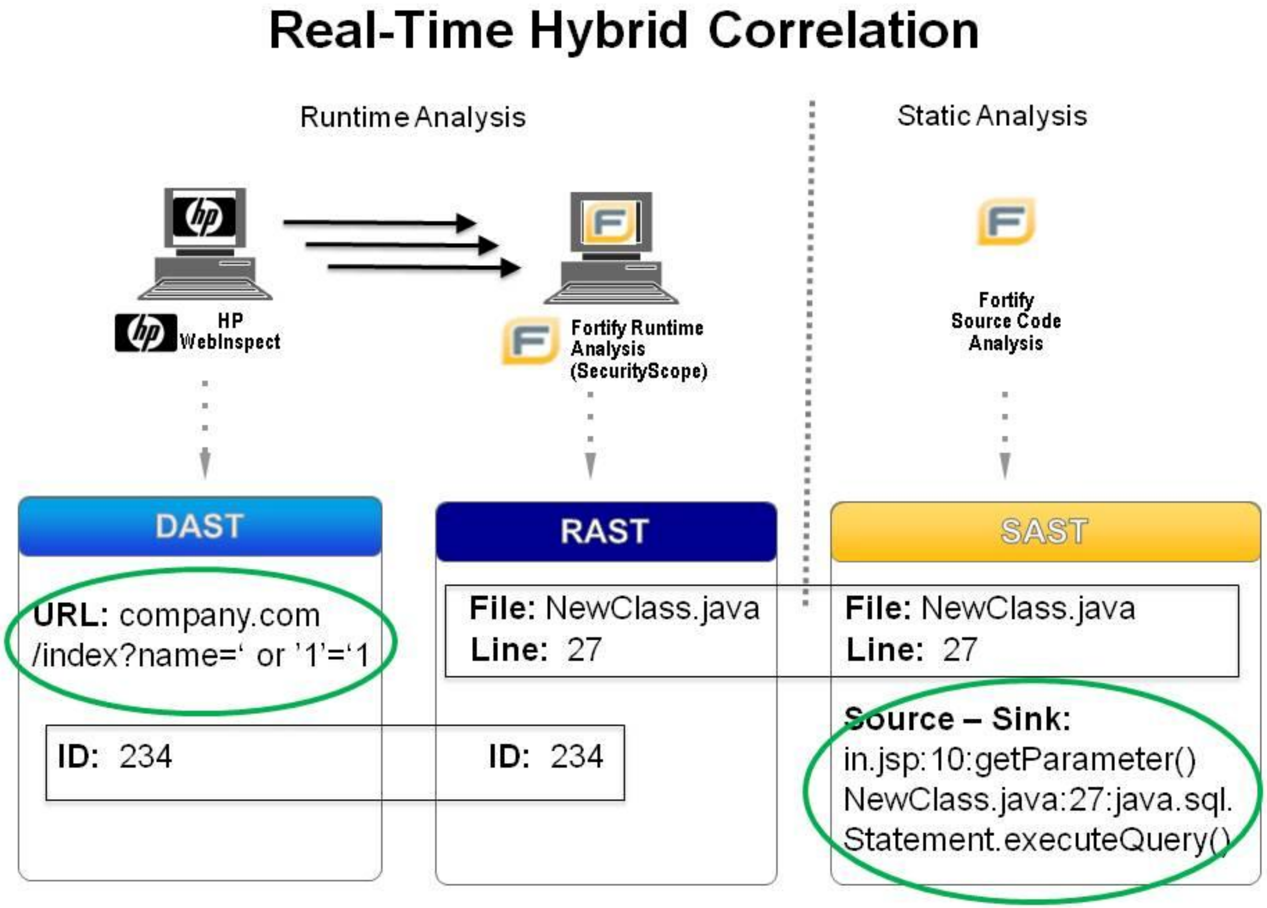
SAST

File: NewClass.java
Line: 27

Source – Sink:
in.jsp:10:getParameter()
NewClass.java:27:java.sql.Statement.executeQuery()

HYBRID ANALYSIS!





Real-Time Hybrid Correlation

Runtime Analysis

Static Analysis

HP WebInspect

Fortify Runtime Analysis (SecurityScope)

Fortify Source Code Analysis

DAST

URL: company.com /index?name=' or '1'='1

ID: 234

RAST

File: NewClass.java
Line: 27

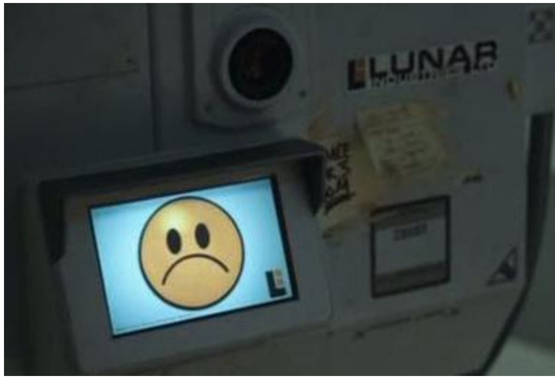
ID: 234

SAST

File: NewClass.java
Line: 27

Source – Sink:
in.jsp:10:getParameter()
NewClass.java:27:java.sql.Statement.executeQuery()

http://bit.ly/17wbvnL



REALITY

Need to have and to patch “live” system/source code

Need to analyze application several times

Magic to correlate “line number” (SAST) and “input data” (DAST)

$(O(c^n), c > 1)$

*

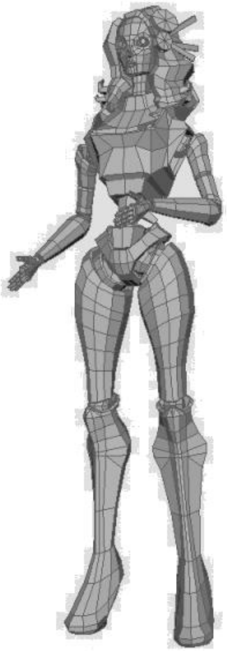
*Never stops

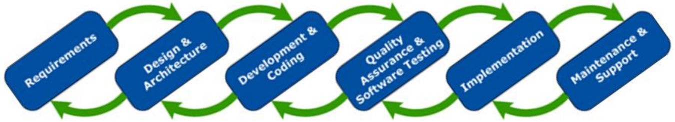
PERFECTION?

No live system

Low level of false positives

Automatic exploits generation!

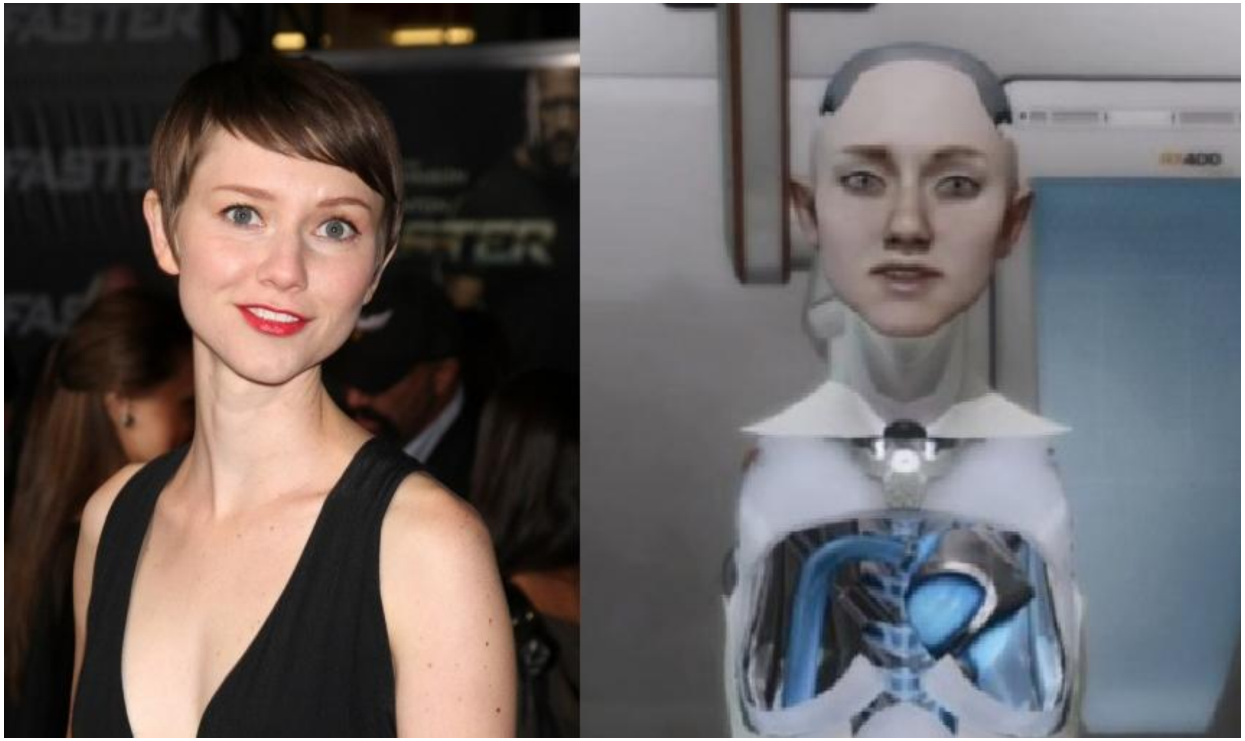




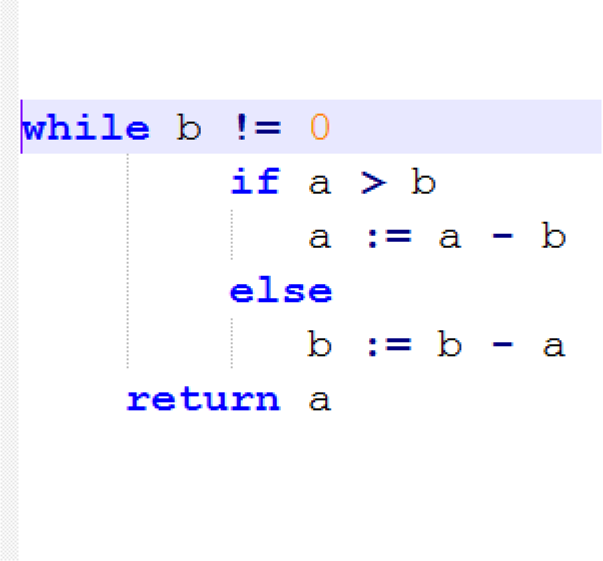
PERFECTION: NO LIVE SYSTEM

Need to use static analysis

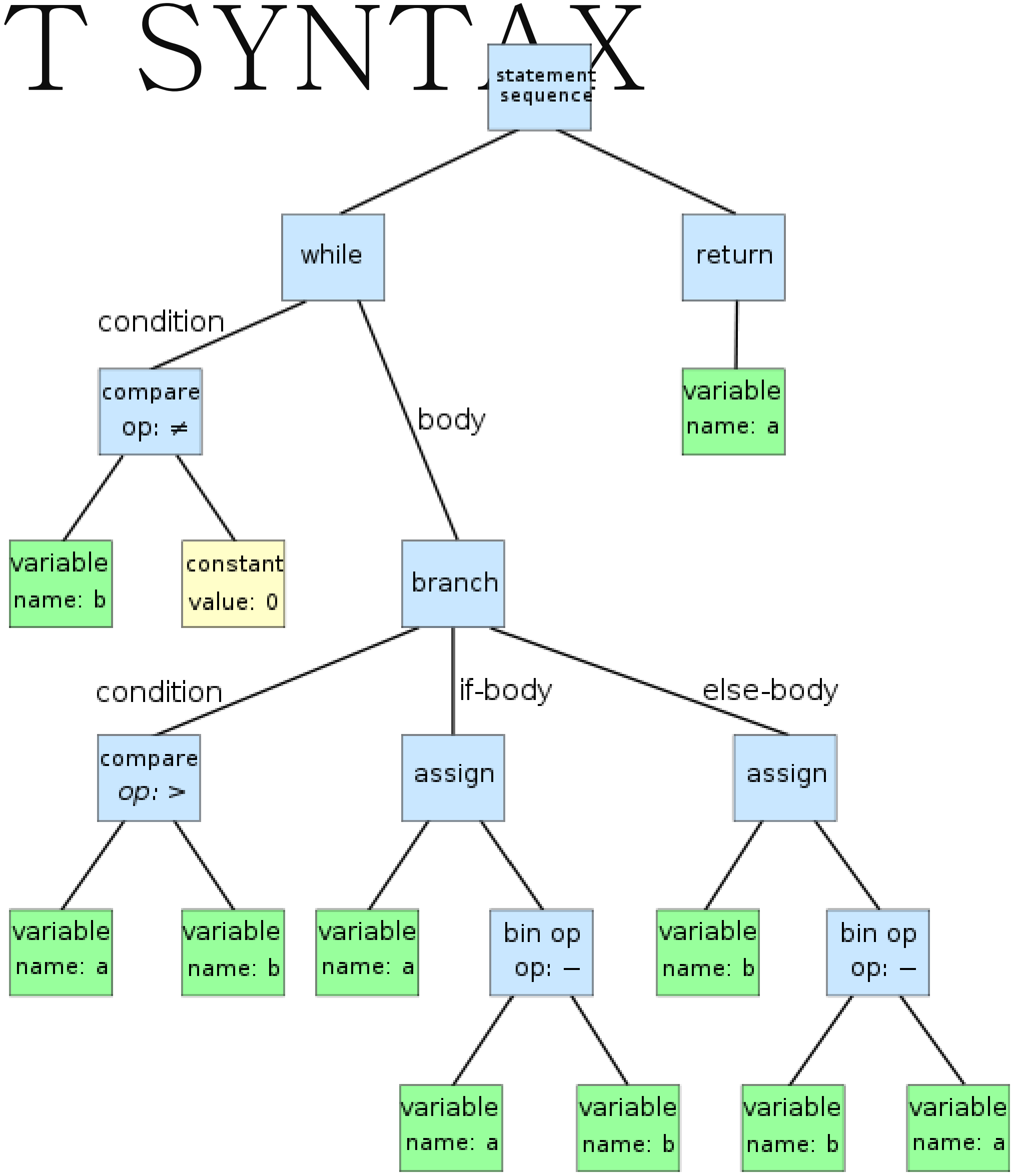
Proper model representation is half the battle



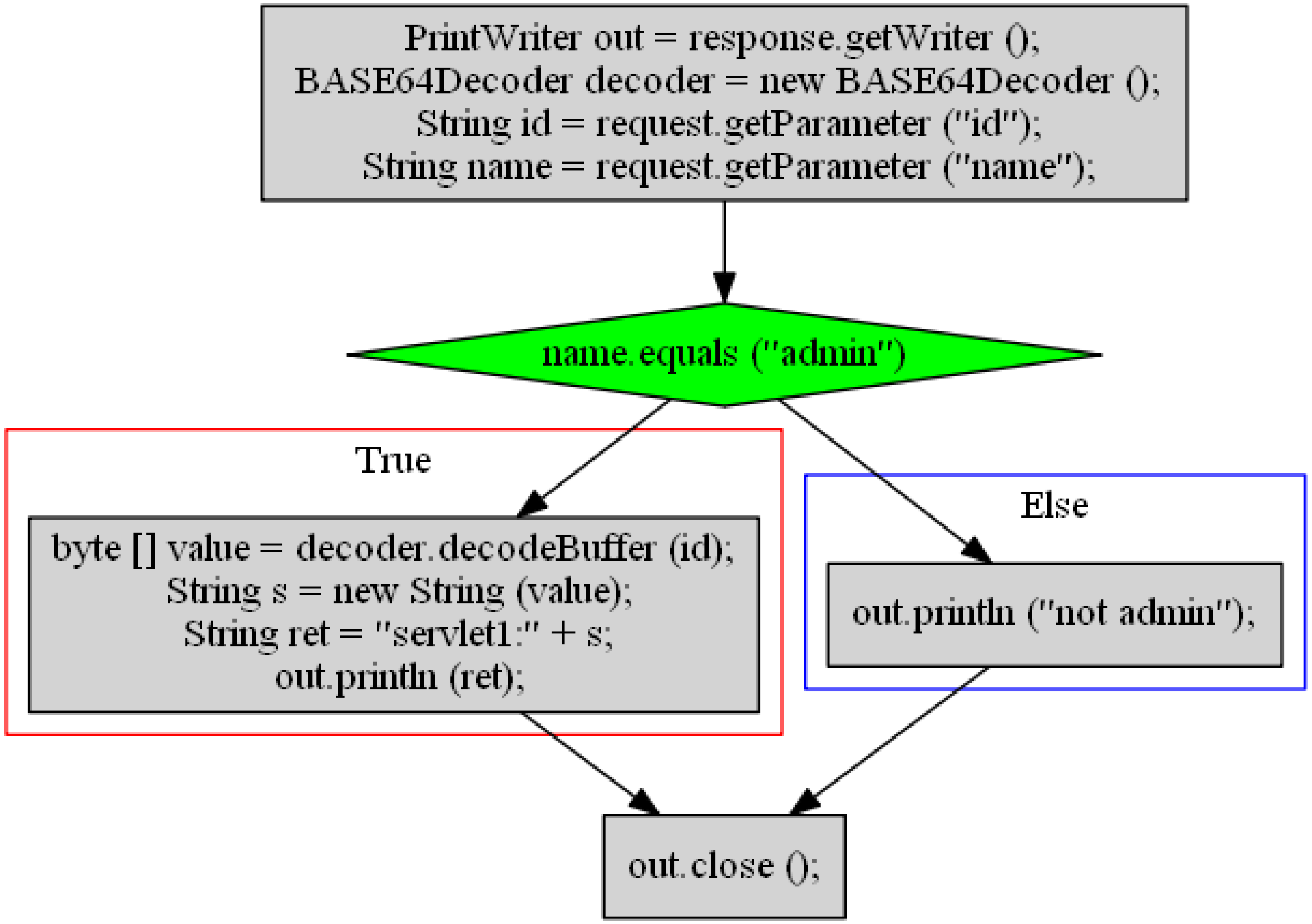
ABSTRACT SYNTAX TREE



```
while b != 0
        if a > b
           a := a - b
        else
           b := b - a
    return a
```



CONTROL FLOW GRAPH



```
PrintWriter out = response.getWriter ();
BASE64Decoder decoder = new BASE64Decoder ();
String id = request.getParameter ("id");
String name = request.getParameter ("name");
```

```
name.equals ("admin")
```

True

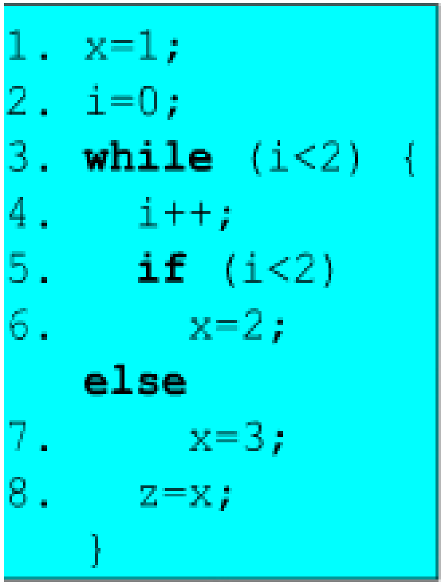
```
byte [] value = decoder.decodeBuffer (id);
String s = new String (value);
String ret = "servlet1:" + s;
out.println (ret);
```

Else

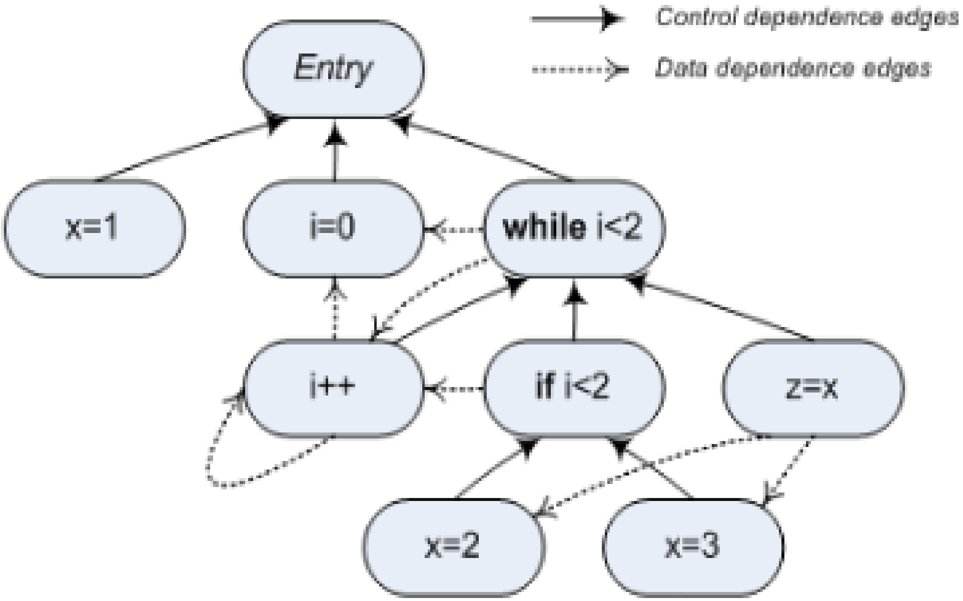
```
out.println ("not admin");
```

```
out.close ();
```

PROGRAM DEPENDENCE GRAPH

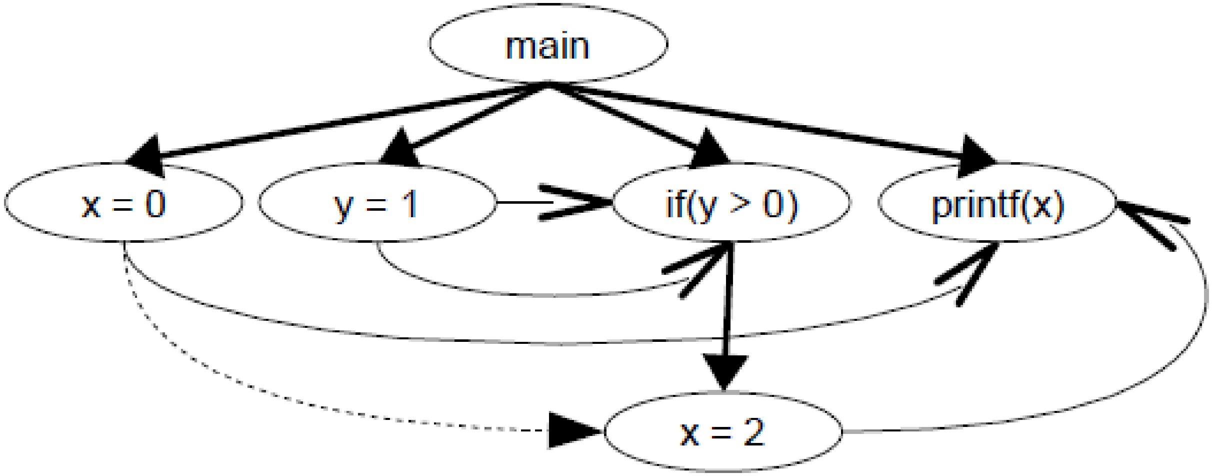


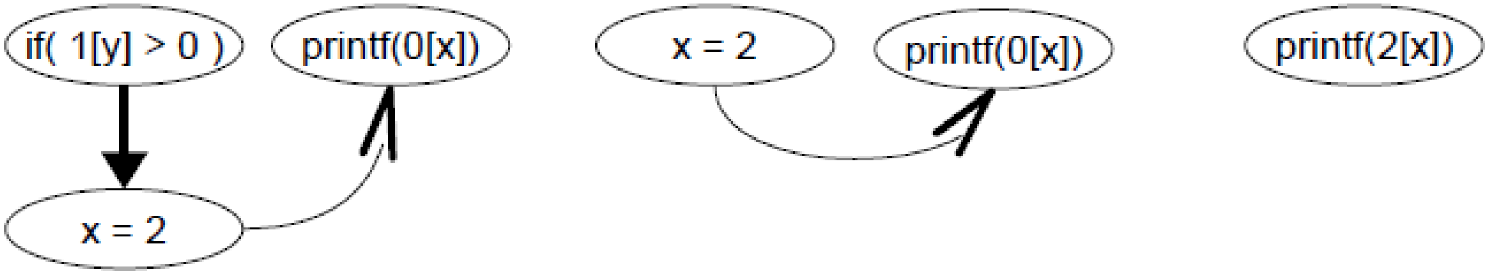
```
x=1;
i=0;
while (i<2) {
  i++;
  if (i<2)
    x=2;
else
    x=3;
  z=x;
}
```



SYMBOLIC EXECUTION

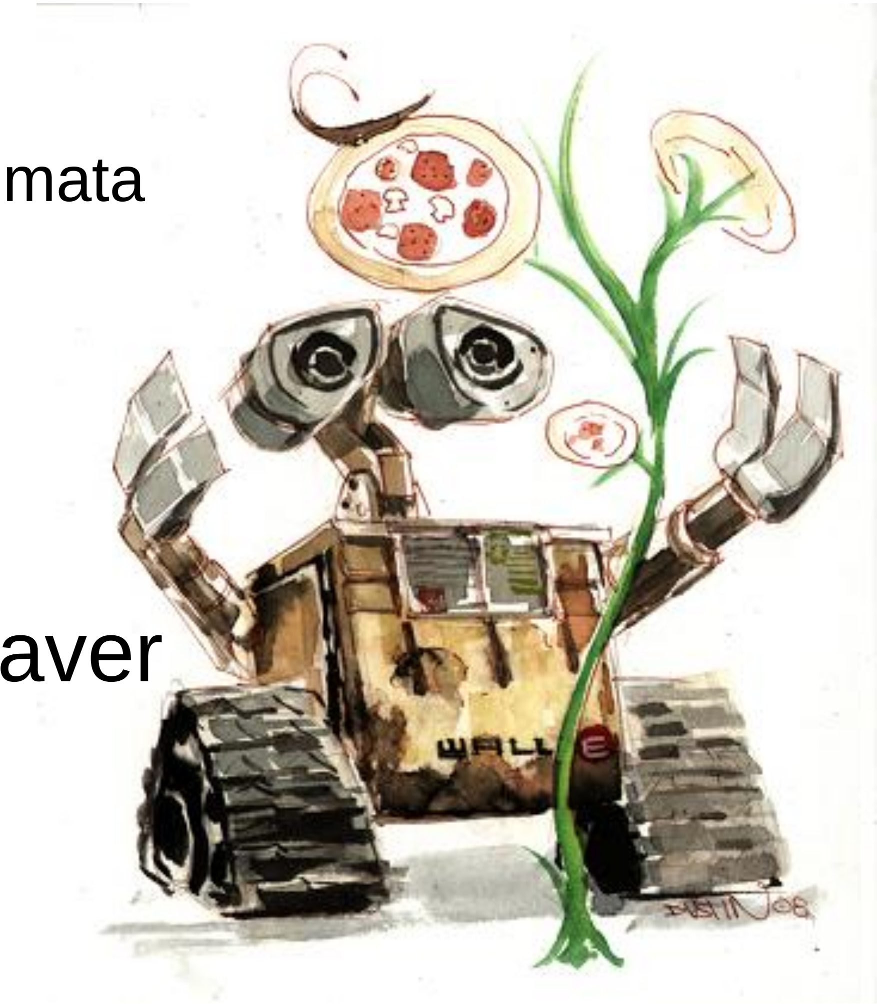
```
main()
{
  x = 0;
  y = 1;
  if(y > 0)
    x = 2;
  print(x);
}
```





SYMBOLIC EXECUTION!

Microsoft Automata



Z3

KLEE/Kleaver

SYMBOLIC EXECUTION :(

Path Explosion *

Full support of language (functions/frameworks/environment)**

[sometimes] too far from real code [execution flow]***

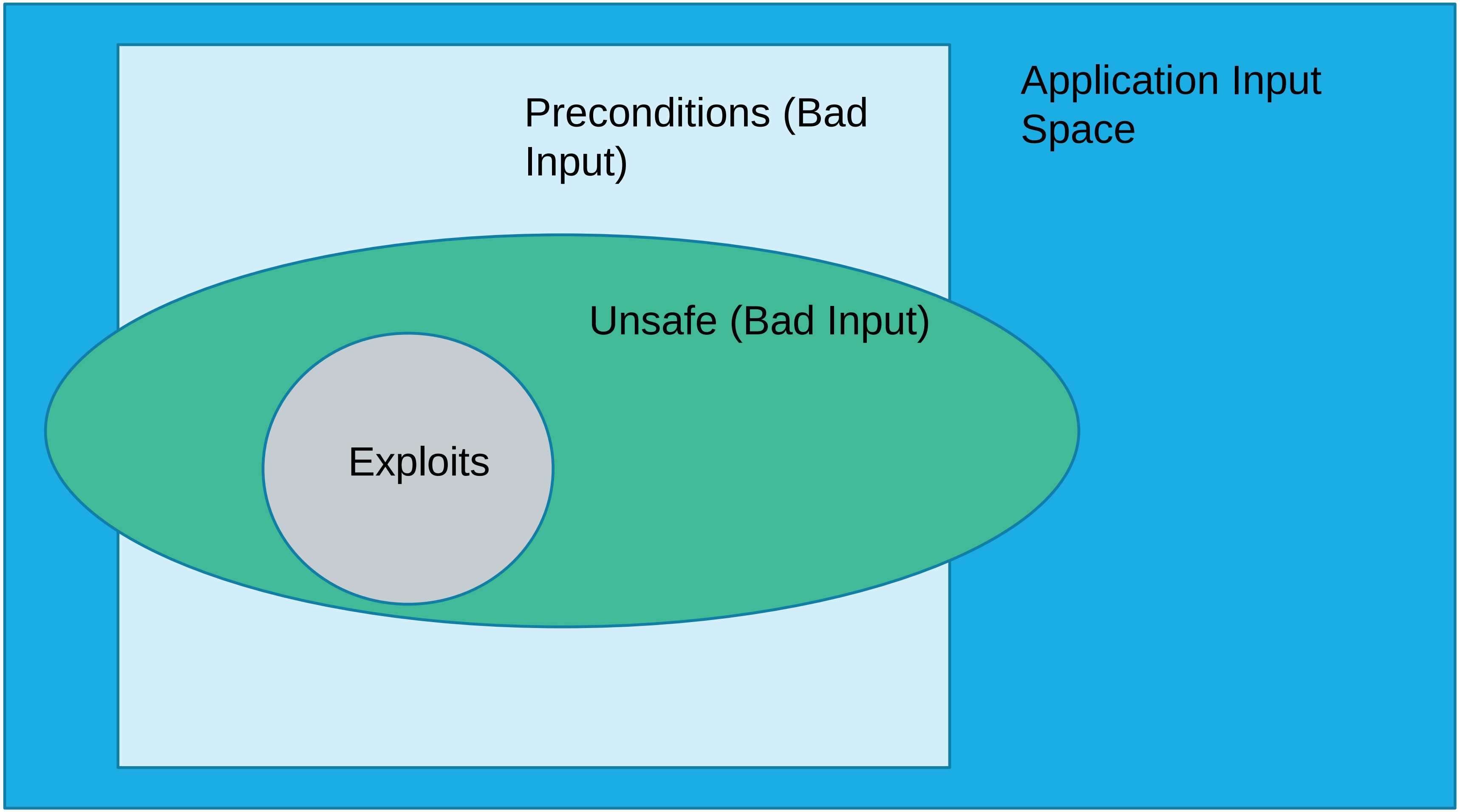
*Number of paths grows exponentially with program size and can be infinite ****

Zillions man-hours with endless updates**

***SAT was the first known NP-complete problem, as proved by Stephen Cook in

**** Never stops

!FALSE POSITIVES == EXPLOITS



EXPLOIT IS USEFUL TO

prove that vulnerability exists*

make additional [dynamic | automatic] checks**

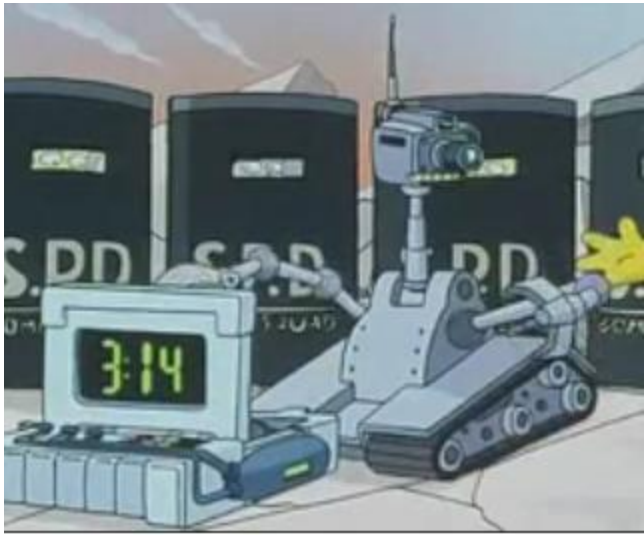
create test cases for QA

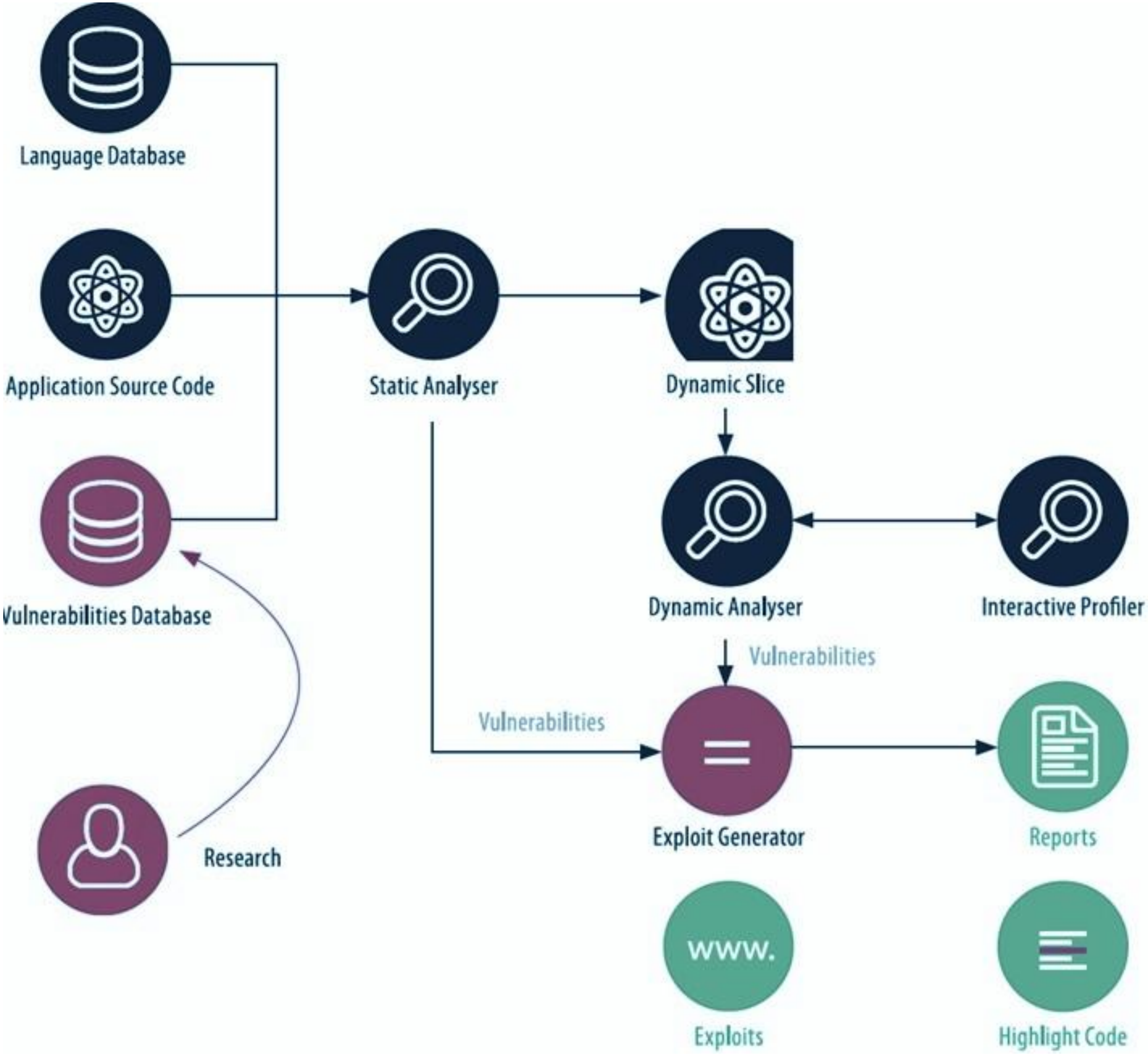
generate signatures/virtual patches for AF/IDS***

* get devs to shut up and fix the bug

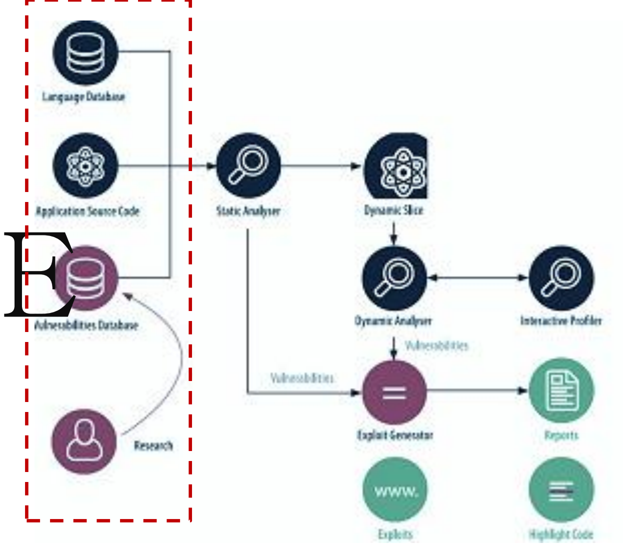
**automatic verification via fuzzing

***self-defending application




Language Database
Application Source Code
Vulnerabilities Database
Research
Static Analyser
Dynamic Slice
Dynamic Analyser
Interactive Profiler
Vulnerabilities
Vulnerabilities
Exploit Generator
Reports
www.
Exploits
Highlight Code

KNOWLEDGE BASE



Languages grammar

- Input functions
- Filtering functions

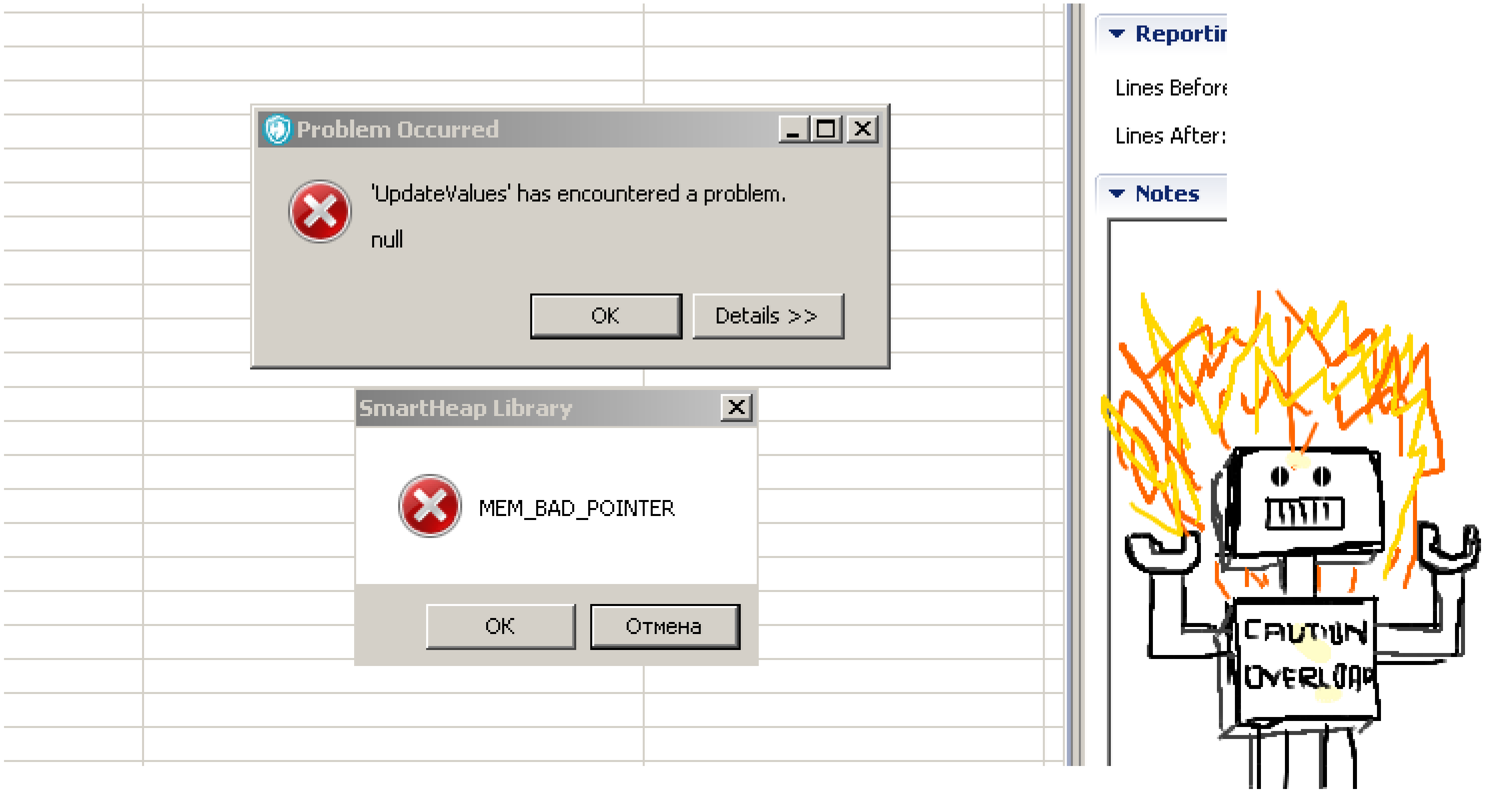
Potentially Vulnerable Functions (PVF)

- Related Vulnerabilities
- Related Preconditions (Bad Inputs)
- Related Exploit Creation Rules

Safe functions

- Can be called without any risk

WHY SLICING?



Problem Occurred

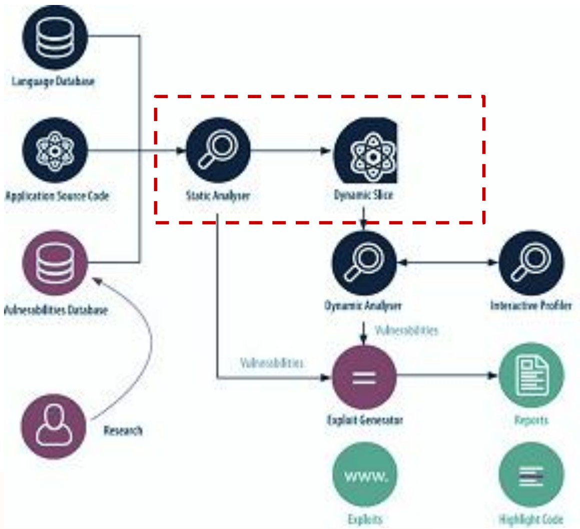
'UpdateValues' has encountered a problem.

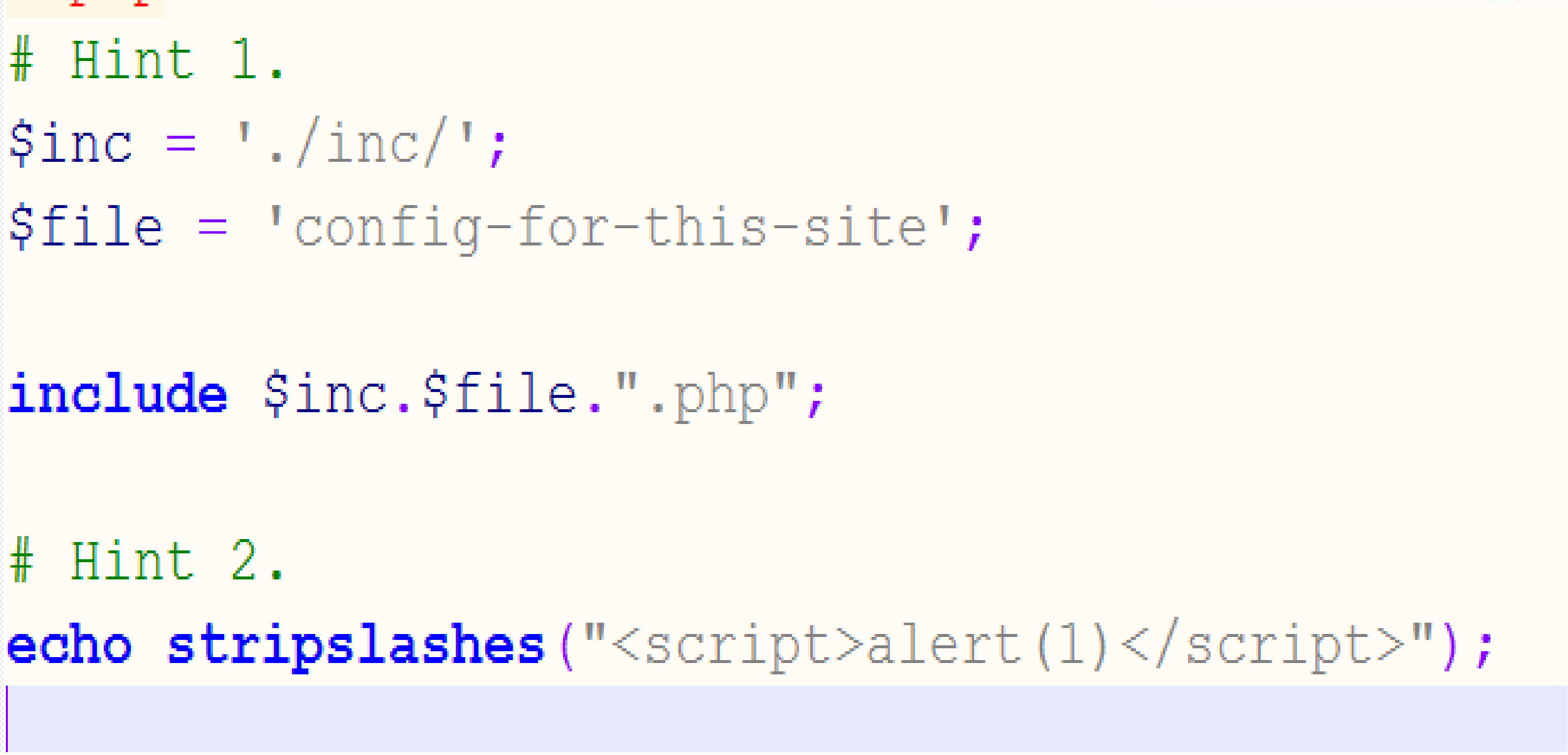
null

SmartHeap Library

MEM_BAD_POINTER

DYNAMIC SLICING



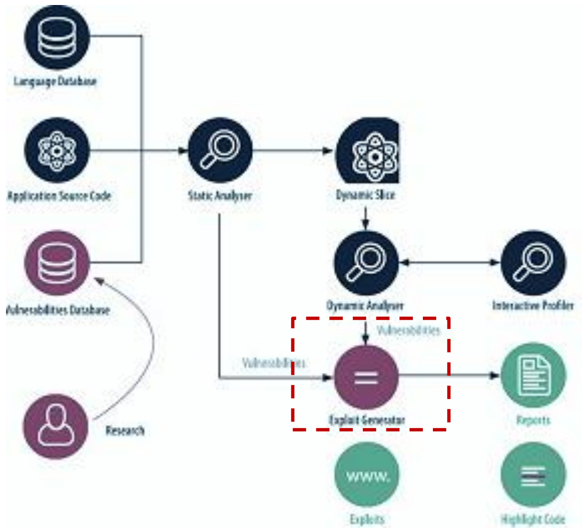


```php
<?php
# Hint 1.
$inc = './inc/';
$file = 'config-for-this-site';

include $inc.$file.".php";

# Hint 2.
echo stripslashes("<script>alert(1)</script>");
```

SOLVER



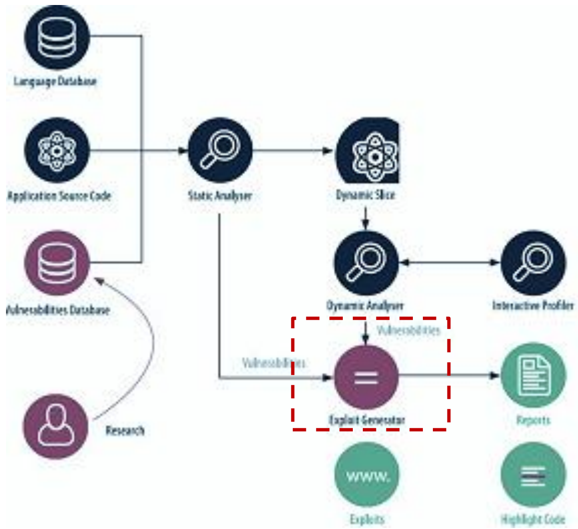
```
<?php // /test.php
print base64_decode ($_GET['x']) ;
?>
```

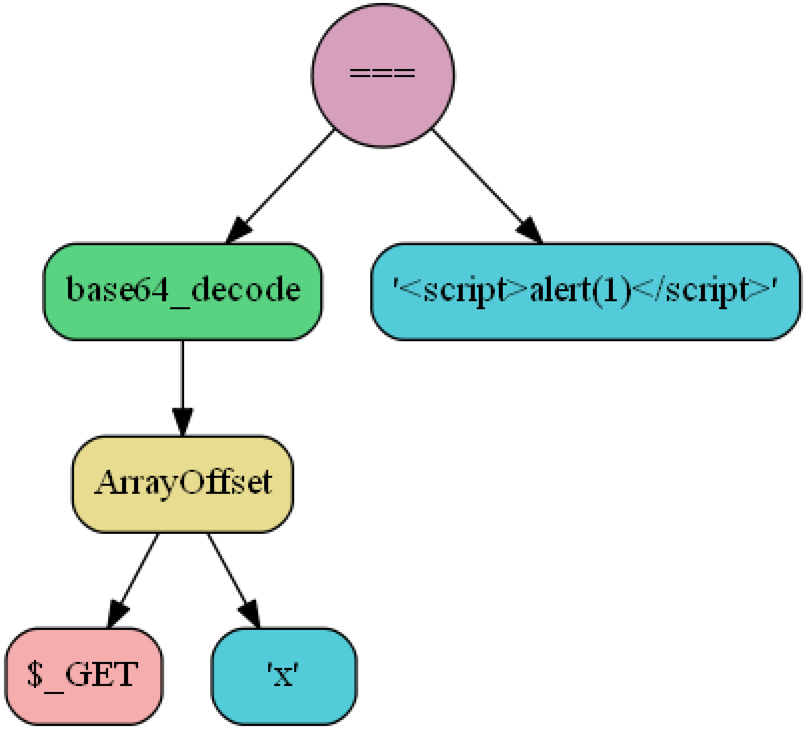
exploit:

GET

/test.php?x=PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg%3D%3D

SOLVER





===

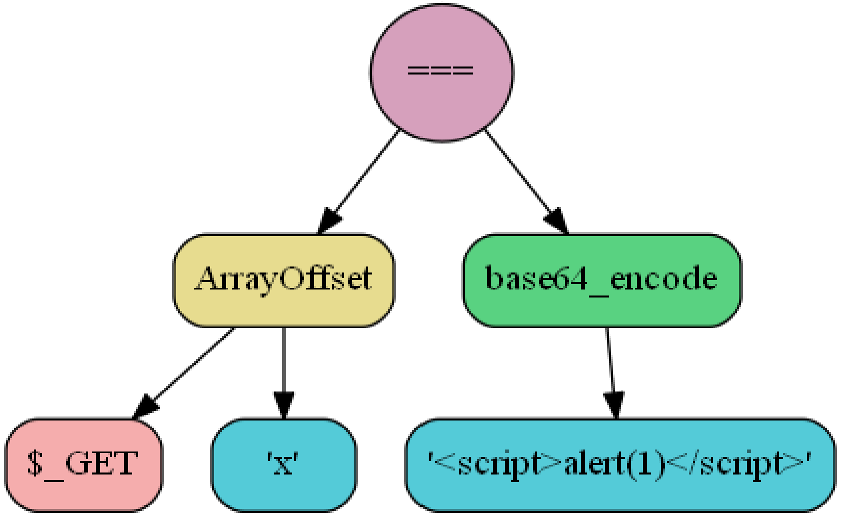
base64_decode

'<script>alert(1)</script>'

ArrayOffset

$_GET

'x'



===

ArrayOffset

base64_encode

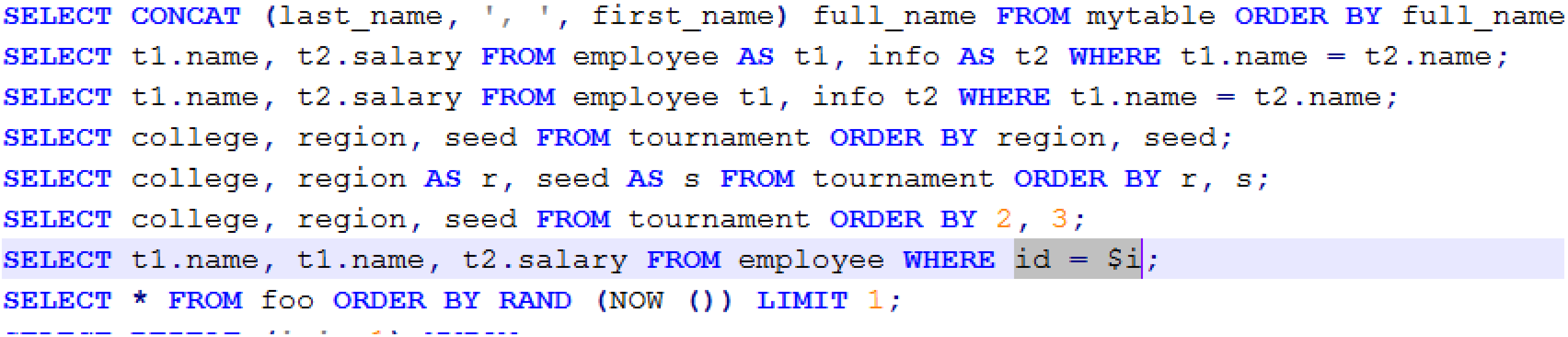
$_GET

'x'

'<script>alert(1)</script>'

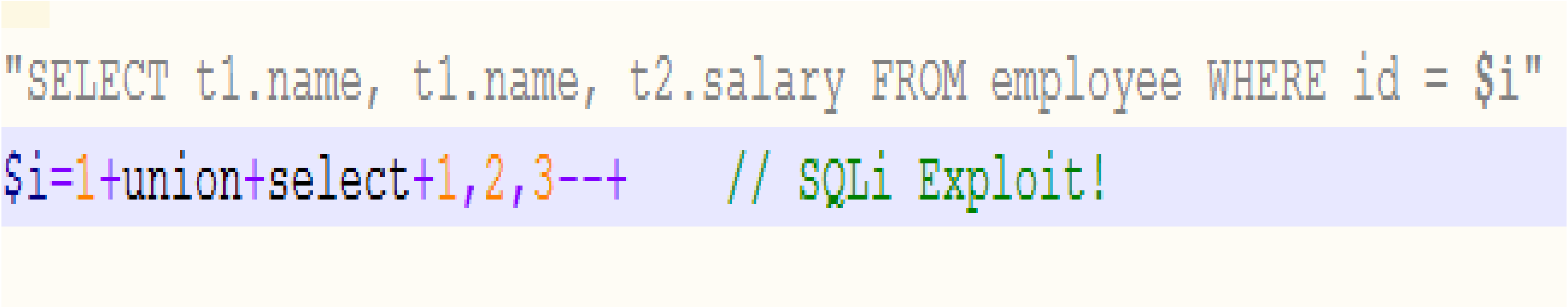
DEMO

INSIDE IN [ISLAND] GRAMMAR



```
SELECT CONCAT (last_name, ', ', first_name) full_name FROM mytable ORDER BY full_name
SELECT t1.name, t2.salary FROM employee AS t1, info AS t2 WHERE t1.name = t2.name;
SELECT t1.name, t2.salary FROM employee t1, info t2 WHERE t1.name = t2.name;
SELECT college, region, seed FROM tournament ORDER BY region, seed;
SELECT college, region AS r, seed AS s FROM tournament ORDER BY r, s;
SELECT college, region, seed FROM tournament ORDER BY 2, 3;
SELECT t1.name, t1.name, t2.salary FROM employee WHERE id = $i;
SELECT * FROM foo ORDER BY RAND (NOW ()) LIMIT 1;
```

Change MySQL Grammar



```
"SELECT t1.name, t1.name, t2.salary FROM employee WHERE id = $i"
$i=1+union+select+1,2,3--+    // SQLi Exploit!
```

CONDITIONS

We can’t [symbolically | interactive] resolve all part of equation

Session id’s in files:

- (file('../admin/conf/config.inc')[2] == session_id())

Session values are set:

- $_SESSION["admin_login"]==true

External connections:

- ftp_connect(str_replace('ftp://', '', $_POST['ftpsite']))

Configuration:

- !((strpos(php_sapi_name(), 'apache') !== False))
- sqlsrv_connect('***', array('Database' => '', 'UID' => '***', 'PWD' => '***'))==True

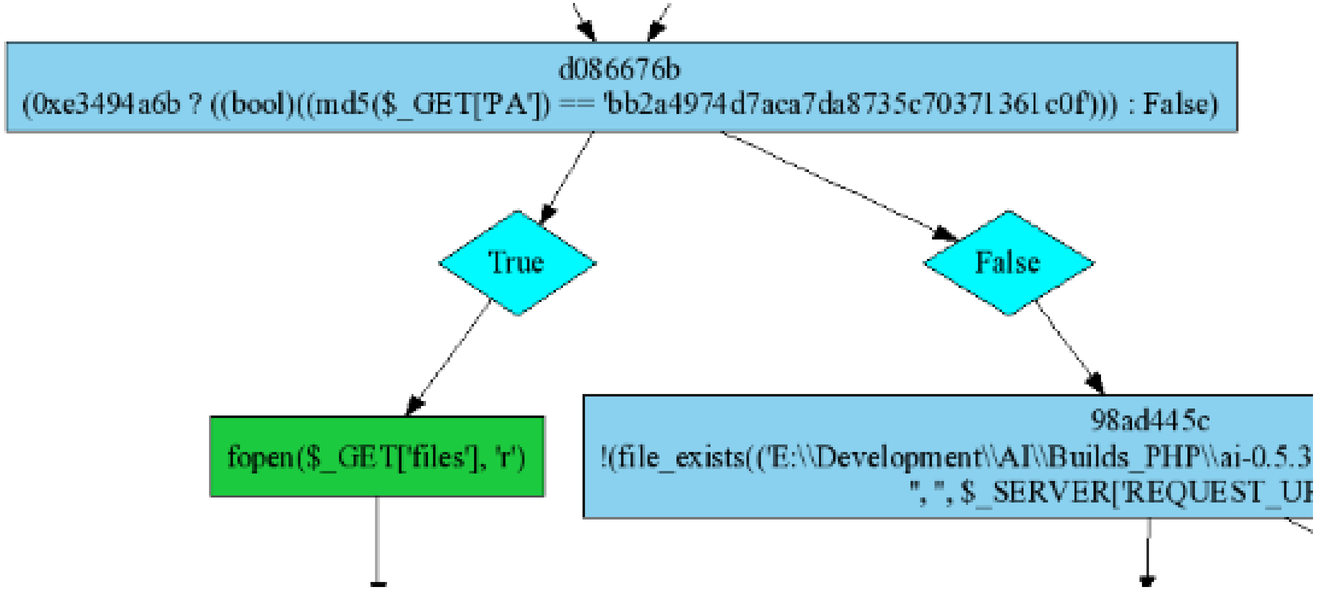
BACKDOORS?

Exploit:

GET/core/jscss.php?files=%2F..%2F..%2Fetc%2Fpasswd

Conditions:

(md5($_GET['PA']) ===
'bb2a4974d7aca7da8735c70371361c0f')


d086676b
(0xe3494a6b ? ((bool)((md5($_GET['PA']) == 'bb2a4974d7aca7da8735c70371361c0f'))) : False)
True
False
fopen($_GET['files'], 'r')
98ad445c
!(file_exists(('E:\\Development\\AI\\Builds_PHP\\ai-0.5.3
", ", $_SERVER['REQUEST_UF

BACKDOORS!

…we use it

for emergency

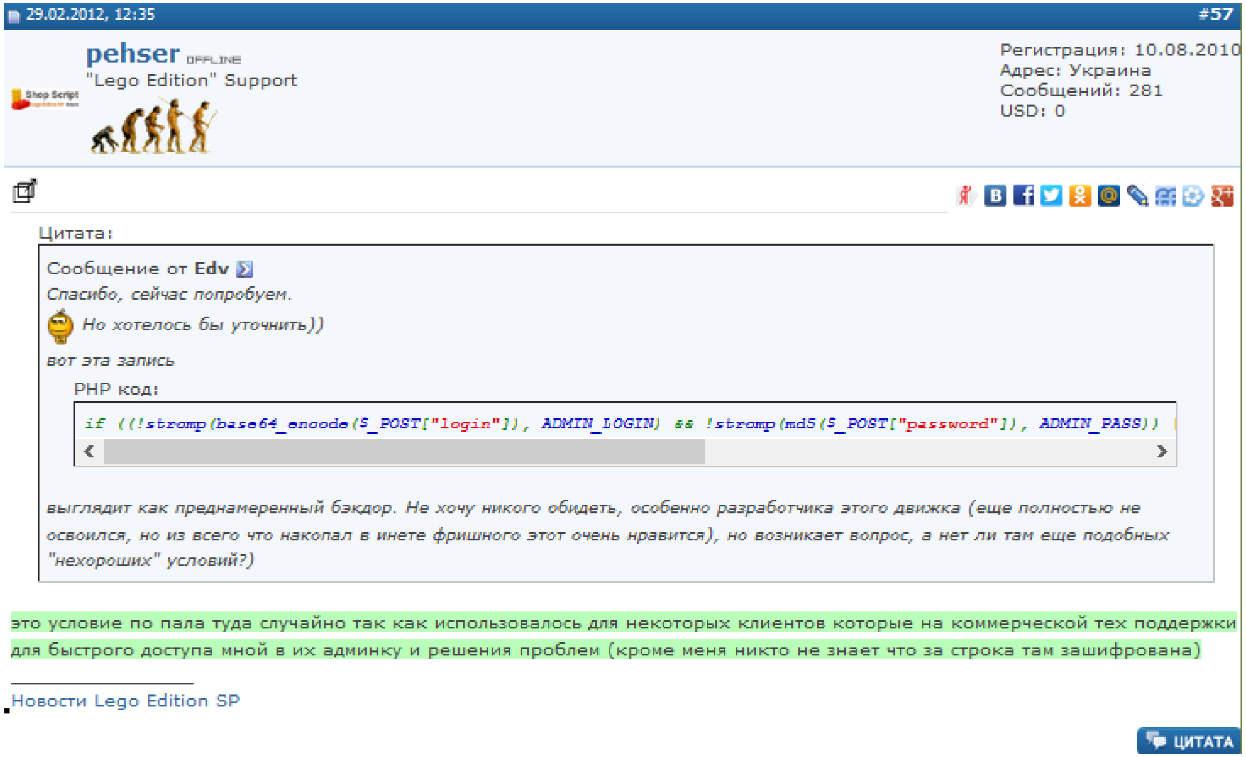
support cases

when we need

to access files

but we don’t

have a password…



29.02.2012, 12:35 #57

pehser OFFLINE
"Lego Edition" Support

Регистрация: 10.08.2010
Адрес: Украина
Сообщений: 281
USD: 0

Цитата:

> Сообщение от **Edv**
> *Спасибо, сейчас попробуем.*
> *Но хотелось бы уточнить))*
>
> *вот эта запись*
>
> PHP код:
>
> ```
> if ((!strcmp(base64_encode($_POST["login"]), ADMIN_LOGIN) && !strcmp(md5($_POST["password"]), ADMIN_PASS))
> ```
>
> *выглядит как преднамеренный бэкдор. Не хочу никого обидеть, особенно разработчика этого движка (еще полностью не освоился, но из всего что накопал в инете фришного этот очень нравится), но возникает вопрос, а нет ли там еще подобных "нехороших" условий?)*

это условие по пала туда случайно так как использовалось для некоторых клиентов которые на коммерческой тех поддержки для быстрого доступа мной в их админку и решения проблем (кроме меня никто не знает что за строка там зашифрована)

Новости Lego Edition SP

DEMO

PRACTICAL TESTS

SECOND CHANCE?

1/6TH SCALE COLLECTIBLE FIGURE
TERMINATOR 2™
JUDGMENT DAY
T-1000
Hot Toys
prototype shown, final product may be slightly different
T2 is a Registered Trademark of StudioCanal S.A.
SIDESHOW
COLLECTIBLES™

Cross Site Scripting Vulnerability

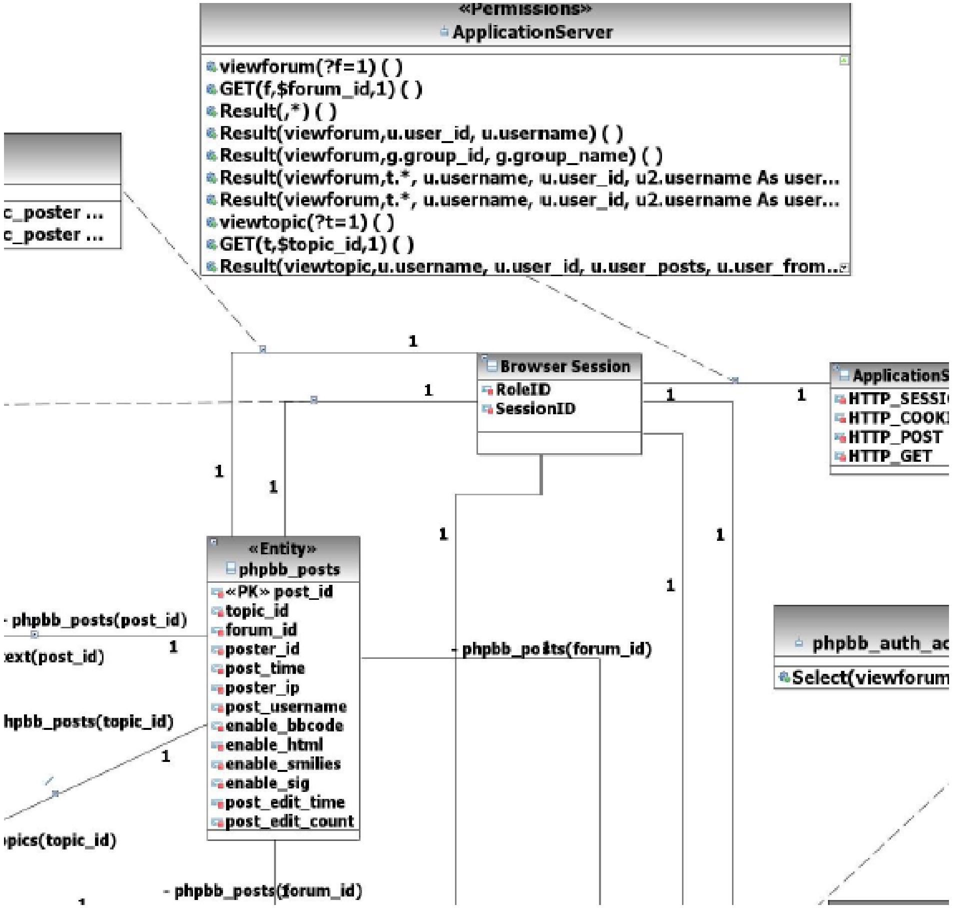
Exploit: GET /viewResults.php HTTP/1.1

Code: print $question . "<BR>";

Condition

(mysql_fetch_assoc(mysql_query(('SELECT * FROM tblquestions, answers WHERE tblquestions.QID = answers.QID AND answers.QID = \'' . $_GET['h1'] . '\'')))['Question'] === '<script>alert(1)</script>')

SECOND CHANCE!



«Permissions»
ApplicationServer

viewforum(?f=1) ( )
GET(f,$forum_id,1) ( )
Result(,*) ( )
Result(viewforum,u.user_id, u.username) ( )
Result(viewforum,g.group_id, g.group_name) ( )
Result(viewforum,t.*, u.username, u.user_id, u2.username As user...
Result(viewforum,t.*, u.username, u.user_id, u2.username As user...
viewtopic(?t=1) ( )
GET(t,$topic_id,1) ( )
Result(viewtopic,u.username, u.user_id, u.user_posts, u.user_from...

c_poster ...
c_poster ...

Browser Session
RoleID
SessionID

ApplicationS
HTTP_SESSI
HTTP_COOK
HTTP_POST
HTTP_GET

«Entity»
phpbb_posts
«PK» post_id
topic_id
forum_id
poster_id
post_time
poster_ip
post_username
enable_bbcode
enable_html
enable_smilies
enable_sig
post_edit_time
post_edit_count

phpbb_auth_ac
Select(viewforum

CONCLUSIONS

Exploit generation can improve .AST

- Reduce false positive
- Add transparency
- Helps o hack stuff

Condition resolver can help do detect

- Authentication condition and access control issues
- Hidden execution paths (e.g. backdoors)
- Hardcoded conditions

Combination of symbolic and real execution is useful

- Reduce labor input
- Improve performance
- Helps to balance CPU/time/memory

RELATED WORKS

Chandrasekhar Boyapati, Paul Darga. Eficient software model checking of data structure properties.

Keshav Pingali, Micah Beck, Richard Johnson, Mayan Moudgill, and Paul Stodghill. Dependence flow graphs: an algebraic approach to program dependencies.

E. Morel and C. Renvoise. Global optimization by suppression of partial redundancies.

R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman, and F. K. Zadeck. An eficient method of computing static single assignment form.

Vugranam C. Sreedhar and Guang R. Gao. Computing u-nodes in linear time using dj-graphs

Mark N. Wegman and F. Kenneth Zadeck. Constant propagation with conditional branches.

Ron K. Cytron and Jeanne Ferrante. Eficiently computing u-nodes on-the-fly.

Thomas Ball, Rupak Majumdar, Todd D. Millstein, and Sriram K. Rajamani. Automatic predicate abstraction of c programs.

Thomas Ball and Sriram K. Rajamani. Bebop: A symbolic model checker for boolean programs.

David Binkley. Interprocedural constant propagation using dependence graphs and a data-flow model.

AEG

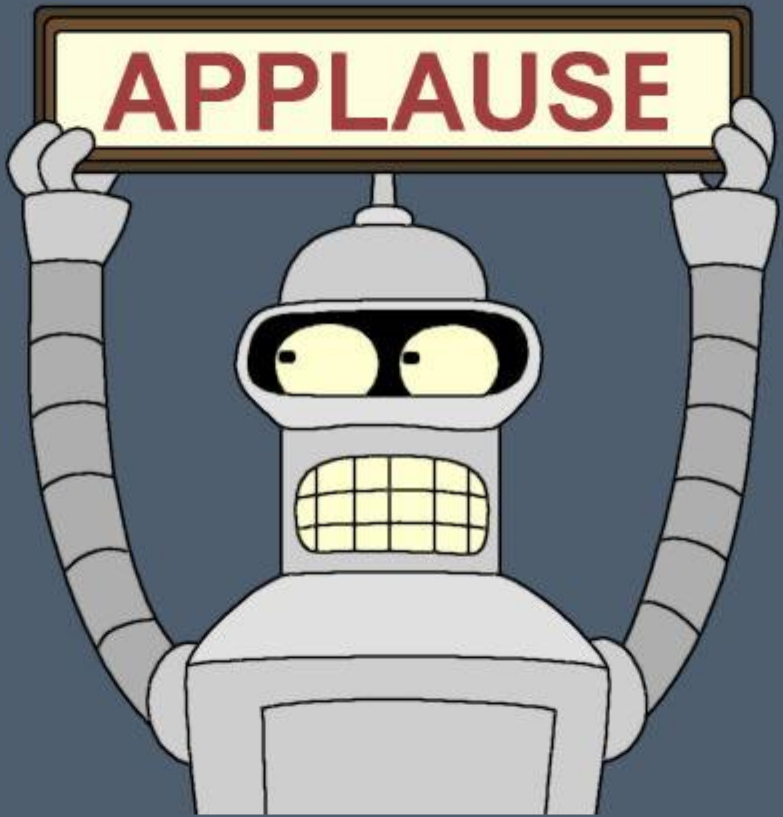
MAYHEM

The Essence of Command Injection Attacks in Web Applications, http://www.cs.ucdavis.edu/~su/publications/popl06.pdf

http://qspace.library.queensu.ca/bitstream/1974/5651/3/Alalfi_Manar_H_2010April_PhD.pdf

SPECIAL THANKS

PT



AI FOR HACKER

Automatic Exploit Generation for Application Source Code Analysis