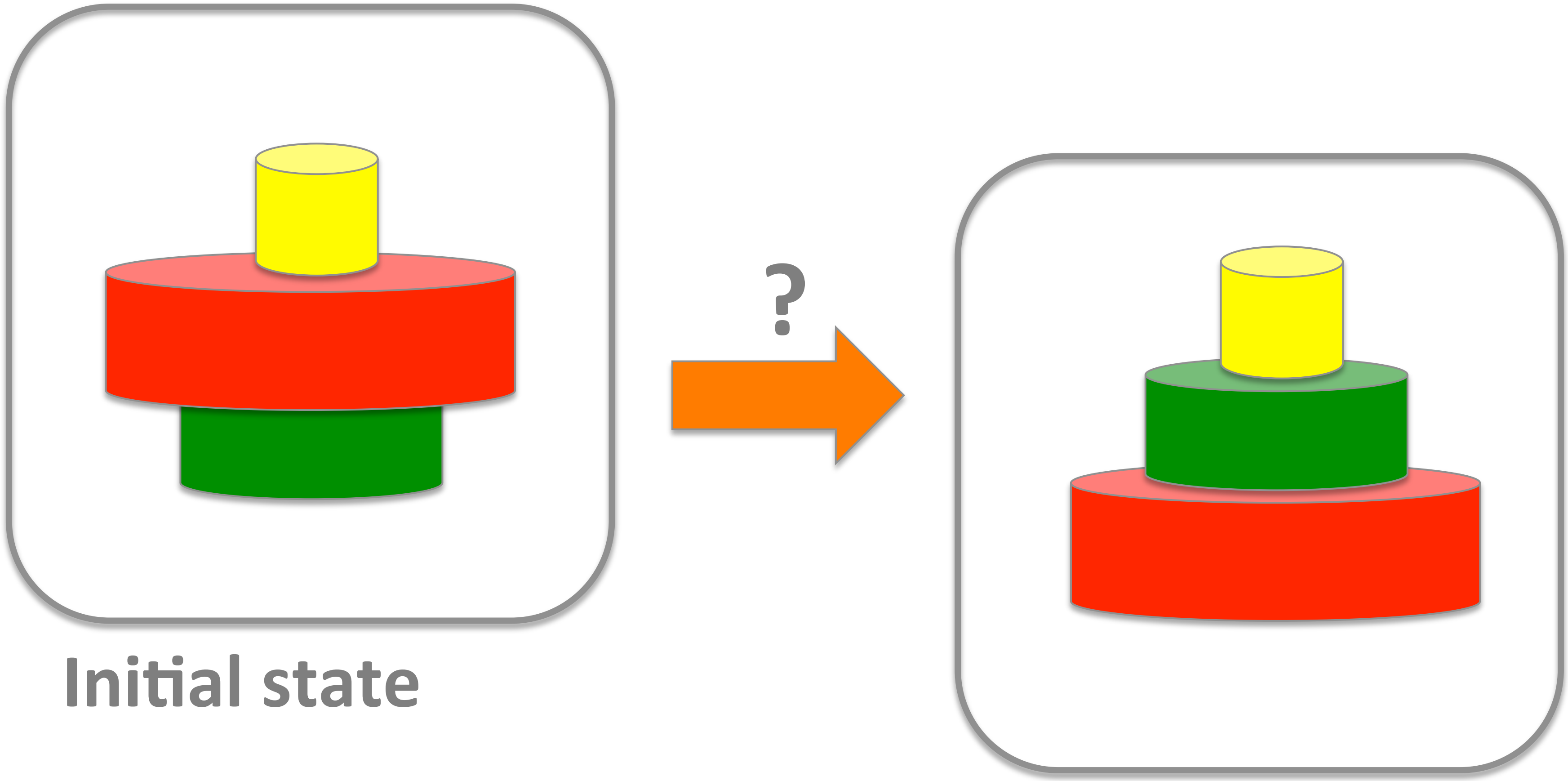
Planning in testing

Franz Wotawa
TU Graz, Institute for Software Technology
wotawa@ist.tugraz.at

Content

- **PART 1: First steps/underlying ideas**
  - AI planning
  - Pre & post conditions
  - Bringing all together
- **PART 2: AI planning in security testing**

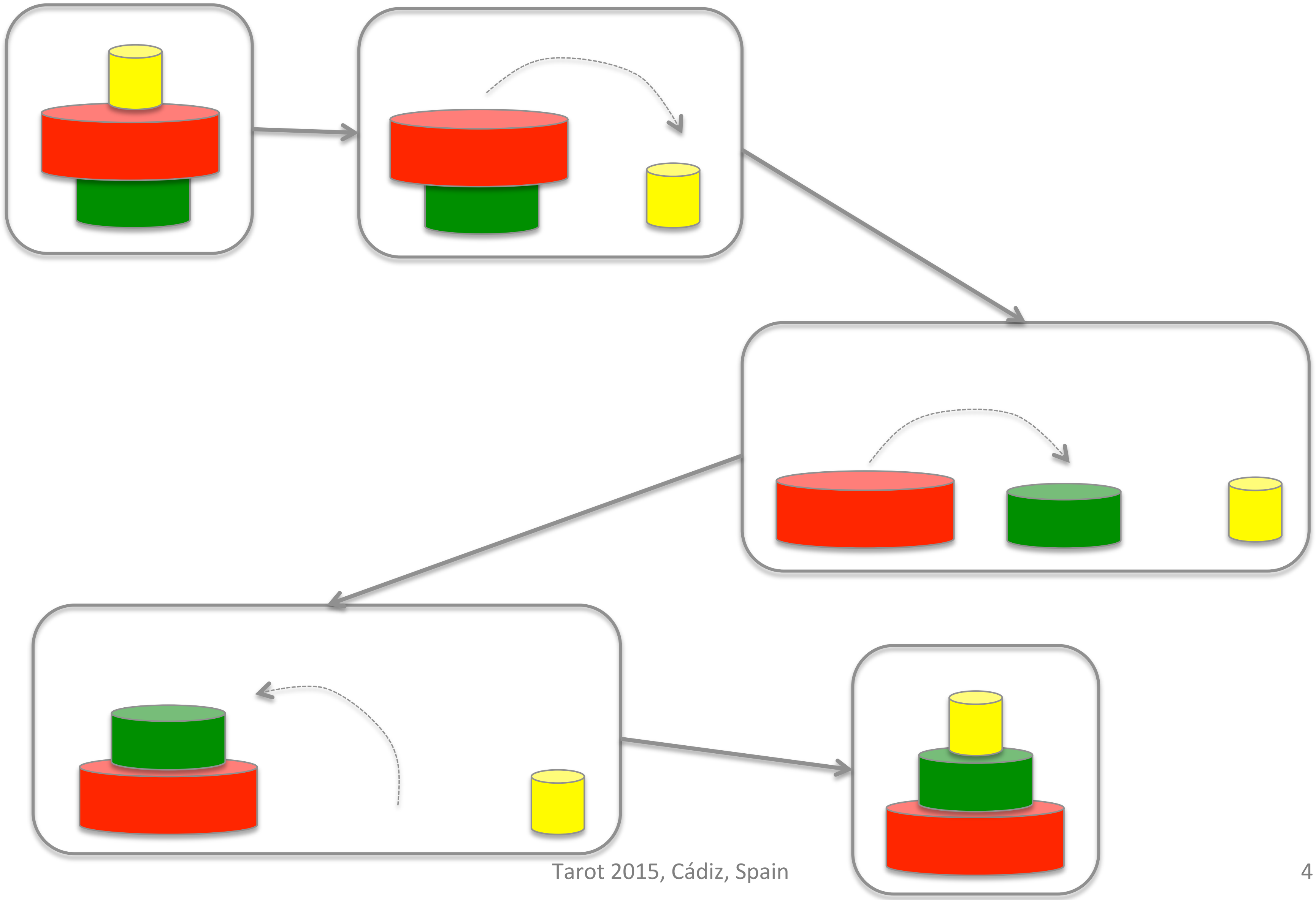
The planning problem



Initial state

Goal state

The solution – the plan

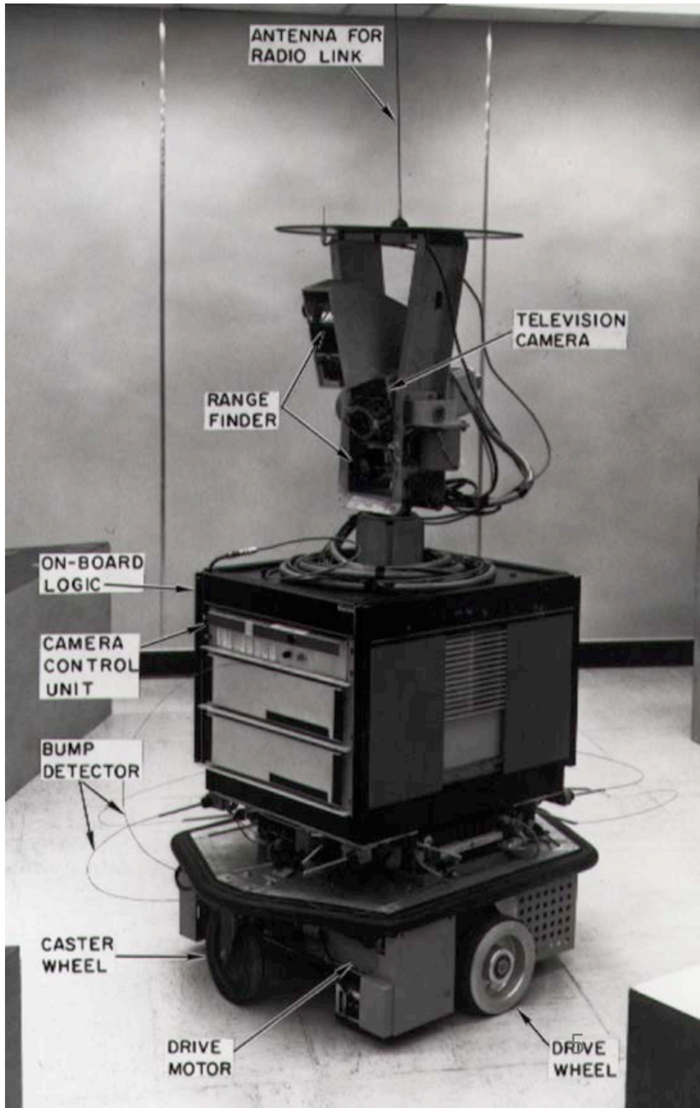


The planning problem - origins

- First formalization and use:

  Richard E. Fikes, Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". Artificial Intelligence 2 (3–4): 189–208, 1971, doi:10.1016/0004-3702(71)90010-5

- Shakey the Robot, 1966-1972, SRI International



Definition of a planning problem

- A planning problem is a tuple *(I, G, A)* where
  - *I* is a description of the initial state
  - *G* is the goal state
  - and *A* is the set of actions comprising an action name, a *precondition* under which an action can be applied and its *effects*.

- A solution of a planning problem is a sequence of actions leading from the initial state to the goal state.

Notes

- *I*, *G*, *A* are usually described in a formal language, i.e., propositional or first-order logic.

- In STRIPS:
  - *I* is a set of ground literals
  - *G* is a conjunction of literals
  - *A* has conjunctive precondition and effect

Notes (cont.)

- Restricting assumptions:
  - Atomic time
  - No exogenous events
  - Deterministic action effects
  - Omni-science on part of the agent
- The ***Closed World Assumption*** states, that if a term can not be proven to be true, it can be considered false. If a planner uses the Closed World Assumption only terms need to be given that are true in the current state. This reduces the amount of statements within planning domain descriptions.
- The **Frame Problem** describes the problem that machines do not know which conditions change if they are not explicitly stated. Planner and similar technologies have to know how to deal with unconstrained variables in order to calculate the next state. A very common interpretation therefore is, that all unconstrained variables keep their value and all state updates have to be explicitly stated in the specification

- How to combine testing and planning?
- Use specification knowledge!
- **Example**: Stack implementation in Java with pre and post conditions

Class Stack

```
@Model(name="mSize", type="int")
@Invariant("size()>=0")
class Stack {
    private int size_;

    @Post("mSize=@Old(mSize)+1")
    void push(double elem) { ... }

    @Pre("size()>0")
    @Post("size()=@Old(size())-1")
    void pop() { ... }

    @Pure
    @Post("@Return=mSize ^ @Return>=0")
    int size() { ... }
    ......
}
```

Pre condition

Post condition

Conversion to planning problem

```
@Model(name="mSize", type="int")
@Invariant("size()>=0")
class Stack {
    private int size_;

    @Post("mSize=@Old(mSize)+1")
    void push(double elem) { ... }

    @Pre("size()>0")
    @Post("size()=@Old(size())-1")
    void pop() { ... }

    @Pure
    @Post("@Return=mSize ^ @Return>=0")
    int size() { ... }
    .....
}
```

```
(:action java_util_stack
     :precondition (not (stack_instantiated))
     :effect (and (stack_instantiated)
                  (assign (size) 0.0)))

(:action push_int
     :precondition (stack_instantiated)
     :effect (increase (size) 1.0))

(:action pop
     :precondition (and  (stack_instantiated)
                         (> (size) 0.0))
     :effect (decrease (size) 1.0))
```

Example (cont.)

- Let us create a stack of size 2!

- *I*: (:init (= (size) -1.0))
- *G*: (:goal (and (= (size) 2.0)))

- Results in plan:
  - 0: JAVA_UTIL_STACK
  - 1: PUSH_INT
  - 2: PUSH_INT

Important to know:

- Planning domain (= action definitions) can be obtained automatically from the source code!
  - Stefan J. Galler, Automatic Object Type Test Input Data Generation for Java Programs based on Design by Contract Specification, Ph.D. Thesis, TU Graz, 2011

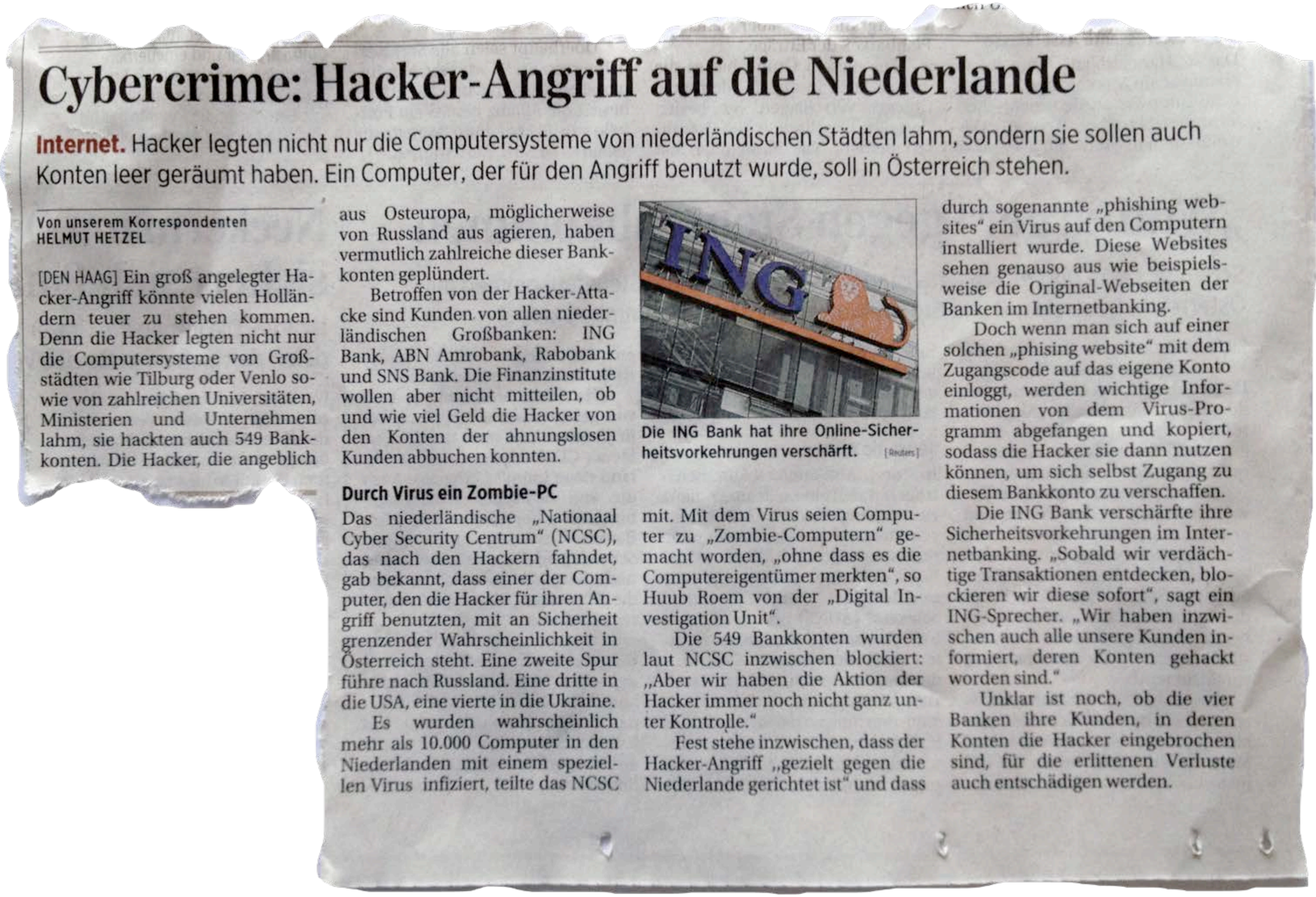
- Planning problem (*I*, *G*,*A*) has to be stated in order to get the required test data!

Related publications

- Andreas Leitner, Strategies to Automatically Test Eiffel Programs, Master's Thesis, TU Graz, 2004
- Leitner, A. and Bloem, R. (2005). Automatic Testing through Planning. Technical report, Graz University of Technology.
- Zehentner, C. (2010). Planning4ObjectCreation: An AI-Planning System for Test Data Generation. Master's thesis, Graz University of Technology.
- Galler, S. J., Zehentner, C., and Wotawa, F. (2010c). AIana: An AI Planning System for Test Data Generation. In 1st Workshop on Testing Object-Oriented Software Systems, pages 30–37, Maribor, Slovenja.

SECURITY TESTING

Are we safe?



Cybercrime: Hacker-Angriff auf die Niederlande

Internet. Hacker legten nicht nur die Computersysteme von niederländischen Städten lahm, sondern sie sollen auch Konten leer geräumt haben. Ein Computer, der für den Angriff benutzt wurde, soll in Österreich stehen.

Von unserem Korrespondenten HELMUT HETZEL

[DEN HAAG] Ein groß angelegter Hacker-Angriff könnte vielen Holländern teuer zu stehen kommen. Denn die Hacker legten nicht nur die Computersysteme von Großstädten wie Tilburg oder Venlo sowie von zahlreichen Universitäten, Ministerien und Unternehmen lahm, sie hackten auch 549 Bankkonten. Die Hacker, die angeblich aus Osteuropa, möglicherweise von Russland aus agieren, haben vermutlich zahlreiche dieser Bankkonten geplündert.

Betroffen von der Hacker-Attacke sind Kunden von allen niederländischen Großbanken: ING Bank, ABN Amrobank, Rabobank und SNS Bank. Die Finanzinstitute wollen aber nicht mitteilen, ob und wie viel Geld die Hacker von den Konten der ahnungslosen Kunden abbuchen konnten.

Die ING Bank hat ihre Online-Sicherheitsvorkehrungen verschärft. [Reuters]

Durch Virus ein Zombie-PC

Das niederländische „Nationaal Cyber Security Centrum" (NCSC), das nach den Hackern fahndet, gab bekannt, dass einer der Computer, den die Hacker für ihren Angriff benutzten, mit an Sicherheit grenzender Wahrscheinlichkeit in Österreich steht. Eine zweite Spur führe nach Russland. Eine dritte in die USA, eine vierte in die Ukraine.

Es wurden wahrscheinlich mehr als 10.000 Computer in den Niederlanden mit einem speziellen Virus infiziert, teilte das NCSC mit. Mit dem Virus seien Computer zu „Zombie-Computern" gemacht worden, „ohne dass es die Computereigentümer merkten", so Huub Roem von der „Digital Investigation Unit".

Die 549 Bankkonten wurden laut NCSC inzwischen blockiert: „Aber wir haben die Aktion der Hacker immer noch nicht ganz unter Kontrolle."

Fest stehe inzwischen, dass der Hacker-Angriff „gezielt gegen die Niederlande gerichtet ist" und dass durch sogenannte „phishing websites" ein Virus auf den Computern installiert wurde. Diese Websites sehen genauso aus wie beispielsweise die Original-Webseiten der Banken im Internetbanking.

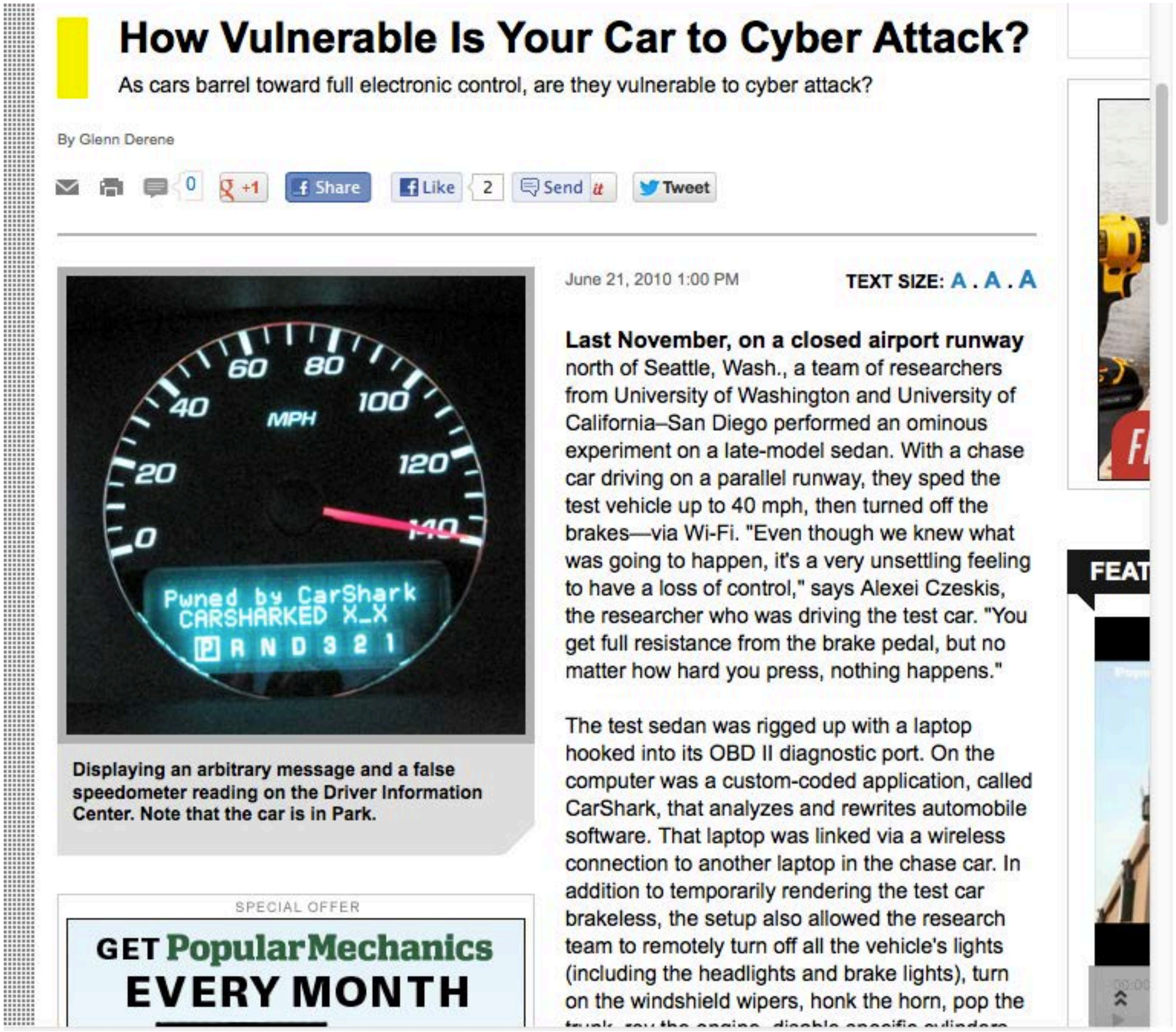
Doch wenn man sich auf einer solchen „phising website" mit dem Zugangscode auf das eigene Konto einloggt, werden wichtige Informationen von dem Virus-Programm abgefangen und kopiert, sodass die Hacker sie dann nutzen können, um sich selbst Zugang zu diesem Bankkonto zu verschaffen.

Die ING Bank verschärfte ihre Sicherheitsvorkehrungen im Internetbanking. „Sobald wir verdächtige Transaktionen entdecken, blockieren wir diese sofort", sagt ein ING-Sprecher. „Wir haben inzwischen auch alle unsere Kunden informiert, deren Konten gehackt worden sind."

Unklar ist noch, ob die vier Banken ihre Kunden, in deren Konten die Hacker eingebrochen sind, für die erlittenen Verluste auch entschädigen werden.

Die Presse, 14. August 2012

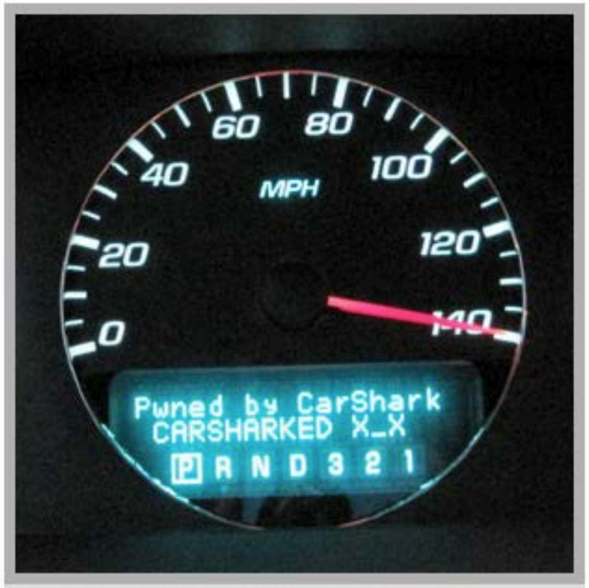
But cars are safe!



How Vulnerable Is Your Car to Cyber Attack?

As cars barrel toward full electronic control, are they vulnerable to cyber attack?

By Glenn Derene



Displaying an arbitrary message and a false speedometer reading on the Driver Information Center. Note that the car is in Park.

June 21, 2010 1:00 PM

TEXT SIZE: A . A . A

Last November, on a closed airport runway north of Seattle, Wash., a team of researchers from University of Washington and University of California–San Diego performed an ominous experiment on a late-model sedan. With a chase car driving on a parallel runway, they sped the test vehicle up to 40 mph, then turned off the brakes—via Wi-Fi. "Even though we knew what was going to happen, it's a very unsettling feeling to have a loss of control," says Alexei Czeskis, the researcher who was driving the test car. "You get full resistance from the brake pedal, but no matter how hard you press, nothing happens."

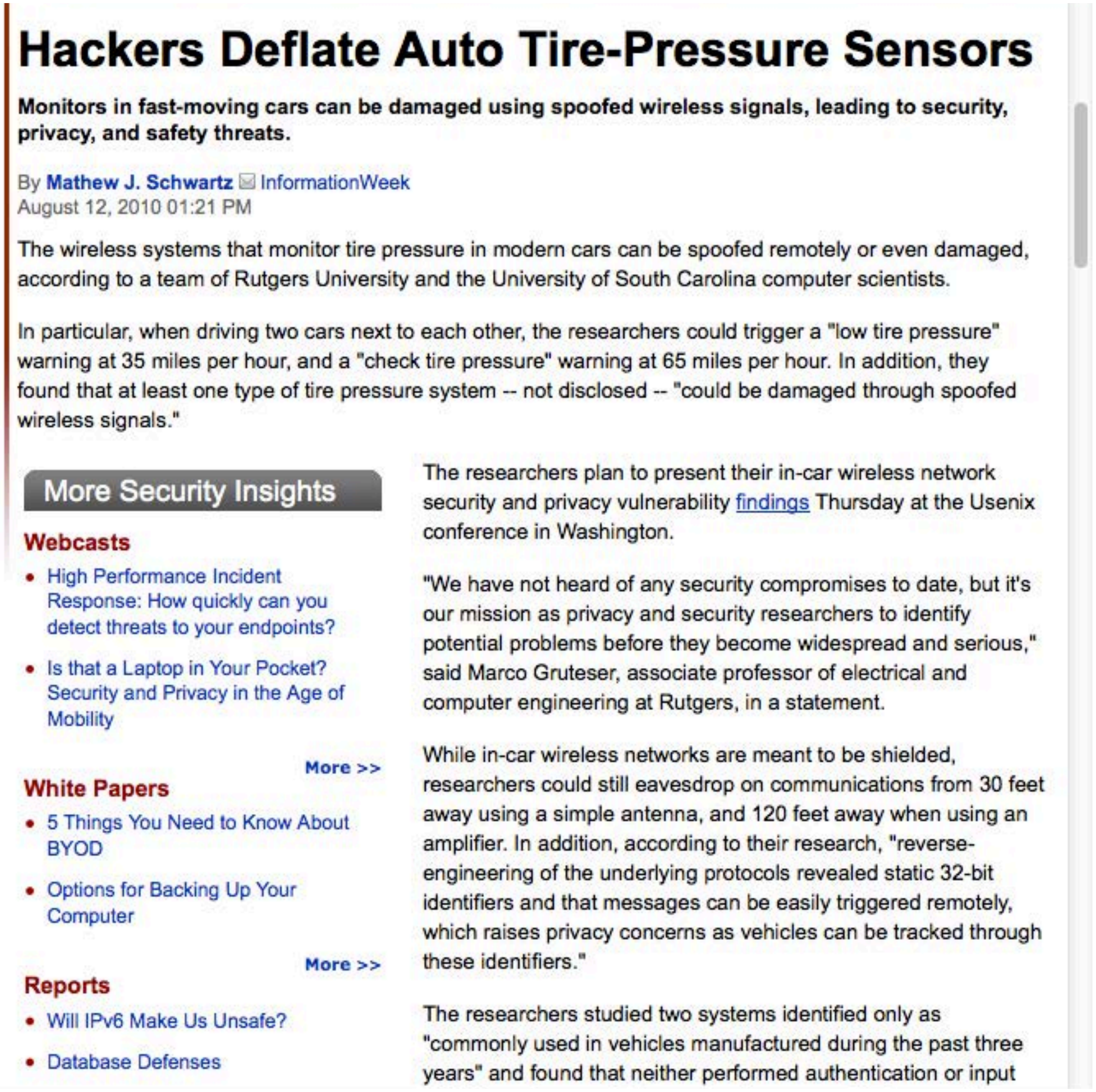
The test sedan was rigged up with a laptop hooked into its OBD II diagnostic port. On the computer was a custom-coded application, called CarShark, that analyzes and rewrites automobile software. That laptop was linked via a wireless connection to another laptop in the chase car. In addition to temporarily rendering the test car brakeless, the setup also allowed the research team to remotely turn off all the vehicle's lights (including the headlights and brake lights), turn on the windshield wipers, honk the horn, pop the trunk, rev the engine, disable specific cylinders

SPECIAL OFFER

GET Popular Mechanics EVERY MONTH

FEAT

Popular Mechanics, 2010



Hackers Deflate Auto Tire-Pressure Sensors

Monitors in fast-moving cars can be damaged using spoofed wireless signals, leading to security, privacy, and safety threats.

By **Mathew J. Schwartz** InformationWeek
August 12, 2010 01:21 PM

The wireless systems that monitor tire pressure in modern cars can be spoofed remotely or even damaged, according to a team of Rutgers University and the University of South Carolina computer scientists.

In particular, when driving two cars next to each other, the researchers could trigger a "low tire pressure" warning at 35 miles per hour, and a "check tire pressure" warning at 65 miles per hour. In addition, they found that at least one type of tire pressure system -- not disclosed -- "could be damaged through spoofed wireless signals."

The researchers plan to present their in-car wireless network security and privacy vulnerability findings Thursday at the Usenix conference in Washington.

"We have not heard of any security compromises to date, but it's our mission as privacy and security researchers to identify potential problems before they become widespread and serious," said Marco Gruteser, associate professor of electrical and computer engineering at Rutgers, in a statement.

While in-car wireless networks are meant to be shielded, researchers could still eavesdrop on communications from 30 feet away using a simple antenna, and 120 feet away when using an amplifier. In addition, according to their research, "reverse-engineering of the underlying protocols revealed static 32-bit identifiers and that messages can be easily triggered remotely, which raises privacy concerns as vehicles can be tracked through these identifiers."

The researchers studied two systems identified only as "commonly used in vehicles manufactured during the past three years" and found that neither performed authentication or input

More Security Insights

Webcasts

- High Performance Incident Response: How quickly can you detect threats to your endpoints?
- Is that a Laptop in Your Pocket? Security and Privacy in the Age of Mobility

More >>

White Papers

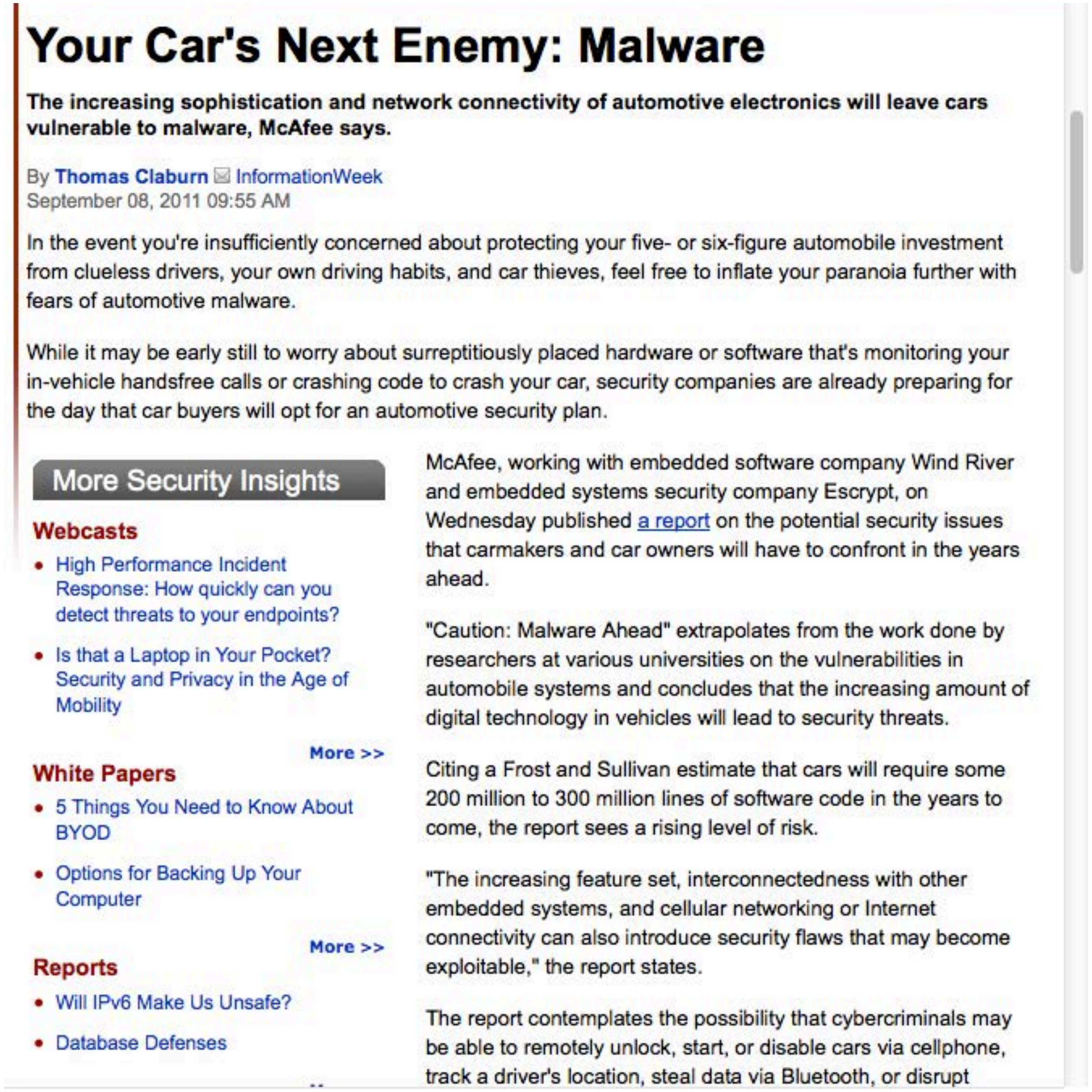
- 5 Things You Need to Know About BYOD
- Options for Backing Up Your Computer

More >>

Reports

- Will IPv6 Make Us Unsafe?
- Database Defenses

Information Week, 2010



Your Car's Next Enemy: Malware

The increasing sophistication and network connectivity of automotive electronics will leave cars vulnerable to malware, McAfee says.

By **Thomas Claburn** ✉ InformationWeek
September 08, 2011 09:55 AM

In the event you're insufficiently concerned about protecting your five- or six-figure automobile investment from clueless drivers, your own driving habits, and car thieves, feel free to inflate your paranoia further with fears of automotive malware.

While it may be early still to worry about surreptitiously placed hardware or software that's monitoring your in-vehicle handsfree calls or crashing code to crash your car, security companies are already preparing for the day that car buyers will opt for an automotive security plan.

McAfee, working with embedded software company Wind River and embedded systems security company Escrypt, on Wednesday published a report on the potential security issues that carmakers and car owners will have to confront in the years ahead.

"Caution: Malware Ahead" extrapolates from the work done by researchers at various universities on the vulnerabilities in automobile systems and concludes that the increasing amount of digital technology in vehicles will lead to security threats.

Citing a Frost and Sullivan estimate that cars will require some 200 million to 300 million lines of software code in the years to come, the report sees a rising level of risk.

"The increasing feature set, interconnectedness with other embedded systems, and cellular networking or Internet connectivity can also introduce security flaws that may become exploitable," the report states.

The report contemplates the possibility that cybercriminals may be able to remotely unlock, start, or disable cars via cellphone, track a driver's location, steal data via Bluetooth, or disrupt

More Security Insights

Webcasts

- High Performance Incident Response: How quickly can you detect threats to your endpoints?
- Is that a Laptop in Your Pocket? Security and Privacy in the Age of Mobility

More >>

White Papers

- 5 Things You Need to Know About BYOD
- Options for Backing Up Your Computer

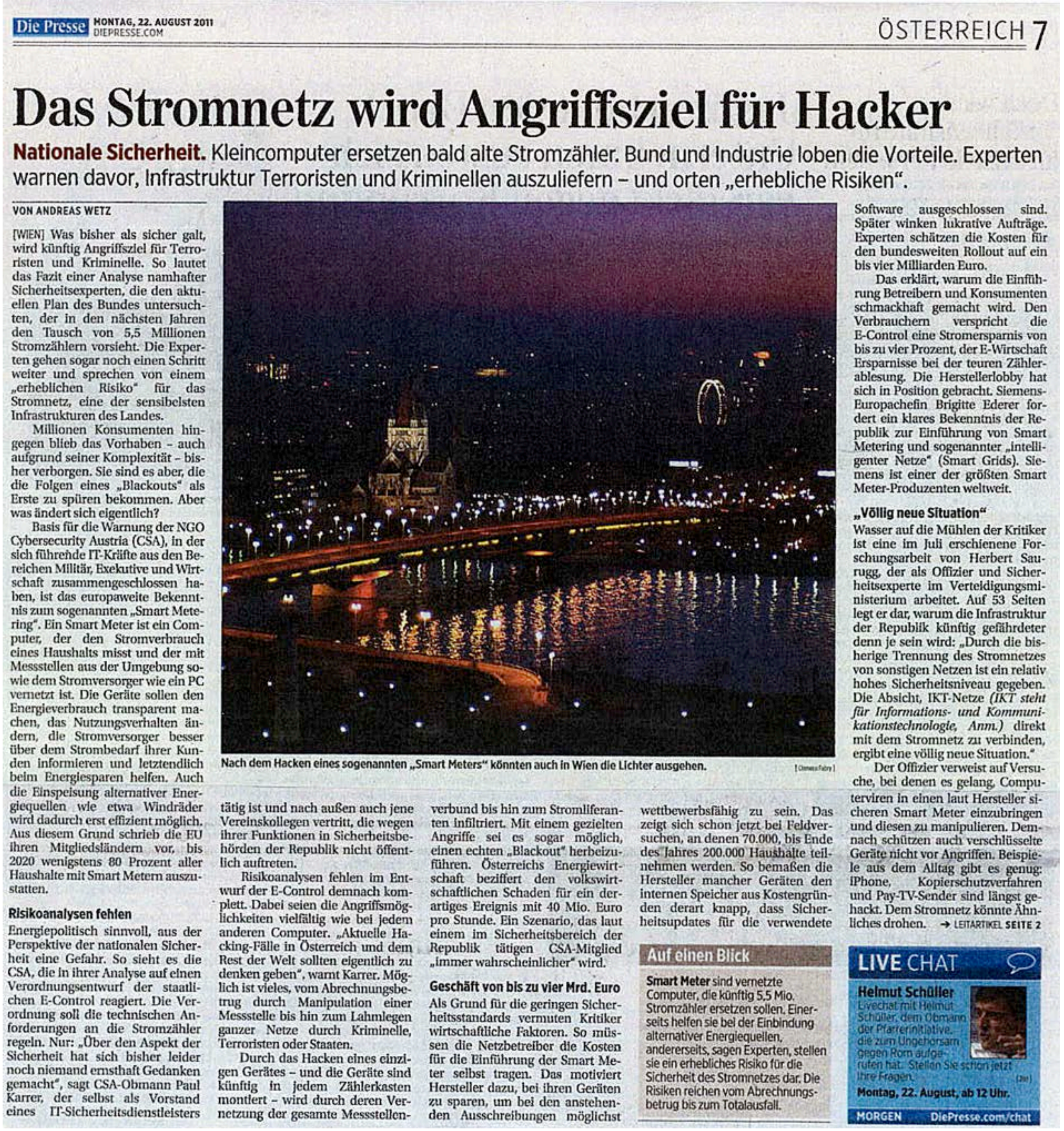
More >>

Reports

- Will IPv6 Make Us Unsafe?
- Database Defenses

Information Week, 2011

Ok! But our infrastructure is safe!?



Die Presse MONTAG, 22. AUGUST 2011
DIEPRESSE.COM

ÖSTERREICH 7

Das Stromnetz wird Angriffsziel für Hacker

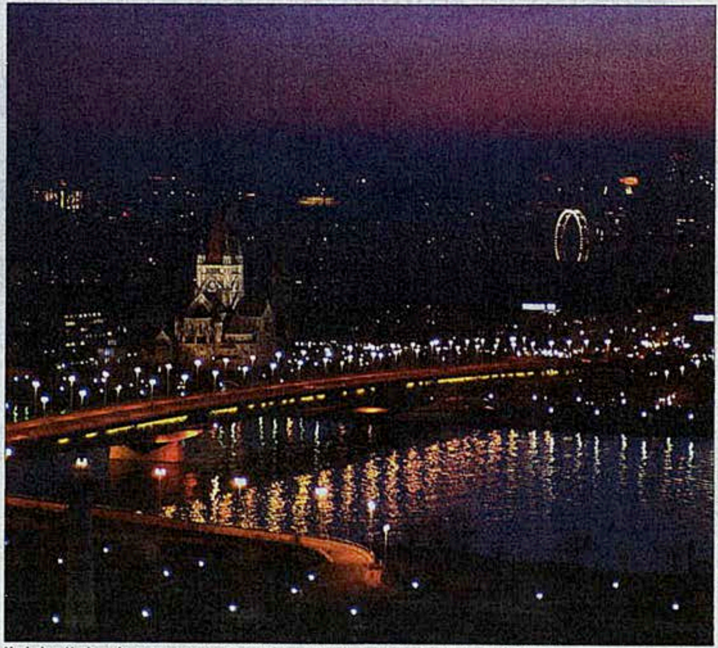
Nationale Sicherheit. Kleincomputer ersetzen bald alte Stromzähler. Bund und Industrie loben die Vorteile. Experten warnen davor, Infrastruktur Terroristen und Kriminellen auszuliefern – und orten „erhebliche Risiken“.

VON ANDREAS WETZ

[WIEN] Was bisher als sicher galt, wird künftig Angriffsziel für Terroristen und Kriminelle. So lautet das Fazit einer Analyse namhafter Sicherheitsexperten, die den aktuellen Plan des Bundes untersuchten, der in den nächsten Jahren den Tausch von 5,5 Millionen Stromzählern vorsieht. Die Experten gehen sogar noch einen Schritt weiter und sprechen von einem „erheblichen Risiko“ für das Stromnetz, eine der sensibelsten Infrastrukturen des Landes.

Millionen Konsumenten hingegen blieb das Vorhaben – auch aufgrund seiner Komplexität – bisher verborgen. Sie sind es aber, die die Folgen eines „Blackouts“ als Erste zu spüren bekommen. Aber was ändert sich eigentlich?

Basis für die Warnung der NGO Cybersecurity Austria (CSA), in der sich führende IT-Kräfte aus den Bereichen Militär, Exekutive und Wirtschaft zusammengeschlossen haben, ist das europaweite Bekenntnis zum sogenannten „Smart Metering“. Ein Smart Meter ist ein Computer, der den Stromverbrauch eines Haushalts misst und der mit Messstellen aus der Umgebung sowie dem Stromversorger wie ein PC vernetzt ist. Die Geräte sollen den Energieverbrauch transparent machen, das Nutzungsverhalten ändern, die Stromversorger besser über dem Strombedarf ihrer Kunden informieren und letztendlich beim Energiesparen helfen. Auch die Einspeisung alternativer Energiequellen wie etwa Windräder wird dadurch erst effizient möglich. Aus diesem Grund schrieb die EU ihren Mitgliedsländern vor, bis 2020 wenigstens 80 Prozent aller Haushalte mit Smart Metern auszustatten.



Nach dem Hacken eines sogenannten „Smart Meters“ könnten auch in Wien die Lichter ausgehen. [Clemens Fabry]

Risikoanalysen fehlen

Energiepolitisch sinnvoll, aus der Perspektive der nationalen Sicherheit eine Gefahr. So sieht es die CSA, die in ihrer Analyse auf einen Verordnungsentwurf der staatlichen E-Control reagiert. Die Verordnung soll die technischen Anforderungen an die Stromzähler regeln. Nur: „Über den Aspekt der Sicherheit hat sich bisher leider noch niemand ernsthaft Gedanken gemacht“, sagt CSA-Obmann Paul Karrer, der selbst als Vorstand eines IT-Sicherheitsdienstleisters tätig ist und nach außen auch jene Vereinskollegen vertritt, die wegen ihrer Funktionen in Sicherheitsbehörden der Republik nicht öffentlich auftreten.

Risikoanalysen fehlen im Entwurf der E-Control demnach komplett. Dabei seien die Angriffsmöglichkeiten vielfältig wie bei jedem anderen Computer. „Aktuelle Hacking-Fälle in Österreich und dem Rest der Welt sollten eigentlich zu denken geben“, warnt Karrer. Möglich ist vieles, vom Abrechnungsbetrug durch Manipulation einer Messstelle bis hin zum Lahmlegen ganzer Netze durch Kriminelle, Terroristen oder Staaten.

Durch das Hacken eines einzigen Gerätes – und die Geräte sind künftig in jedem Zählerkasten montiert – wird durch deren Vernetzung der gesamte Messstellenverbund bis hin zum Stromliferanten infiltriert. Mit einem gezielten Angriffe sei es sogar möglich, einen echten „Blackout“ herbeizuführen. Österreichs Energiewirtschaft beziffert den volkswirtschaftlichen Schaden für ein derartiges Ereignis mit 40 Mio. Euro pro Stunde. Ein Szenario, das laut einem im Sicherheitsbereich der Republik tätigen CSA-Mitglied „immer wahrscheinlicher“ wird.

Geschäft von bis zu vier Mrd. Euro

Als Grund für die geringen Sicherheitsstandards vermuten Kritiker wirtschaftliche Faktoren. So müssen die Netzbetreiber die Kosten für die Einführung der Smart Meter selbst tragen. Das motiviert Hersteller dazu, bei ihren Geräten zu sparen, um bei den anstehenden Ausschreibungen möglichst wettbewerbsfähig zu sein. Das zeigt sich schon jetzt bei Feldversuchen, an denen 70.000, bis Ende des Jahres 200.000 Haushalte teilnehmen werden. So bemaßen die Hersteller mancher Geräten den internen Speicher aus Kostengründen derart knapp, dass Sicherheitsupdates für die verwendete Software ausgeschlossen sind. Später winken lukrative Aufträge. Experten schätzen die Kosten für den bundesweiten Rollout auf ein bis vier Milliarden Euro.

Das erklärt, warum die Einführung Betreibern und Konsumenten schmackhaft gemacht wird. Den Verbrauchern verspricht die E-Control eine Stromersparnis von bis zu vier Prozent, der E-Wirtschaft Ersparnisse bei der teuren Zählerablesung. Die Herstellerlobby hat sich in Position gebracht. Siemens-Europachefin Brigitte Ederer fordert ein klares Bekenntnis der Republik zur Einführung von Smart Metering und sogenannter „intelligenter Netze“ (Smart Grids). Siemens ist einer der größten Smart Meter-Produzenten weltweit.

„Völlig neue Situation“

Wasser auf die Mühlen der Kritiker ist eine im Juli erschienene Forschungsarbeit von Herbert Saurugg, der als Offizier und Sicherheitsexperte im Verteidigungsministerium arbeitet. Auf 53 Seiten legt er dar, warum die Infrastruktur der Republik künftig gefährdeter denn je sein wird: „Durch die bisherige Trennung des Stromnetzes von sonstigen Netzen ist ein relativ hohes Sicherheitsniveau gegeben. Die Absicht, IKT-Netze *(IKT steht für Informations- und Kommunikationstechnologie, Anm.)* direkt mit dem Stromnetz zu verbinden, ergibt eine völlig neue Situation.“

Der Offizier verweist auf Versuche, bei denen es gelang, Computerviren in einen laut Hersteller sicheren Smart Meter einzubringen und diesen zu manipulieren. Demnach schützen auch verschlüsselte Geräte nicht vor Angriffen. Beispiele aus dem Alltag gibt es genug: iPhone, Kopierschutzverfahren und Pay-TV-Sender sind längst gehackt. Dem Stromnetz könnte Ähnliches drohen.

→ LEITARTIKEL SEITE 2

Auf einen Blick

Smart Meter sind vernetzte Computer, die künftig 5,5 Mio. Stromzähler ersetzen sollen. Einerseits helfen sie bei der Einbindung alternativer Energiequellen, andererseits, sagen Experten, stellen sie ein erhebliches Risiko für die Sicherheit des Stromnetzes dar. Die Risiken reichen vom Abrechnungsbetrug bis zum Totalausfall.

LIVE CHAT

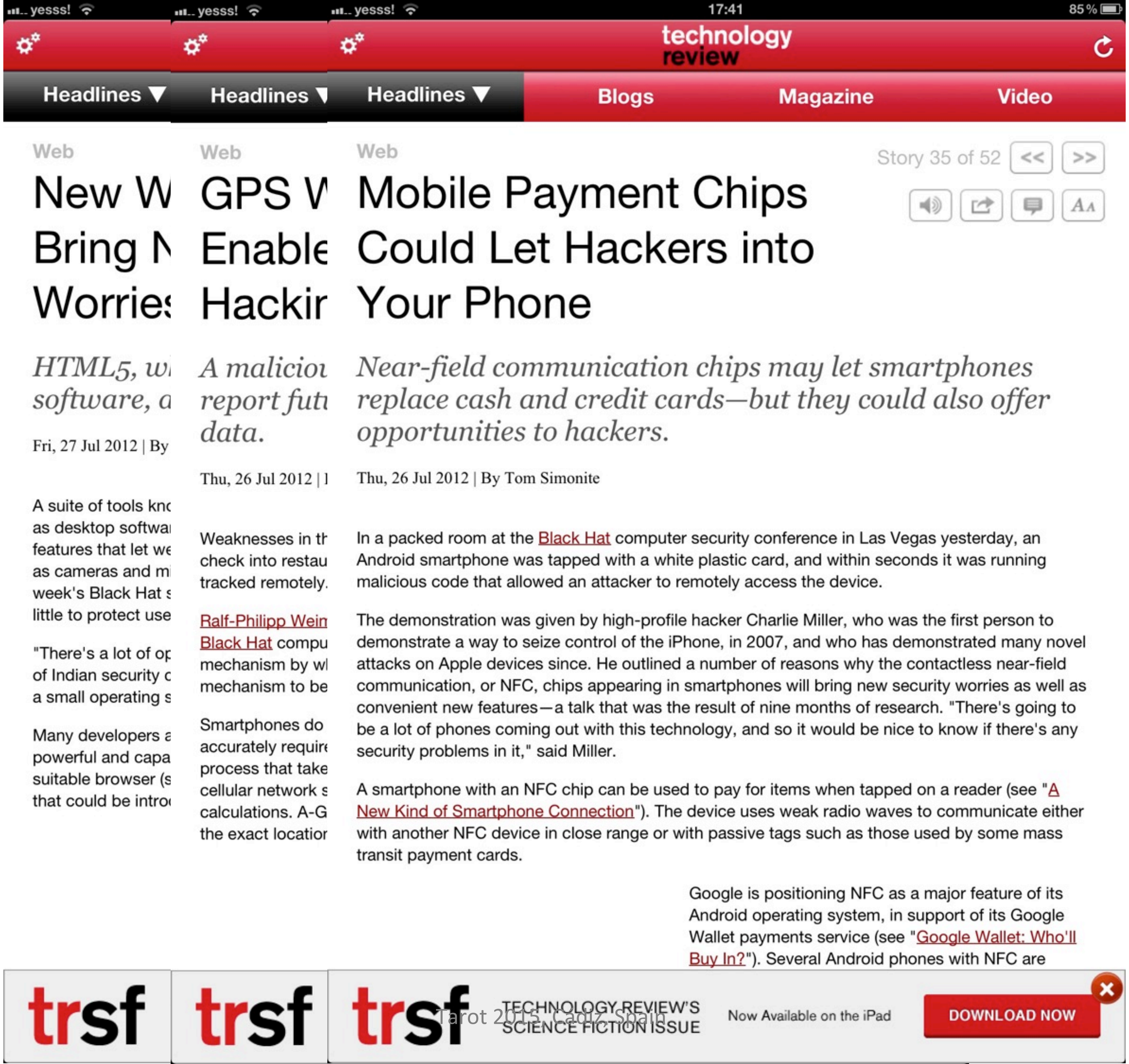
Helmut Schüller

Livechat mit Helmut Schüller, dem Obmann der Pfarrerinitiative, die zum Ungehorsam gegen Rom aufgerufen hat. Stellen Sie schon jetzt Ihre Fragen. (Die)

Montag, 22. August, ab 12 Uhr.

MORGEN DiePresse.com/chat

Die Presse, 22. August 2011



Web

New W
Bring N
Worries

HTML5, w
software, a

Fri, 27 Jul 2012 | By

A suite of tools kn
as desktop softwa
features that let we
as cameras and mi
week's Black Hat s
little to protect use

"There's a lot of op
of Indian security c
a small operating s

Many developers a
powerful and capa
suitable browser (s
that could be intro

Web

GPS W
Enable
Hackir

A maliciou
report futu
data.

Thu, 26 Jul 2012 | 1

Weaknesses in th
check into restau
tracked remotely.

Ralf-Philipp Wein
Black Hat compu
mechanism by wl
mechanism to be

Smartphones do
accurately require
process that take
cellular network s
calculations. A-G
the exact locatior

Web

Mobile Payment Chips Could Let Hackers into Your Phone

Near-field communication chips may let smartphones replace cash and credit cards—but they could also offer opportunities to hackers.

Thu, 26 Jul 2012 | By Tom Simonite

In a packed room at the Black Hat computer security conference in Las Vegas yesterday, an Android smartphone was tapped with a white plastic card, and within seconds it was running malicious code that allowed an attacker to remotely access the device.

The demonstration was given by high-profile hacker Charlie Miller, who was the first person to demonstrate a way to seize control of the iPhone, in 2007, and who has demonstrated many novel attacks on Apple devices since. He outlined a number of reasons why the contactless near-field communication, or NFC, chips appearing in smartphones will bring new security worries as well as convenient new features—a talk that was the result of nine months of research. "There's going to be a lot of phones coming out with this technology, and so it would be nice to know if there's any security problems in it," said Miller.

A smartphone with an NFC chip can be used to pay for items when tapped on a reader (see "A New Kind of Smartphone Connection"). The device uses weak radio waves to communicate either with another NFC device in close range or with passive tags such as those used by some mass transit payment cards.

Google is positioning NFC as a major feature of its Android operating system, in support of its Google Wallet payments service (see "Google Wallet: Who'll Buy In?"). Several Android phones with NFC are

- There are much more examples!
- What are the consequences?
- How can we ensure security?

A SIMPLE EXAMPLE

A simple example

- Given:
  - Web page providing the user name as argument
  - Server side script

HTML page content (Client):

```
<form action=“test.cgi“ method=GET>
<input maxlength=10 type=“input“
          name=“ARGV“>Username</input>
</form>
```

Script `test.cgi` (Server):

```
$username = $ARGV;
system(“cat /logs/$username“ . “.log“);
```

HTML page sends content:

```
http://to_server/test.cgi?username=Mary
```

But maybe also

```
http://to_server/test.cgi?
username=TOO_LONG_FOR_A_USERNAME
```

We might also use this to submit other parameters like:

```
../etc/passwd
```

```
$username = “../etc/passwd“;
system(“cat /logs/$username“ . “.log“);
```

or

```
Mary.log; rm –rf /; cat blah
```

```
$username = “Mary.log; rm –rf /; cat blah“;
system(“cat /logs/$username“ . “.log“);
```

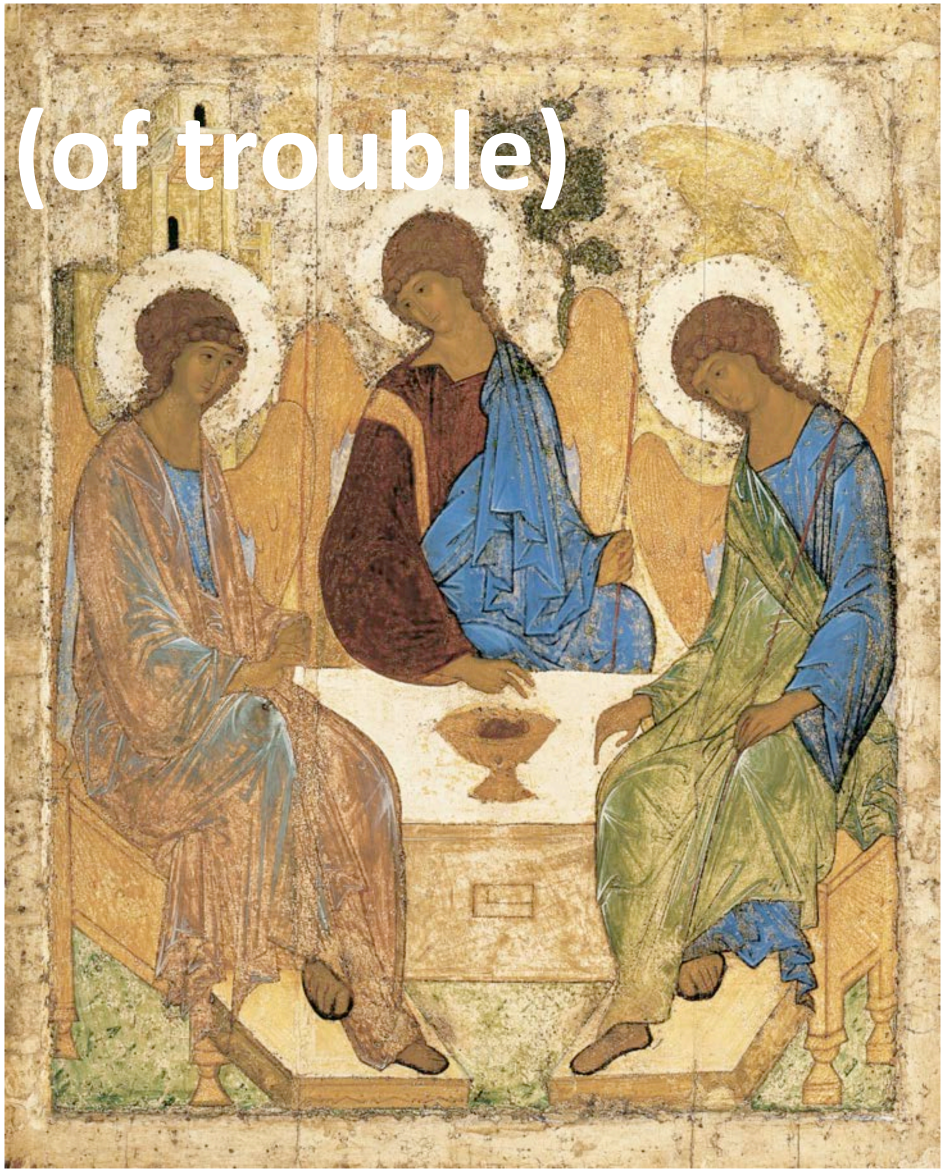
So why is this possible?

- Trusting the user input
- No check for invalide arguments
- Access to OS / system calls via script

THE (REAL) ORIGINS OF THE PROBLEM

The trinity (of trouble)

- **Complexity**
- Extensibility
- *Connectivity*


Russian icon of the Old Testament Trinity by Andrey Rublev, between 1408-25

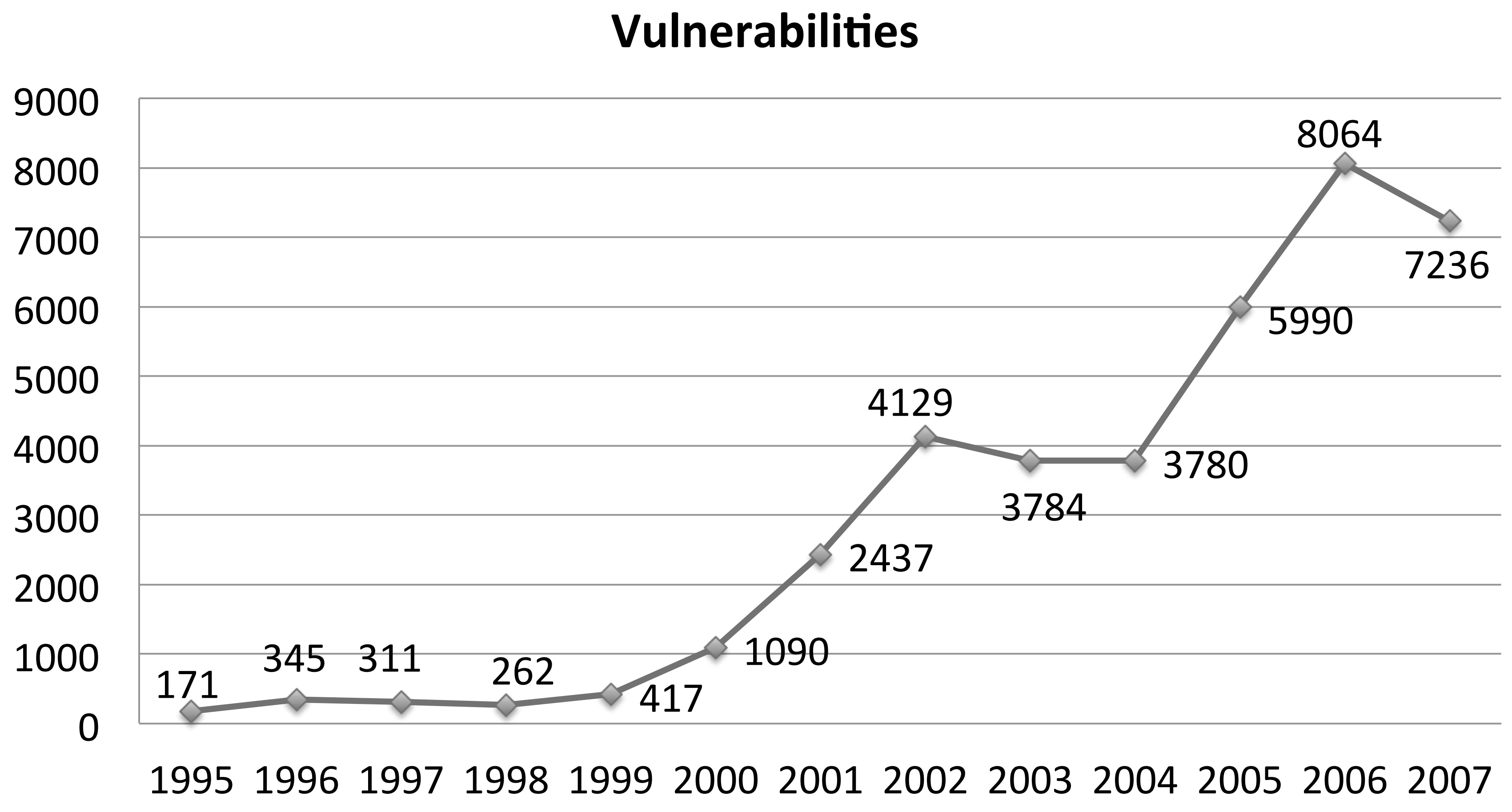
Complexity

- Solaris 7: 400,000 LoC
- Linux: 1.5 Mio LoC
- Windows 95: < 5 Mio LoC
- Windows XP: 40 Mio LoC

- **More LoC** lead to more bugs!!!

- **More vulnerabilities**!

Consequences of complexity



From http://www.cert.org/stats/

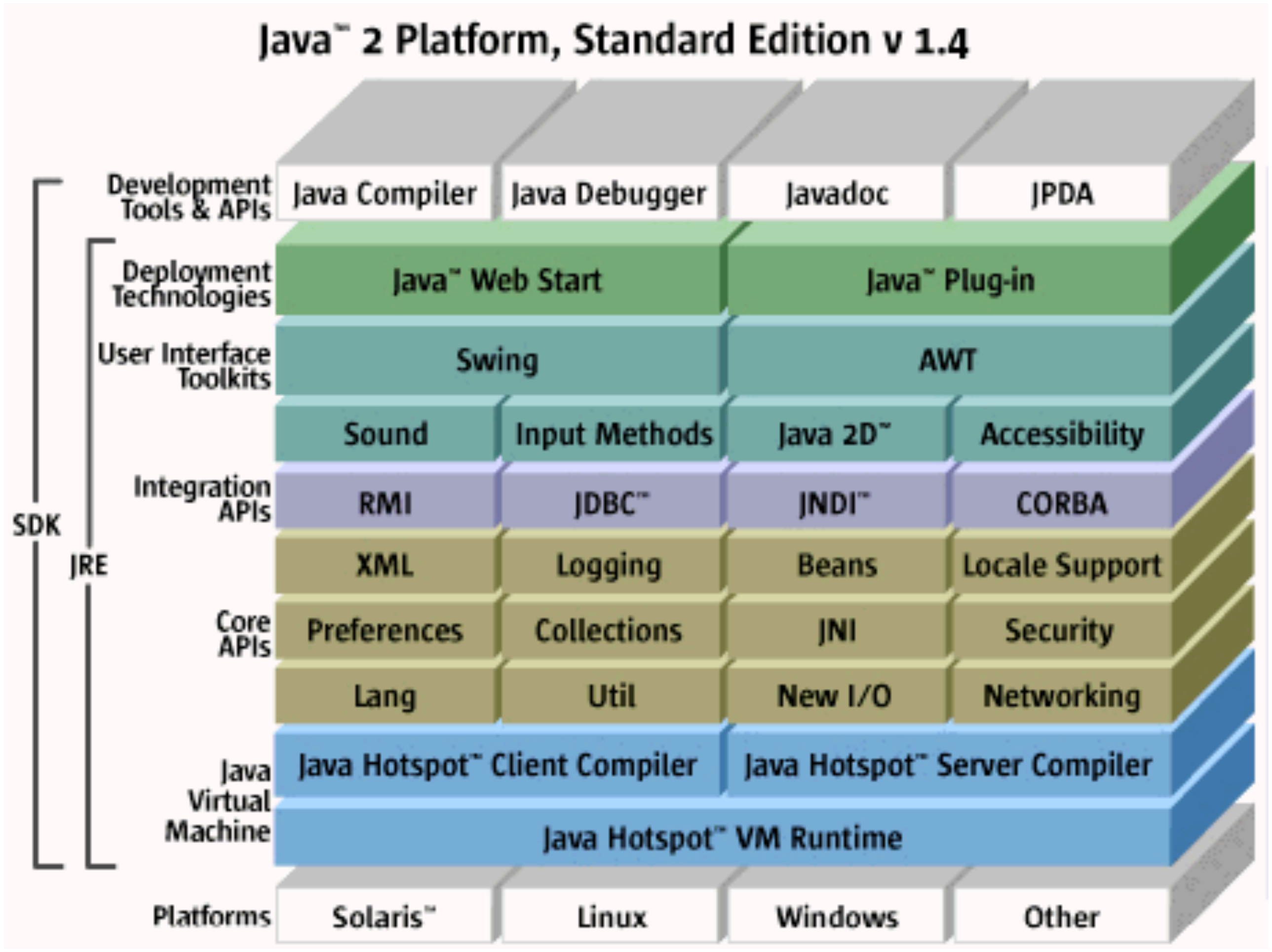
Consequences of complexity

- Software Engineering Institute: Research outline, `http://www.sei.cmu.edu/tsp/research/index.html`:

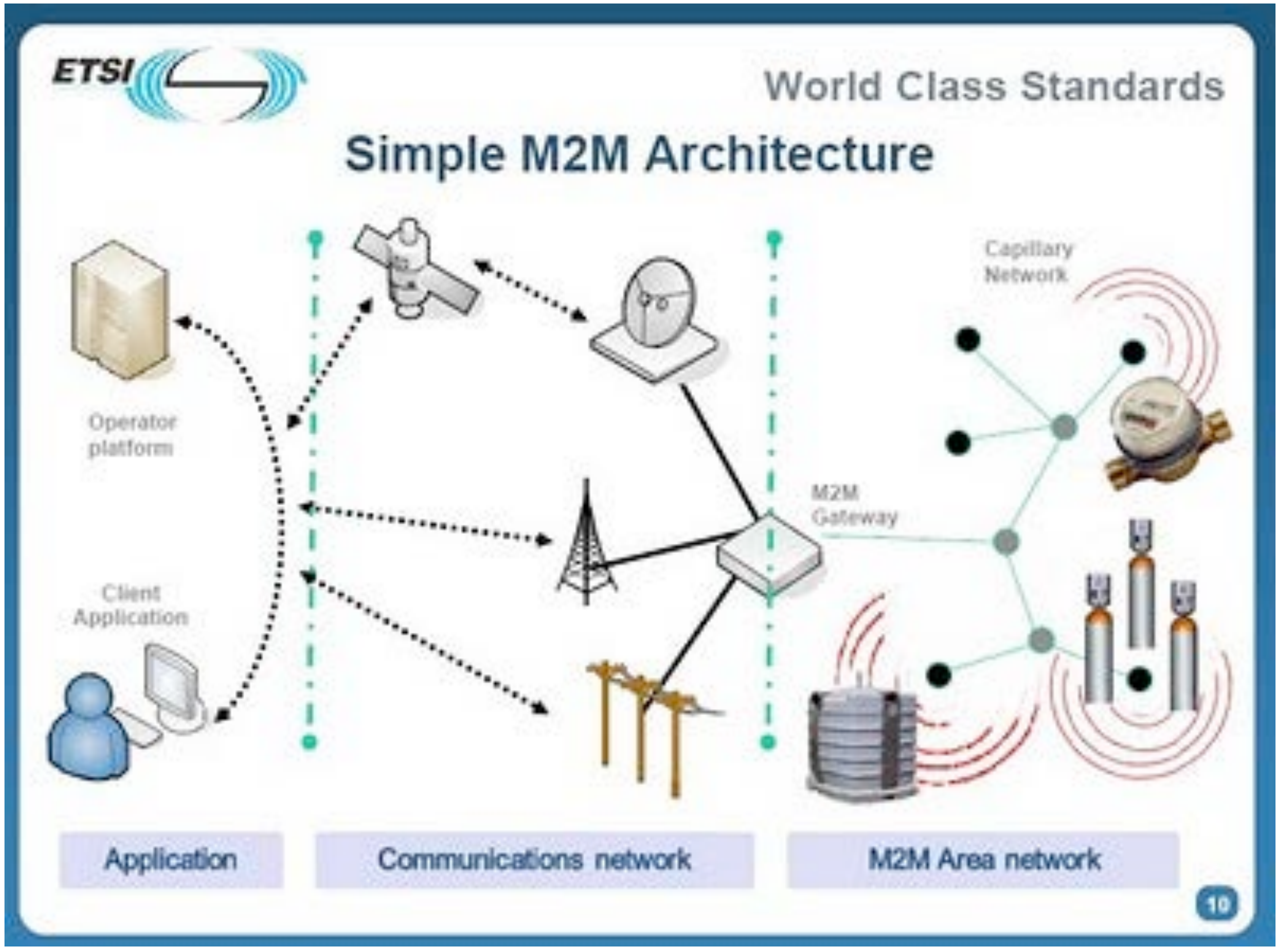
  "*The security of a software-intensive system is directly related to the quality of its software*".

- Over 90% of software security incidents are caused by attackers exploiting known software defects.
- An analysis of 45 e-business applications showed that 70% of security defects were design defects.

Extensibility

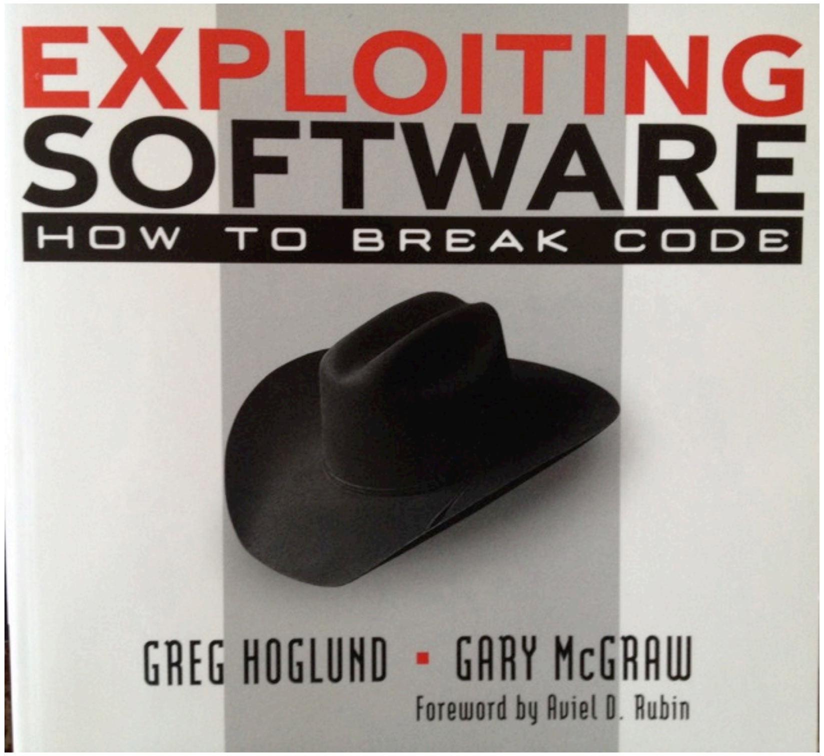


Connectivity



How to avoid trouble?

- Greg Hoglund and Gary McGraw, Exploiting Software – How to break code, Addison-Wesley, 2008.



In order to avoid trouble...

- We have to use engineering principles!
  - Learn from the past
  - Avoid making faults twice
  - Verify your results
    - Test your software carefully
    - Test the whole system carefully
    - Add tests that check for security issues

However...

- There is always the (infinite) game Attackers vs. Defenders!!!!

Some notations

- Vulnerability: „A problem that can be exploited by an attacker“
  - Bugs (implementation specific)
  - Flaws (design,...)

- A vulnerability may never be exploited!
- Exploitations may depend on the whole system‘s architecture (e.g. use of firewalls,..)

Some notations (cont.)

- Risk: „Captures the probability that a vulnerability will be exploited.“

- Attack pattern: „..is a blueprint for exploiting a software vulnerability.“
- Exploit: „Instance of an attack pattern.“
- Attack: „Act of carrying out an exploit.“

Attack patterns

- See for example: `http://capec.mitre.org/`

CAPEC™ **Common Attack Pattern Enumeration and Classification**
A Community Knowledge Resource for Building Secure Software

- Or: `https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack.html`

Homeland Security

Build Security In
Setting a higher standard for software assurance

Sponsored by DHS National Cyber Security Division

Security testing & planning

- Every attack (pattern) comprises
  - a sequence of actions

- When specifying actions, the initial state, and the goal state we come up with a planning problem!

Josip Bozic and **Franz Wotawa**

TESTING AS A PLANNING PROBLEM

Introduction

- Common vulnerabilities in web applications represent a major challenge in security testing [1].
- Several scanners and manual testing tools are available.
- Previously depiction of attacks as patterns [2, 3].
- Represent an attack as a sequence of actions.
- Testing as a planning problem [4].
- Automated testing approach for finding vulnerabilities without or at least with little user interactions.

Introduction

- Introducing a method that is based on planning for computing test cases where a test case is a sequence of interactions with the web application under test.
- Attack is nothing else than finding an interaction sequence that finally leads to a situation where we can exploit a vulnerability.
- Every request of an attacker can be seen as potential action having a precondition and an effect.
- Algorithm that makes use of a planner for generating test cases.

Security Testing via Planning

- Planners generate action sequences that lead a system from the initial state into a goal state.
- Plans instruct the system what to do in each step.
- Considering that the environment does not change during plan execution.
- In security testing: Plan for testing web applications against SQL injection (SQLI) and reflected and stored cross-site scripting (XSS).
- Test case generation problem as a planning problem.

Security Testing via Planning

Definition 1. *A tuple (I,G,A) is a planning problem, where I is the initial state, G is the goal state, and A is a set of actions, each of them comprising a pre condition and an effect (or post condition). For simplicity, we assume that each state is given as a set of (first order logic) predicates that are valid within this state. We also assume that the preconditions and the effects of an action* a_i *from A can be accessed via a function pre(*a_i*) and eff(*a_i*) respectively.*

Security Testing via Planning

- considers the currently available information of a web application (URL, parameters etc.).
- specifies what to expect from the SUT in case of a detected vulnerability.
- An action *a* from *A* can be executed in a state s_i if and only if its precondition *pre(a)* is fulfilled in s_i. Then new state s_{i+1} becomes active where *eff(a)* holds.

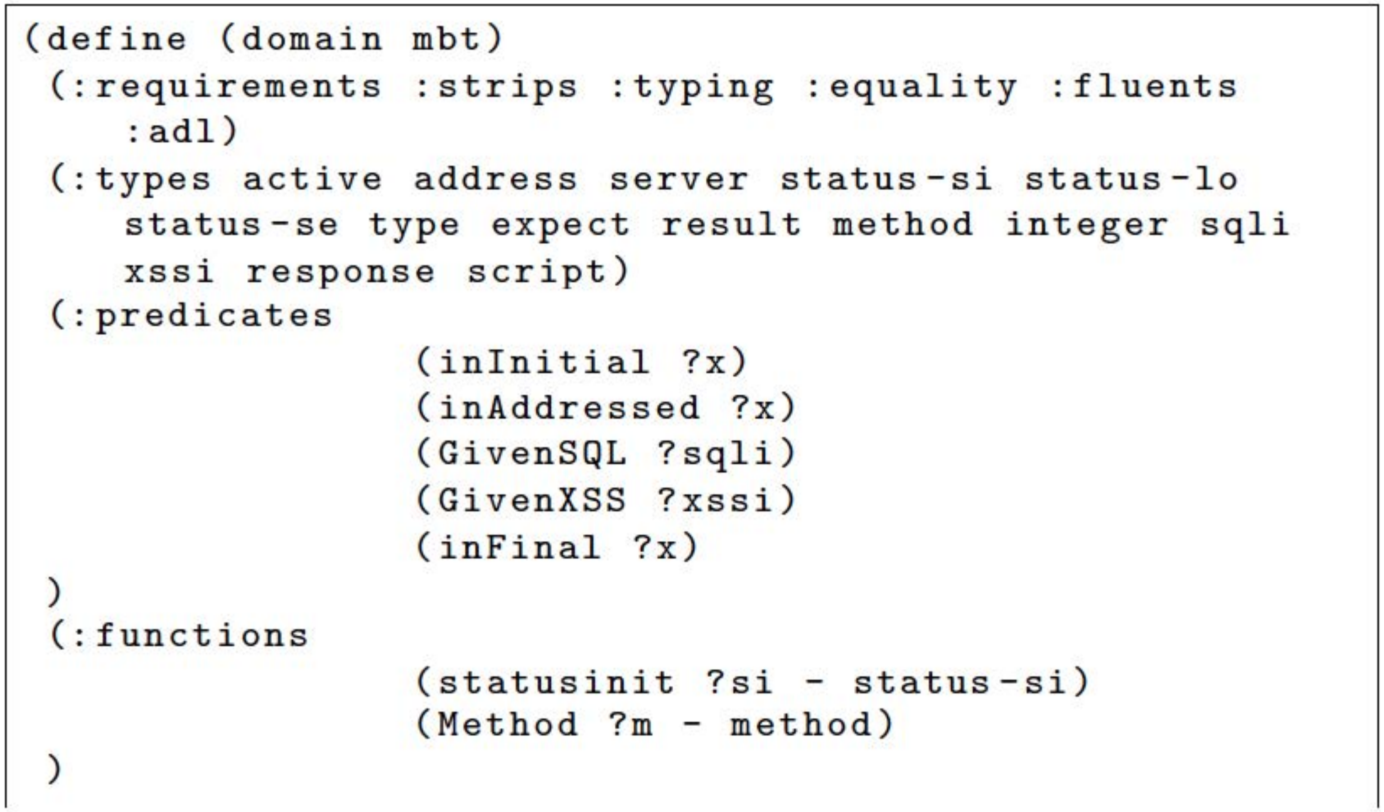
Definition 2. *A solution to the planning problem (I,G,A) is a sequence of actions* a_1 *to* a_n *in A such that:*

$$I\ a_1\ S_1\ a_2\ S_2 \ldots S_{n-1}\ a_n\ G$$

Security Testing via Planning

Planning Domain Definition Language (PDDL):

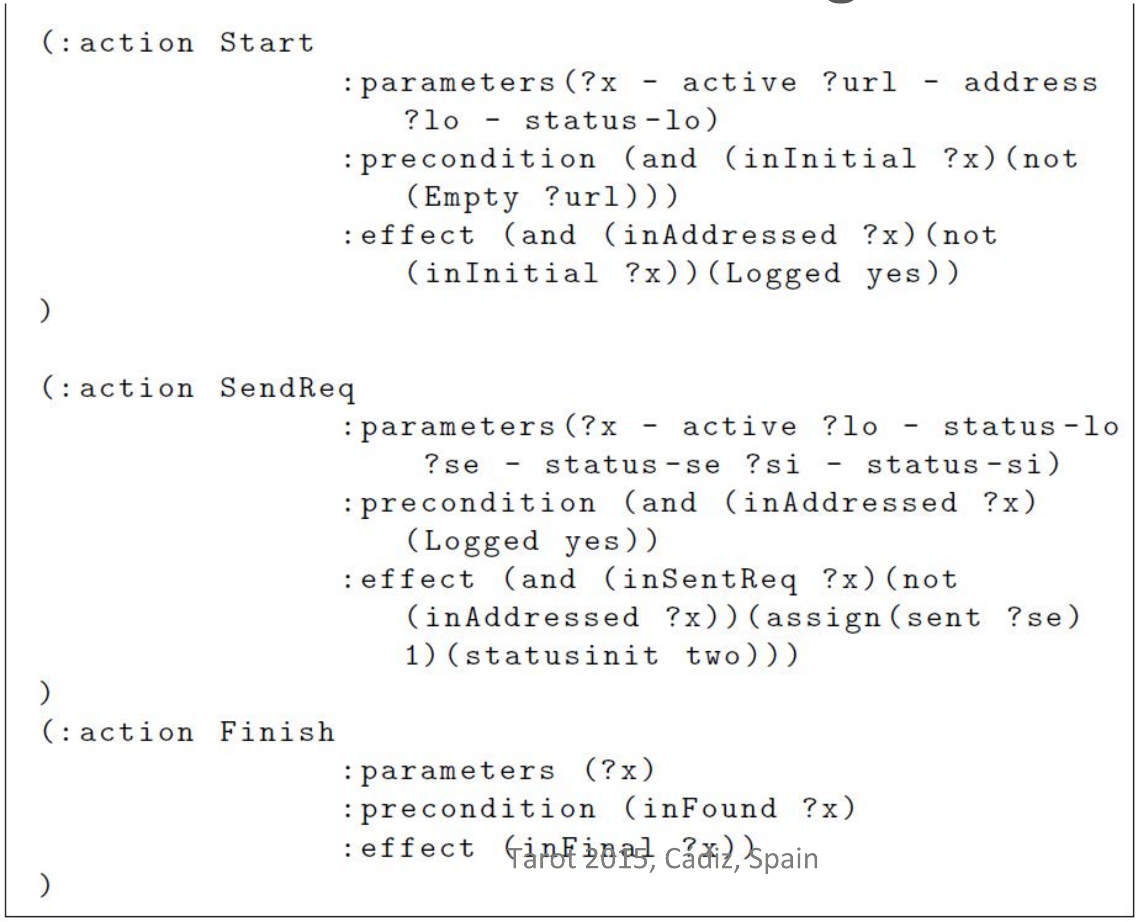
- Domain file: actions that are problem independent.



```
(define (domain mbt)
 (:requirements :strips :typing :equality :fluents
     :adl)
 (:types active address server status-si status-lo
     status-se type expect result method integer sqli
     xssi response script)
 (:predicates
                (inInitial ?x)
                (inAddressed ?x)
                (GivenSQL ?sqli)
                (GivenXSS ?xssi)
                (inFinal ?x)
 )
 (:functions
                (statusinit ?si - status-si)
                (Method ?m - method)
 )
```

Security Testing via Planning

- Domain file – actions: list of parameters and preconditions with the resulting effects.



```
(:action Start
               :parameters(?x - active ?url - address
                  ?lo - status-lo)
               :precondition (and (inInitial ?x)(not
                  (Empty ?url)))
               :effect (and (inAddressed ?x)(not
                  (inInitial ?x))(Logged yes))
)

(:action SendReq
               :parameters(?x - active ?lo - status-lo
                   ?se - status-se ?si - status-si)
               :precondition (and (inAddressed ?x)
                  (Logged yes))
               :effect (and (inSentReq ?x)(not
                  (inAddressed ?x))(assign(sent ?se)
                  1)(statusinit two)))
)
(:action Finish
               :parameters (?x)
               :precondition (inFound ?x)
               :effect (inFinal ?x))
)
```

Security Testing via Planning

- Problem file: application-specific values.

```
(define (problem mbt-problem)
 (:domain mbt)
 (:objects
                x - active
                type - type
                url - address
                m - method
 )
 (:init
                (inInitial x)
                (Logged no)
                (not (statusinit two))
                (Type sqli)
                (= (sent se) 0)
                (Method post)
 )
 (:goal (inFinal x)))
```

Security Testing via Planning

- In case the initial values satisfy a specific precondition from some action, this action is put on top of the planner.
- During execution: action effects might change values, which leads to the satisfaction of preconditions from some other action.
- The action generation continues as long the specified goal is not reached, thus generating a new plan.
- Otherwise: problem is considered improvable.
- Generated plan: abstract test case.

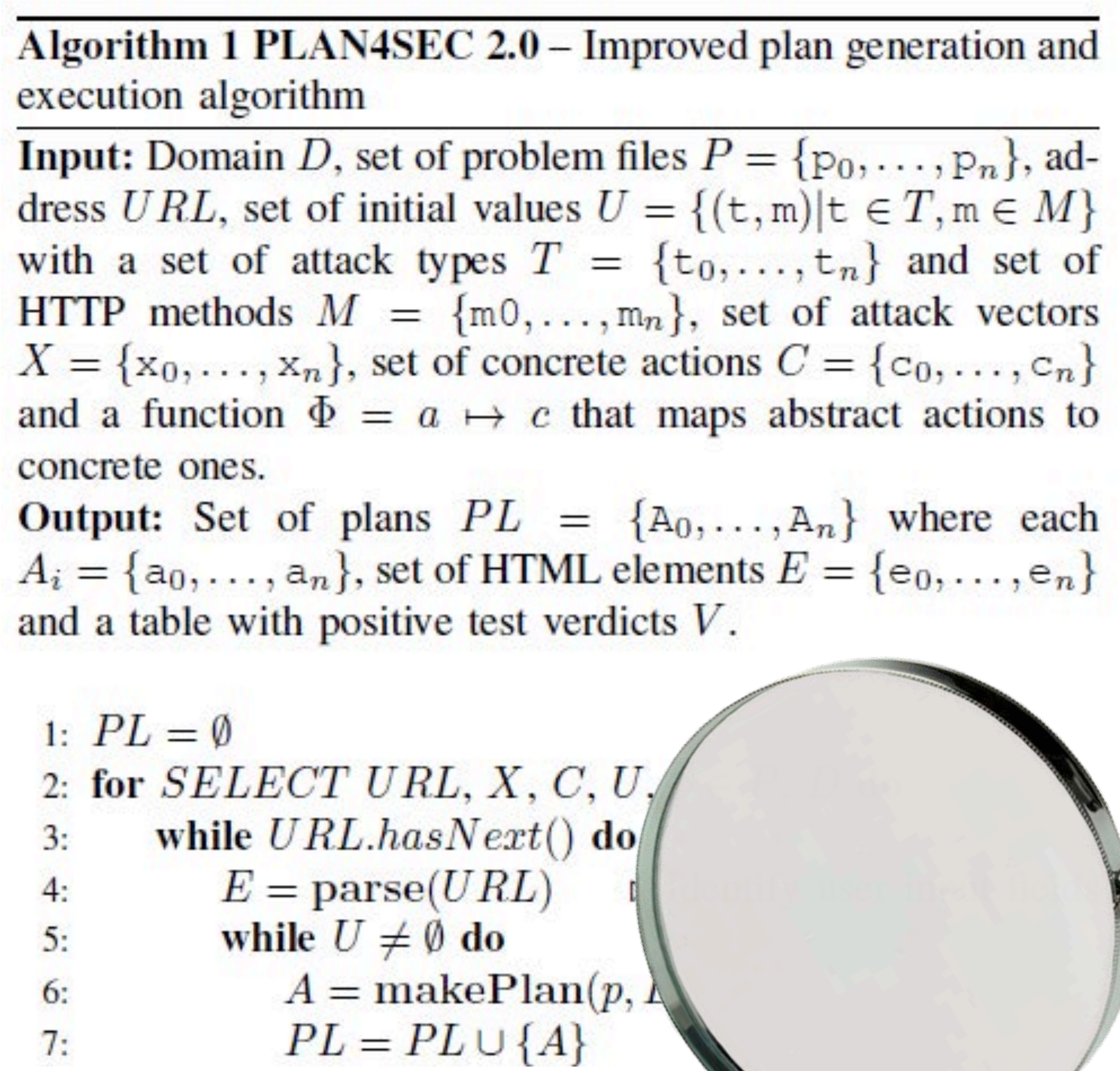
Security Testing via Planning

- Concretization with parser from **JavaFF** [5]: each abstract action has its concrete counterpart as a Java method.
- Communication with SUT: **HttpClient** [6] for creating and reading of HTTP messages. Response parsing with **jsoup** [7].
- Automated detection mechanisms for SQLI and XSS.
- Bypass communication via browser and escape application specific constraints.
- **Crawler4j** [8]: An open source Web crawler for Java.

PLAN4SEC

- Implementation of an algorithm: PLAN4SEC.
- Relies on the planning system **Metric-FF** [9], which itself uses the **FF planner** [10].
- Other planners can be used as well.
- Output: set of plans and verdict about success.
- PLAN4SEC has to terminate because all input sets are finite, determining the number of iterations.

PLAN4SEC



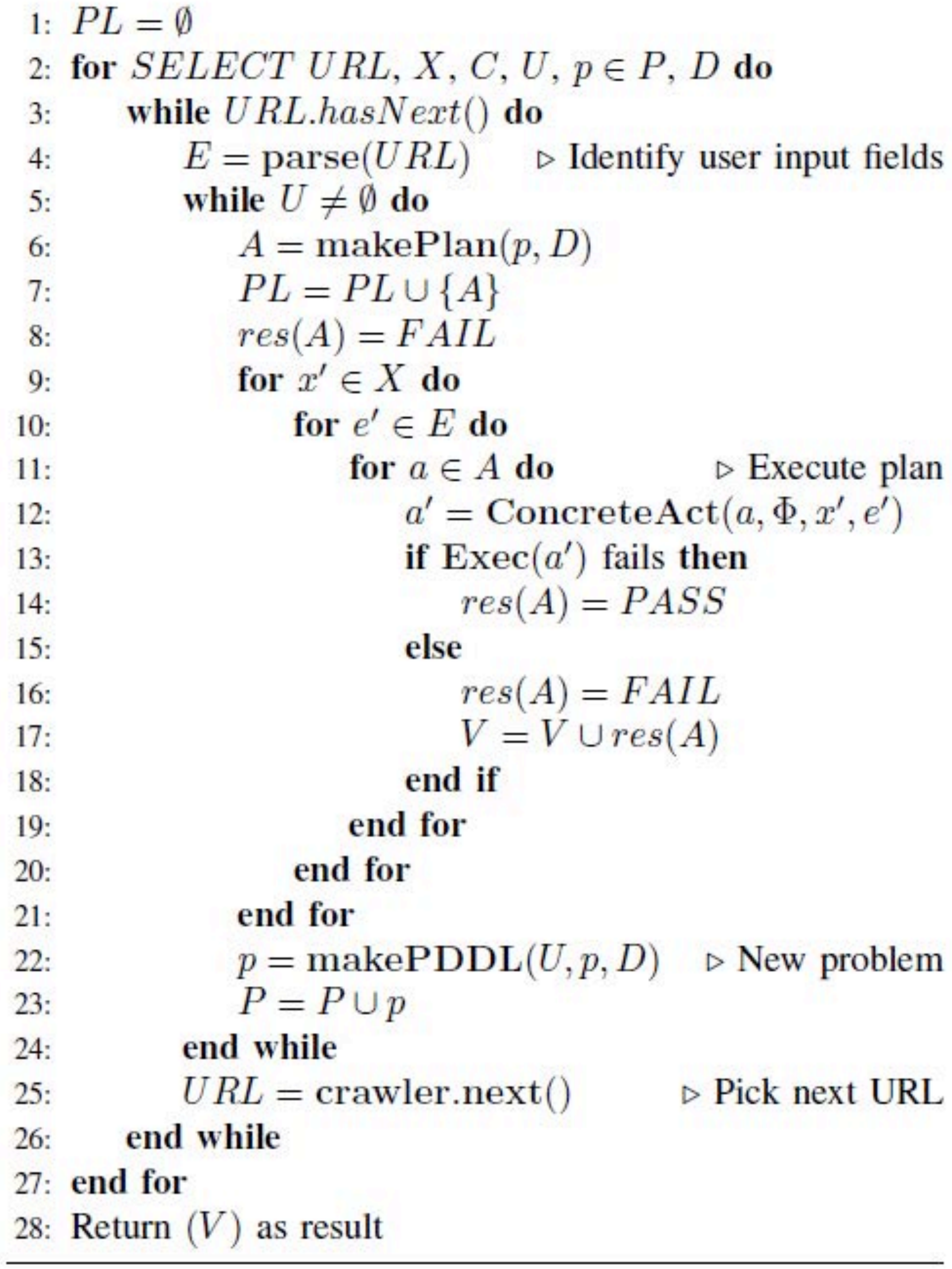
Algorithm 1 PLAN4SEC 2.0 – Improved plan generation and execution algorithm

Input: Domain D, set of problem files $P = \{\mathtt{p}_0, \ldots, \mathtt{p}_n\}$, address URL, set of initial values $U = \{(\mathtt{t}, \mathtt{m}) | \mathtt{t} \in T, \mathtt{m} \in M\}$ with a set of attack types $T = \{\mathtt{t}_0, \ldots, \mathtt{t}_n\}$ and set of HTTP methods $M = \{\mathtt{m0}, \ldots, \mathtt{m}_n\}$, set of attack vectors $X = \{\mathtt{x}_0, \ldots, \mathtt{x}_n\}$, set of concrete actions $C = \{\mathtt{c}_0, \ldots, \mathtt{c}_n\}$ and a function $\Phi = a \mapsto c$ that maps abstract actions to concrete ones.

Output: Set of plans $PL = \{\mathtt{A}_0, \ldots, \mathtt{A}_n\}$ where each $A_i = \{\mathtt{a}_0, \ldots, \mathtt{a}_n\}$, set of HTML elements $E = \{\mathtt{e}_0, \ldots, \mathtt{e}_n\}$ and a table with positive test verdicts V.

```
1: $PL = \emptyset$
2: for $SELECT\ URL, X, C, U,$
3:     while $URL.hasNext()$ do
4:         $E = \text{parse}(URL)$
5:         while $U \neq \emptyset$ do
6:             $A = \text{makePlan}(p,$
7:             $PL = PL \cup \{A\}$
```

PLAN4SEC



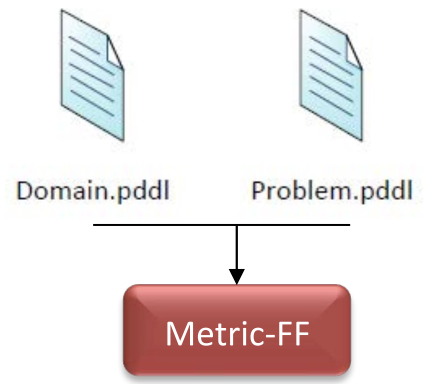
```
$PL = \emptyset$
for $SELECT\ URL, X, C, U, p \in P, D$ do
    while $URL.hasNext()$ do
        $E = \text{parse}(URL)$                ▷ Identify user input fields
        while $U \neq \emptyset$ do
            $A = \text{makePlan}(p, D)$
            $PL = PL \cup \{A\}$
            $res(A) = FAIL$
            for $x' \in X$ do
                for $e' \in E$ do
                    for $a \in A$ do                 ▷ Execute plan
                        $a' = \text{ConcreteAct}(a, \Phi, x', e')$
                        if $\text{Exec}(a')$ fails then
                            $res(A) = PASS$
                        else
                            $res(A) = FAIL$
                            $V = V \cup res(A)$
                        end if
                    end for
                end for
            end for
            $p = \text{makePDDL}(U, p, D)$     ▷ New problem
            $P = P \cup p$
        end while
        $URL = \text{crawler.next}()$          ▷ Pick next URL
    end while
end for
Return $(V)$ as result
```

Running Example

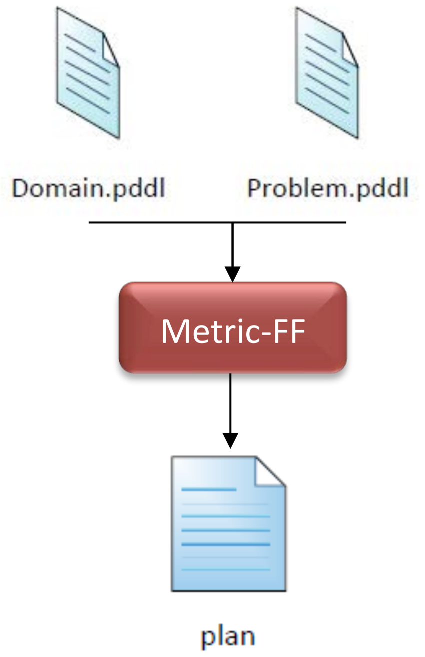
Domain.pddl

Problem.pddl

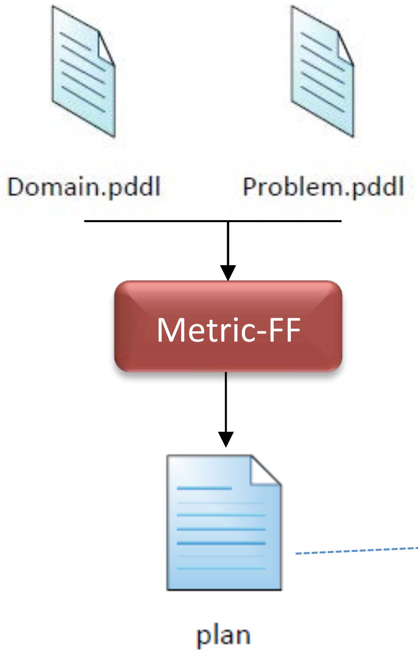
Running Example

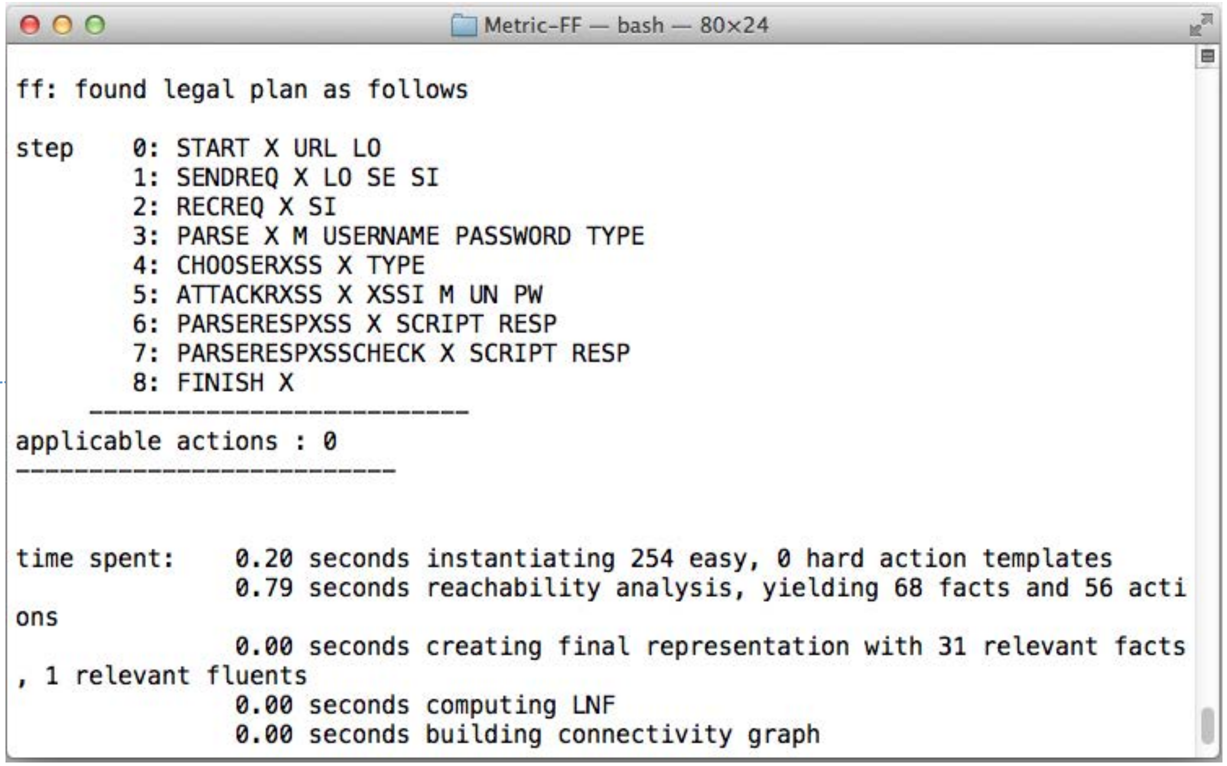


Running Example



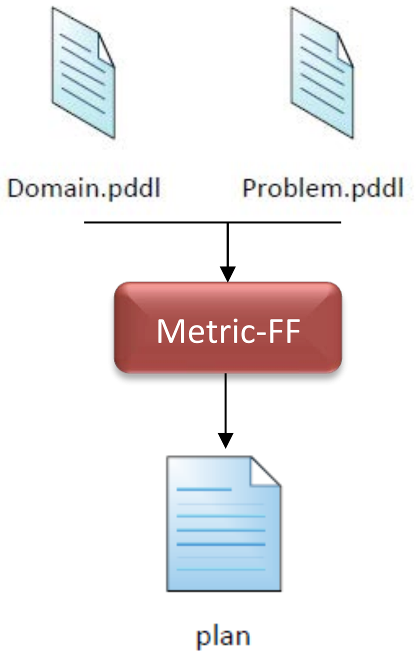
Running Example



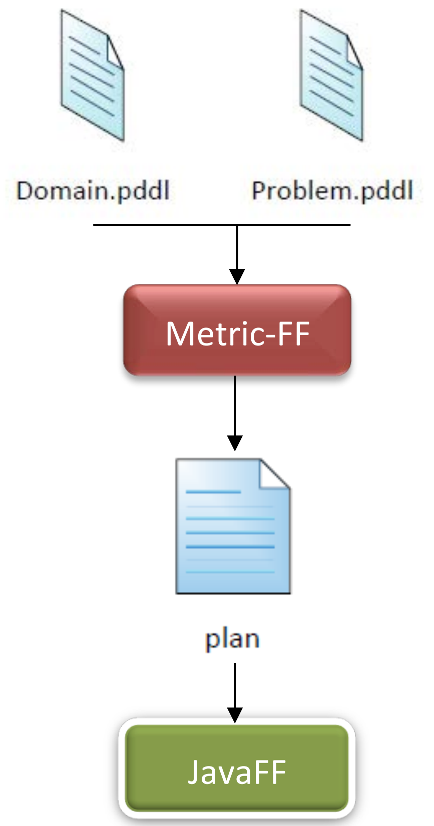


```
ff: found legal plan as follows

step    0: START X URL LO
        1: SENDREQ X LO SE SI
        2: RECREQ X SI
        3: PARSE X M USERNAME PASSWORD TYPE
        4: CHOOSERXSS X TYPE
        5: ATTACKRXSS X XSSI M UN PW
        6: PARSERESPXSS X SCRIPT RESP
        7: PARSERESPXSSCHECK X SCRIPT RESP
        8: FINISH X
     --------------------------
applicable actions : 0
--------------------------


time spent:    0.20 seconds instantiating 254 easy, 0 hard action templates
               0.79 seconds reachability analysis, yielding 68 facts and 56 acti
ons
               0.00 seconds creating final representation with 31 relevant facts
, 1 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
```

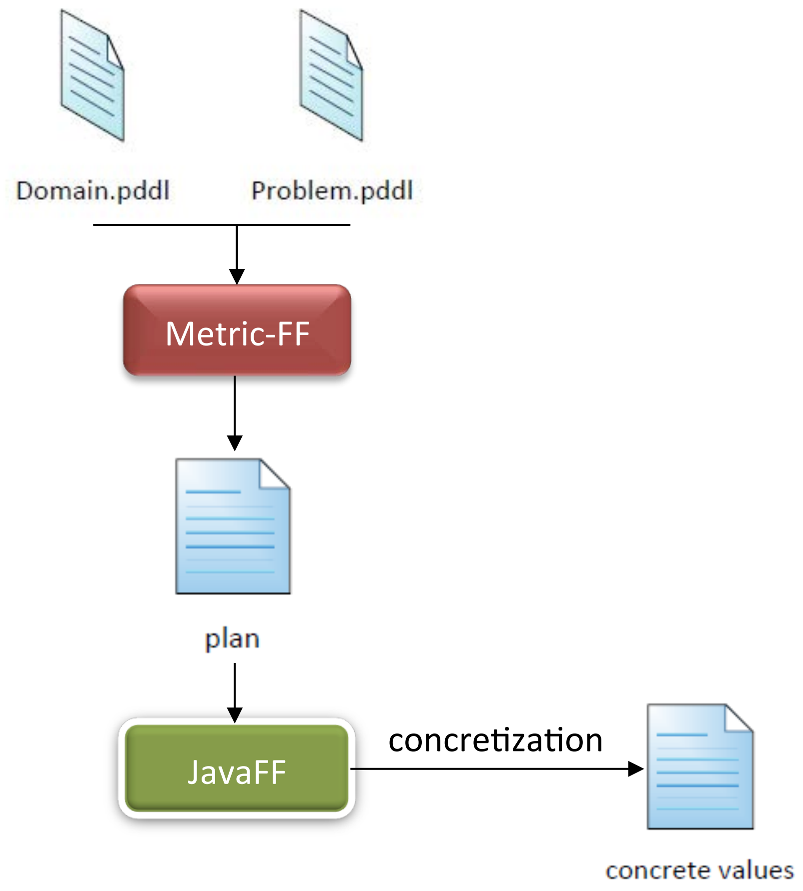
Running Example



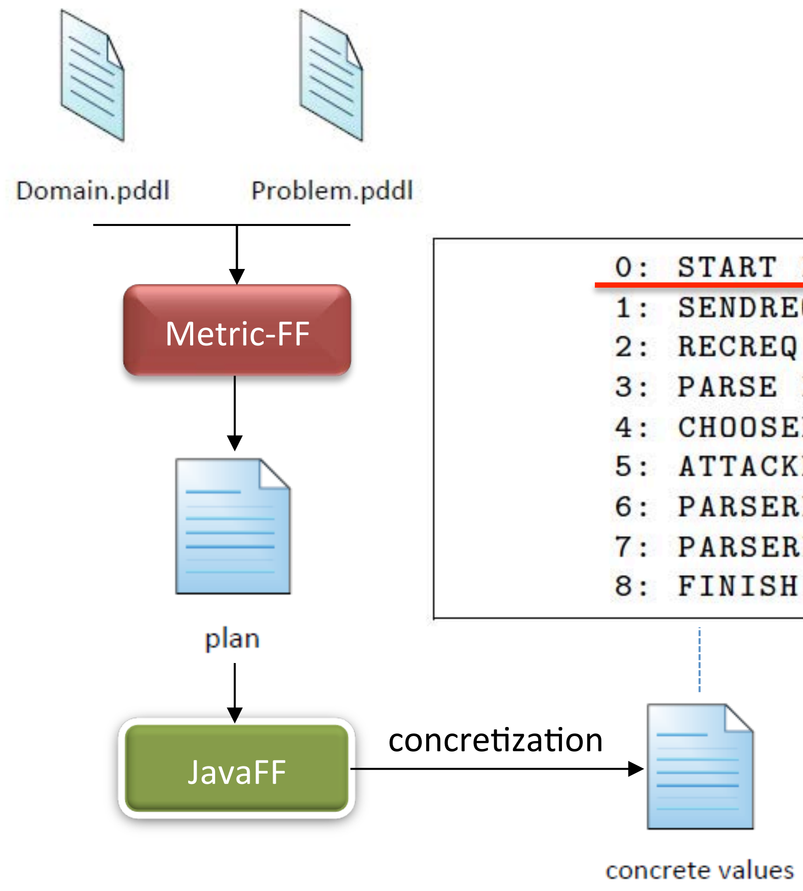
Running Example



Running Example

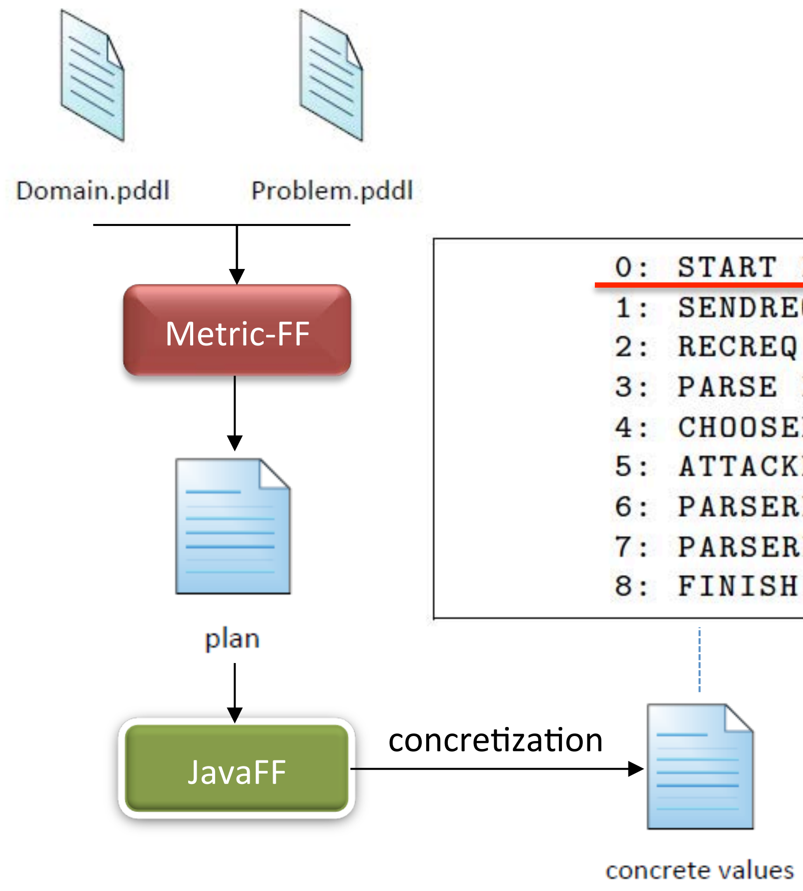


Running Example



```
0: START X URL LO
1: SENDREQ X LO SE SI
2: RECREQ X SI
3: PARSE X M USERNAME PASSWORD TYPE
4: CHOOSERXSS X TYPE
5: ATTACKRXSS X XSSI M UN PW
6: PARSERESPXSS X SCRIPT RESP
7: PARSERESPXSSCHECK X SCRIPT RESP
8: FINISH X
```

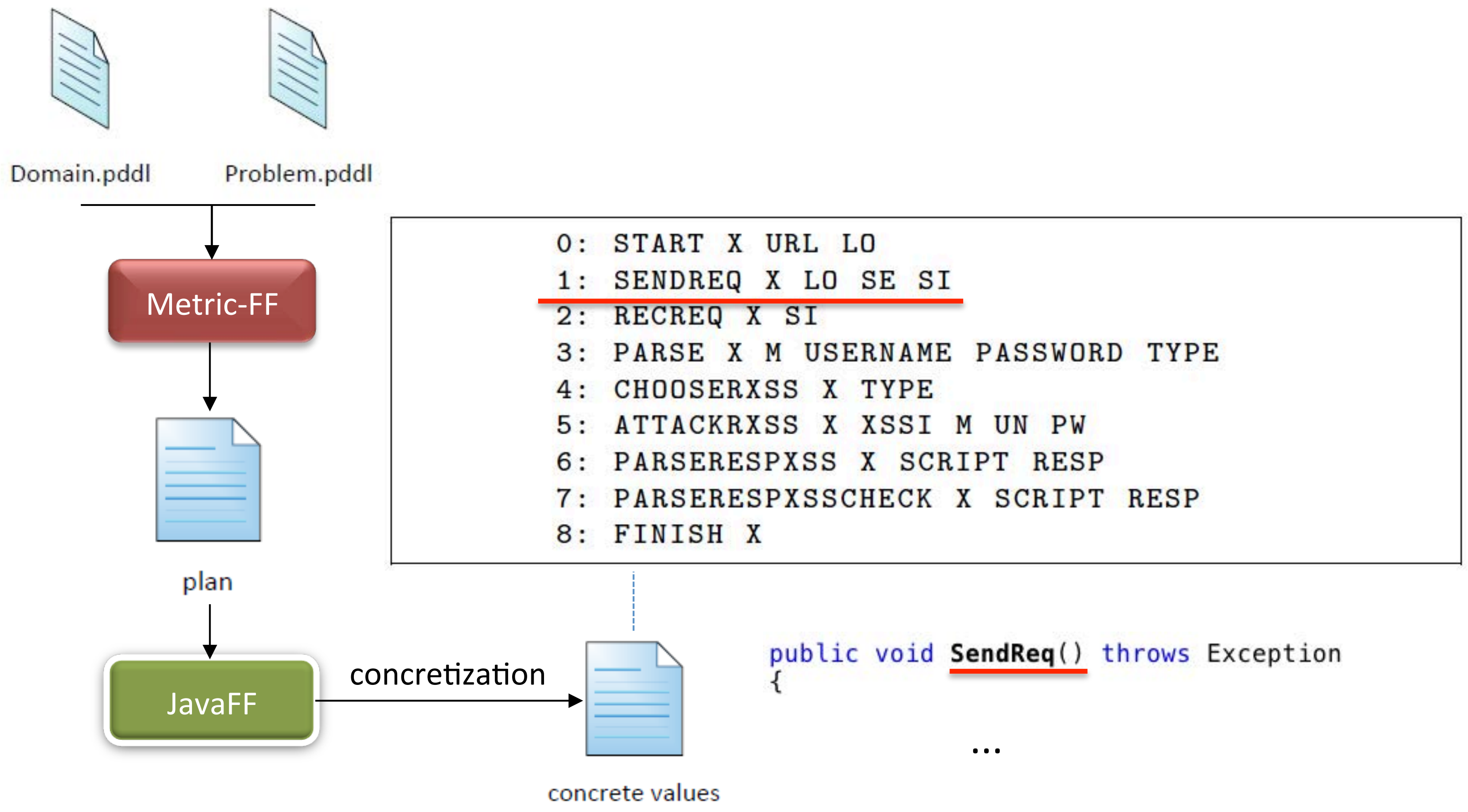
Running Example



```
0: START X URL LO
1: SENDREQ X LO SE SI
2: RECREQ X SI
3: PARSE X M USERNAME PASSWORD TYPE
4: CHOOSERXSS X TYPE
5: ATTACKRXSS X XSSI M UN PW
6: PARSERESPXSS X SCRIPT RESP
7: PARSERESPXSSCHECK X SCRIPT RESP
8: FINISH X
```

```java
public void start() throws Exception
{
    if (x.equals("inInitial") && !address.equals(""))
    {
        x = "inAddressed";
        lo = true;
    }
}
```

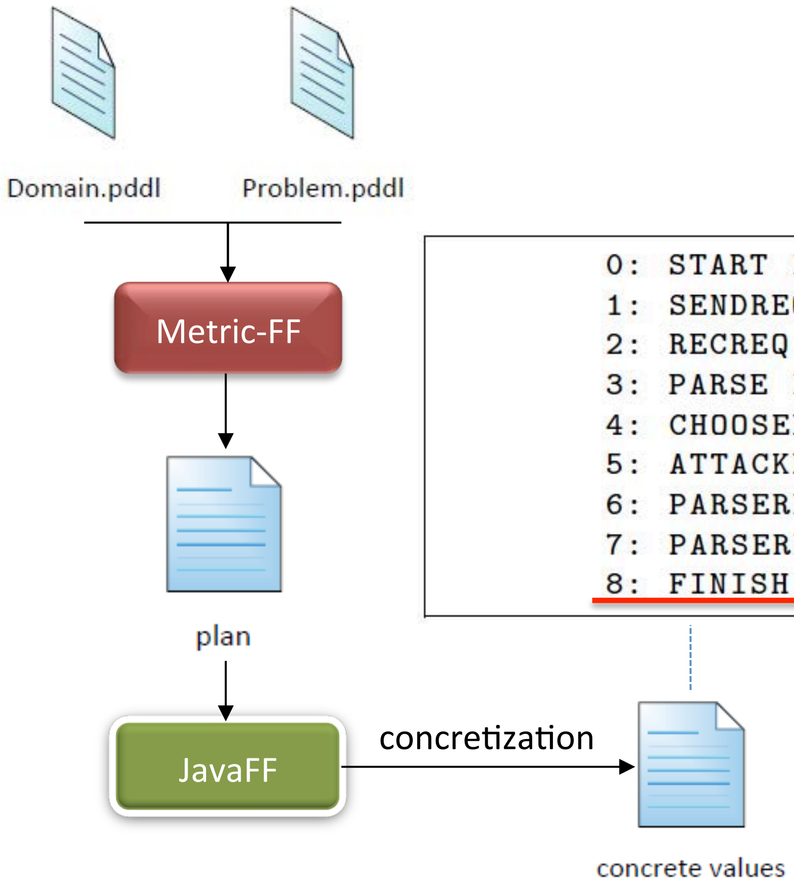
Running Example



```
0: START X URL LO
1: SENDREQ X LO SE SI
2: RECREQ X SI
3: PARSE X M USERNAME PASSWORD TYPE
4: CHOOSERXSS X TYPE
5: ATTACKRXSS X XSSI M UN PW
6: PARSERESPXSS X SCRIPT RESP
7: PARSERESPXSSCHECK X SCRIPT RESP
8: FINISH X
```

```
public void SendReq() throws Exception
{

        ...
```

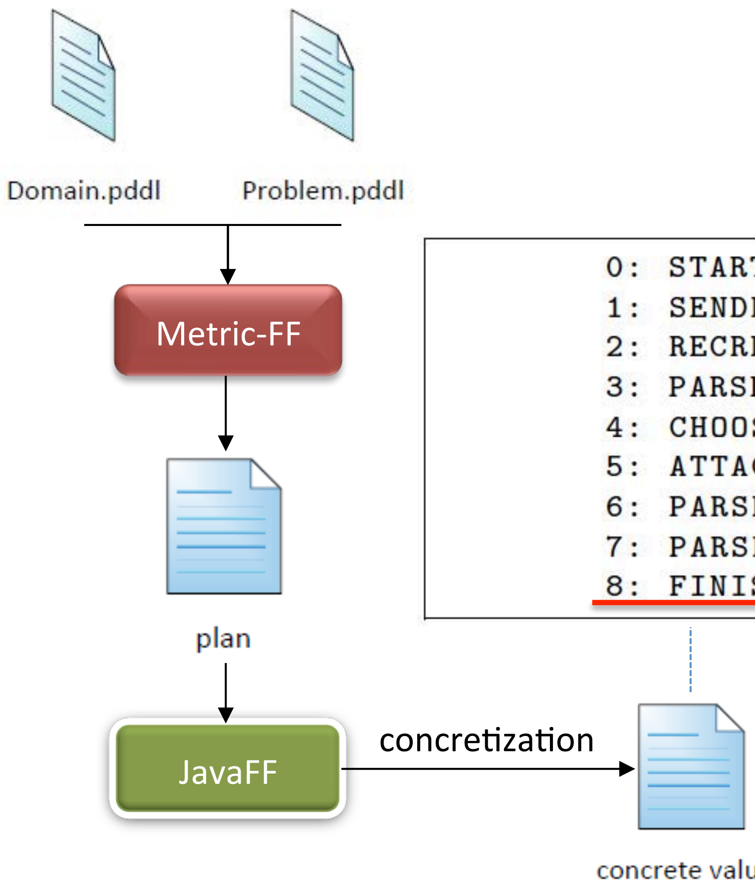
Running Example



```
0: START X URL LO
1: SENDREQ X LO SE SI
2: RECREQ X SI
3: PARSE X M USERNAME PASSWORD TYPE
4: CHOOSERXSS X TYPE
5: ATTACKRXSS X XSSI M UN PW
6: PARSERESPXSS X SCRIPT RESP
7: PARSERESPXSSCHECK X SCRIPT RESP
8: FINISH X
```

```
public void Finish() throws Exception
{
    if (x.equals("inFound"))
        x = "inFinal";
}
```

Running Example



```
0:  START X URL LO
1:  SENDREQ X LO SE SI
2:  RECREQ X SI
3:  PARSE X M USERNAME PASSWORD TYPE
4:  CHOOSERXSS X TYPE
5:  ATTACKRXSS X XSSI M UN PW
6:  PARSERESPXSS X SCRIPT RESP
7:  PARSERESPXSSCHECK X SCRIPT RESP
8:  FINISH X
```

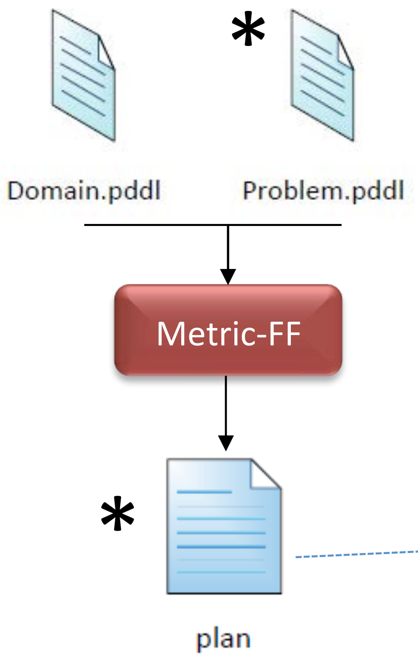
```
public void Finish() throws Exception
{
    if (x.equals("inFound"))
        x = "inFinal";
}
```

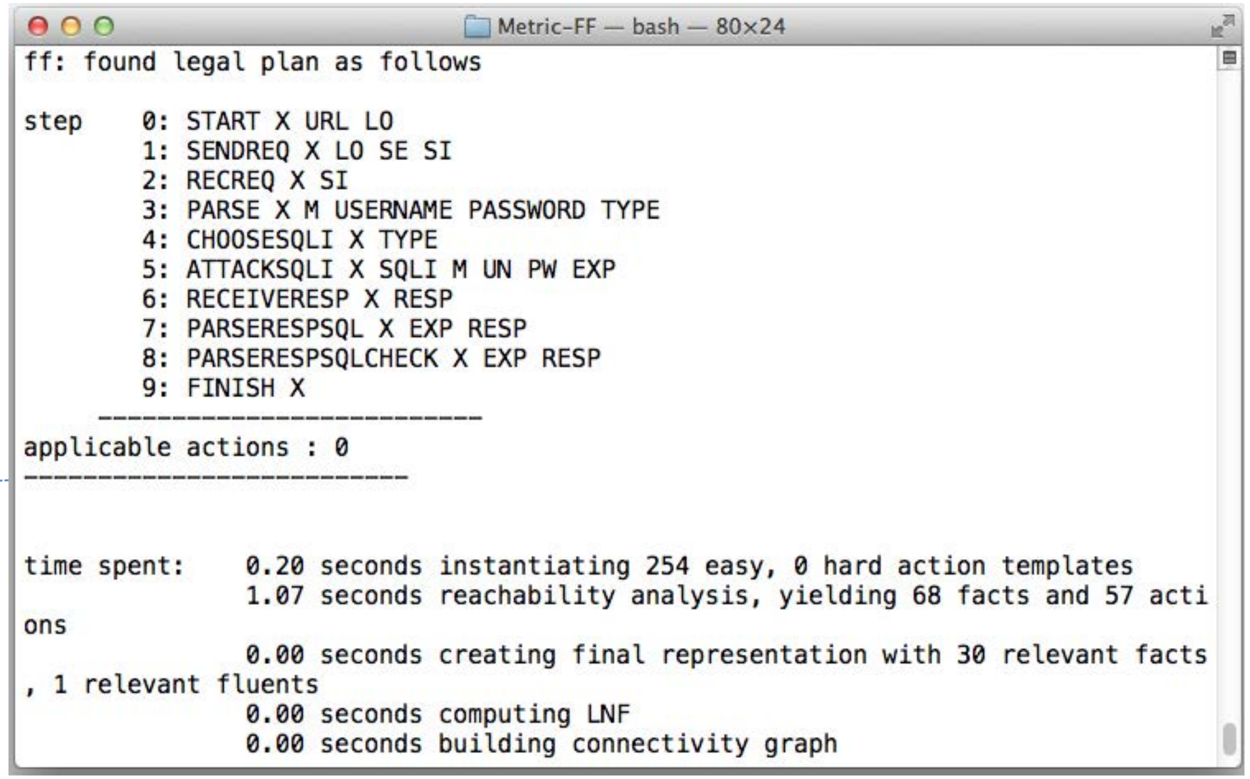
> complete!

Running Example

Domain.pddl *Problem.pddl

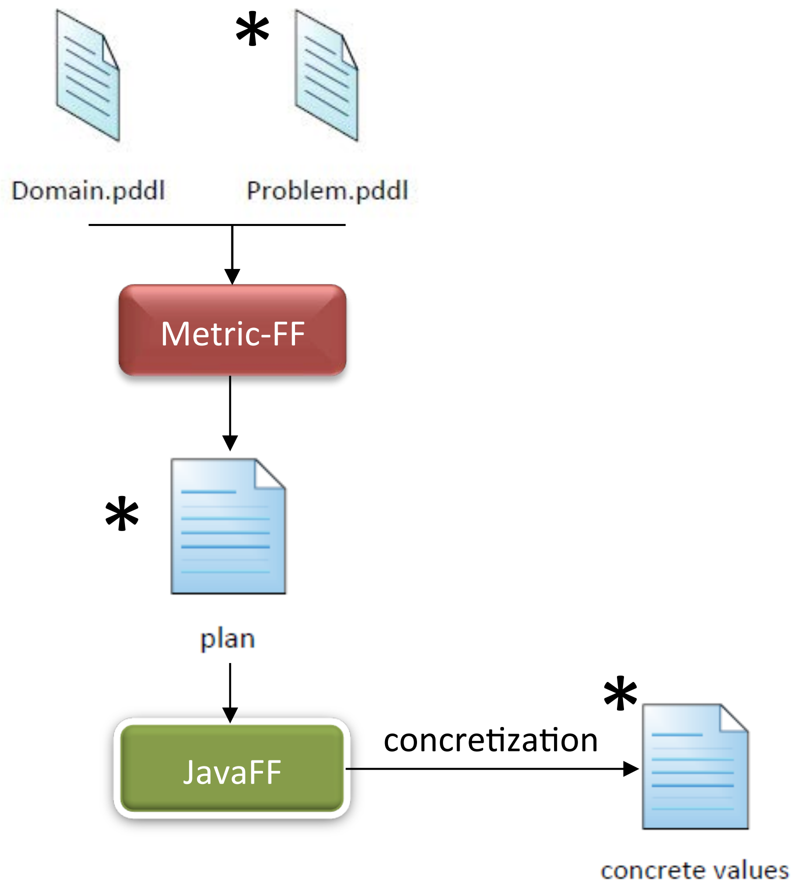
Running Example



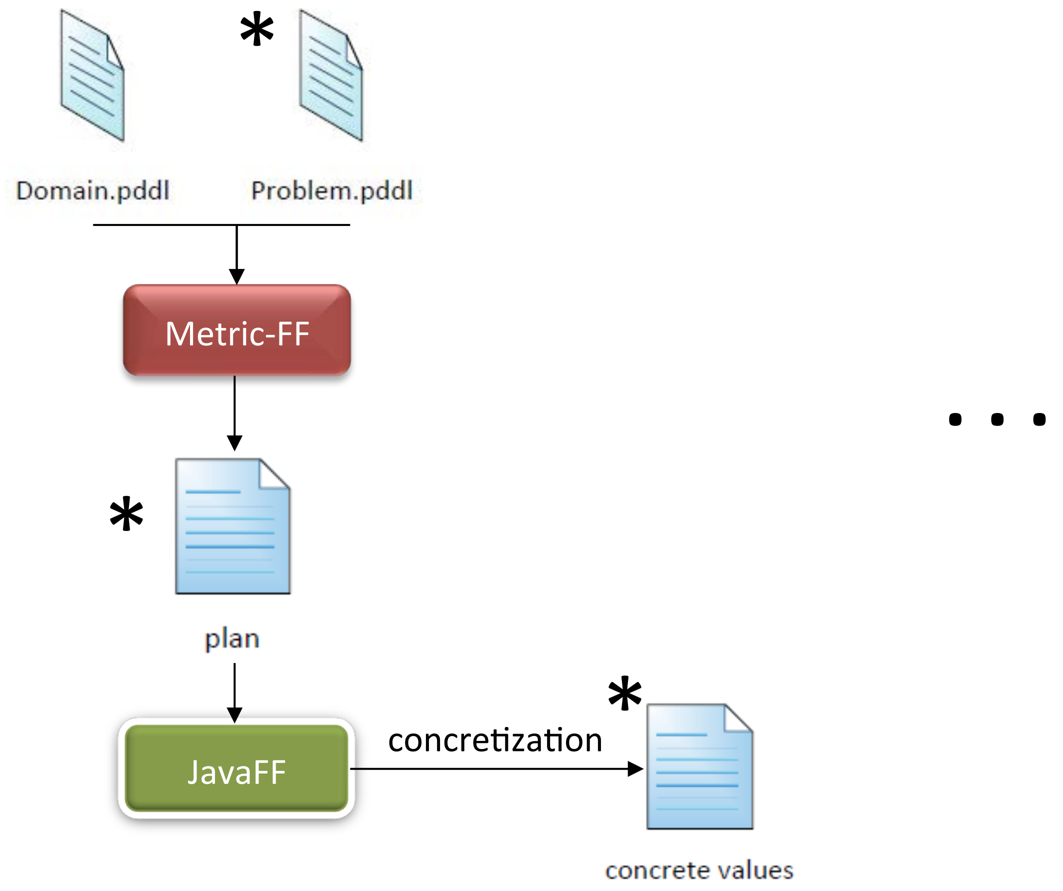


```
ff: found legal plan as follows

step    0: START X URL LO
        1: SENDREQ X LO SE SI
        2: RECREQ X SI
        3: PARSE X M USERNAME PASSWORD TYPE
        4: CHOOSESQLI X TYPE
        5: ATTACKSQLI X SQLI M UN PW EXP
        6: RECEIVERESP X RESP
        7: PARSERESPSQL X EXP RESP
        8: PARSERESPSQLCHECK X EXP RESP
        9: FINISH X
     --------------------------
applicable actions : 0
--------------------------



time spent:    0.20 seconds instantiating 254 easy, 0 hard action templates
               1.07 seconds reachability analysis, yielding 68 facts and 57 acti
ons
               0.00 seconds creating final representation with 30 relevant facts
, 1 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
```

Running Example



Running Example

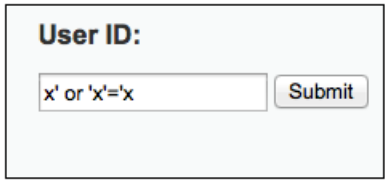


Vulnerability Detection - SQLI

- Expected value needed.
- Contains information about a database entry.
- If the information is returned when parsing the response body of the applicaton, it can be affirmed that the test case was successful.

Vulnerability Detection - SQLI

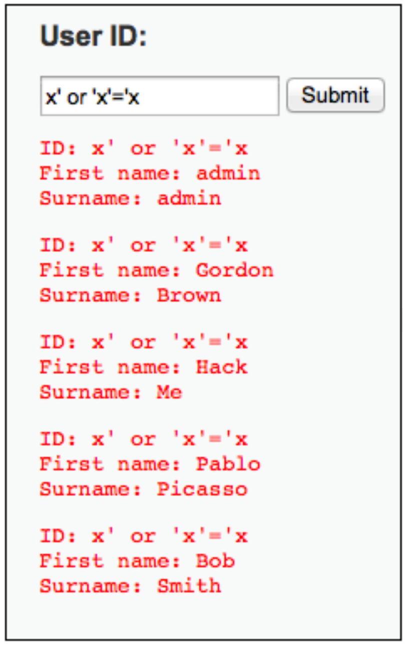
`input: x' OR 'x'='x ; expected: Brown`


User ID:

x' or 'x'='x

Vulnerability Detection - SQLI

input: x' OR 'x'='x ; expected: Brown



User ID:

x' or 'x'='x

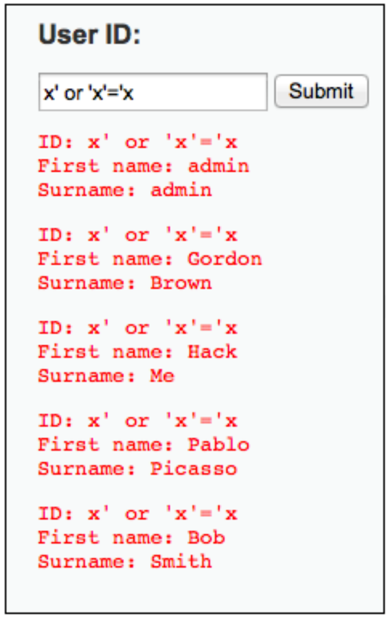
Submit

```
ID: x' or 'x'='x
First name: admin
Surname: admin

ID: x' or 'x'='x
First name: Gordon
Surname: Brown

ID: x' or 'x'='x
First name: Hack
Surname: Me

ID: x' or 'x'='x
First name: Pablo
Surname: Picasso

ID: x' or 'x'='x
First name: Bob
Surname: Smith
```

```
        <pre>ID: x' or 'x'='x<br>First name: admin<br>Surname: admin</pre><pre>ID: x' or
'x'='x<br>First name: Gordon<br>Surname: Brown</pre><pre>ID: x' or 'x'='x<br>First name:
Hack<br>Surname: Me</pre><pre>ID: x' or 'x'='x<br>First name: Pablo<br>Surname:
Picasso</pre><pre>ID: x' or 'x'='x<br>First name: Bob<br>Surname: Smith</pre>
```

Vulnerability Detection - SQLI

input: x' OR 'x'='x ; expected: Brown



User ID:

```
x' or 'x'='x
```

Submit

```
ID: x' or 'x'='x
First name: admin
Surname: admin

ID: x' or 'x'='x
First name: Gordon
Surname: Brown

ID: x' or 'x'='x
First name: Hack
Surname: Me

ID: x' or 'x'='x
First name: Pablo
Surname: Picasso

ID: x' or 'x'='x
First name: Bob
Surname: Smith
```

Success!

```
        <pre>ID: x' or 'x'='x<br>First name: admin<br>Surname: admin</pre><pre>ID: x' or
'x'='x<br>First name: Gordon<br>Surname: Brown</pre><pre>ID: x' or 'x'='x<br>First name:
Hack<br>Surname: Me</pre><pre>ID: x' or 'x'='x<br>First name: Pablo<br>Surname:
Picasso</pre><pre>ID: x' or 'x'='x<br>First name: Bob<br>Surname: Smith</pre>
```

Vulnerability Detection - XSS

- XSS in web applications is found upon un-sanitized user inputs, i.e. if a submitted script is processed unfiltered by an application.
- Harmful content is sent to a client and executed, causing damage.
- Indicator for vulnerability is a returned HTML element in server's HTTP response.
- In reflected XSS, the vulnerability is triggered immediately.
- In stored XSS, the input is stored in the database, which may trigger whenever that data is loaded.

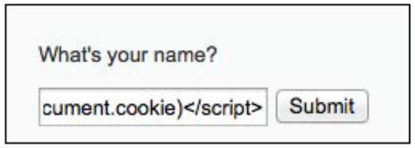
Vulnerability Detection - XSS

- XSS-critical HTML elements are *<script>*, *<src>*, *<img>*, *<iframe>* etc.
- First, the number of already existing critical HTML elements in the website is counted.
- Then, the input is submitted and the response is parsed in search for unfiltered HTML elements.

Vulnerability Detection - XSS

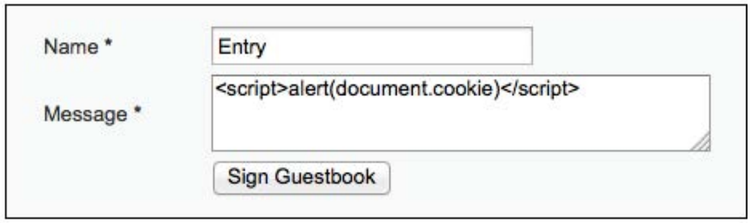
input: `<script>alert(document.cookie)</script>`

reflected


What's your name?

cument.cookie)</script>

stored

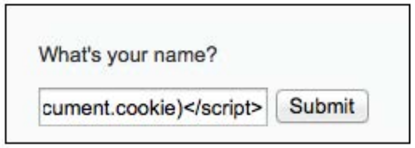


Name *	Entry
Message *	<script>alert(document.cookie)</script>

Vulnerability Detection - XSS

`input: <script>alert(document.cookie)</script>`

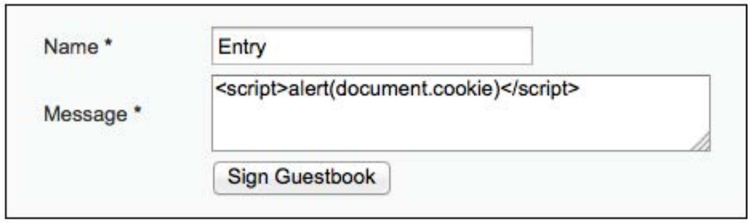
reflected



What's your name?

cument.cookie)</script> Submit

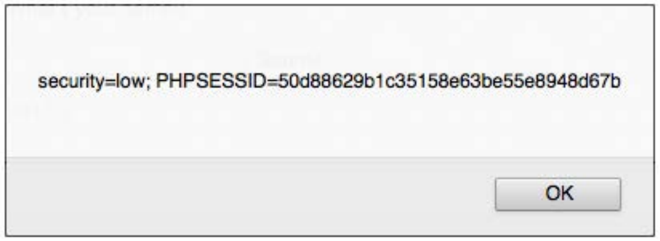
stored



Name * Entry

Message * <script>alert(document.cookie)</script>

Sign Guestbook



```
security=low; PHPSESSID=50d88629b1c35158e63be55e8948d67b
```

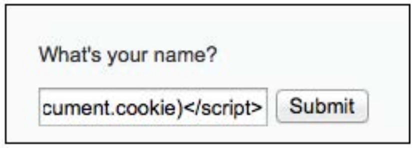
OK

```
<pre>Hello <script>alert(document.cookie)</script></pre>
```

Vulnerability Detection - XSS

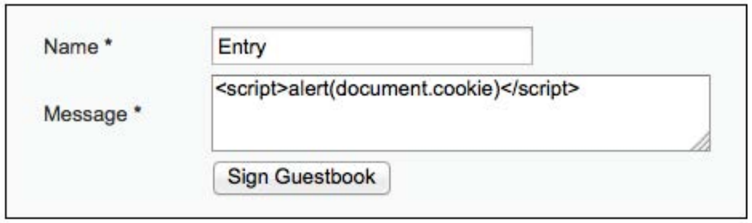
`input: <script>alert(document.cookie)</script>`

reflected


What's your name?

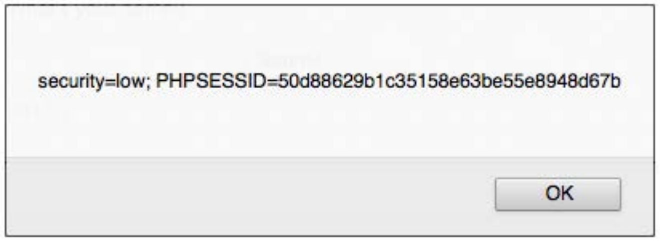
cument.cookie)</script> Submit

stored


Name * Entry

Message * <script>alert(document.cookie)</script>

Sign Guestbook


security=low; PHPSESSID=50d88629b1c35158e63be55e8948d67b

OK

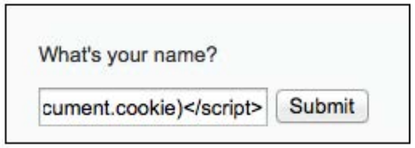
> Success!

```
<pre>Hello <script>alert(document.cookie)</script></pre>
```

Vulnerability Detection - XSS

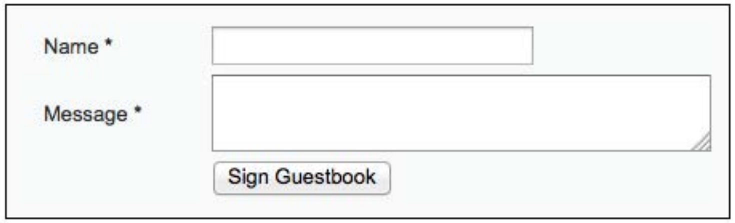
`input: <script>alert(document.cookie)</script>`

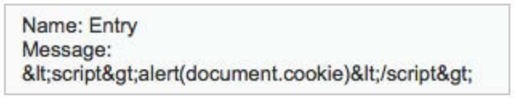
`reflected`


What's your name?

cument.cookie)</script>

`stored`

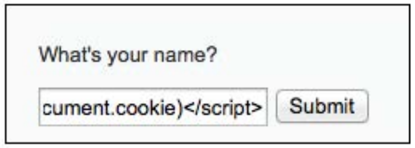



Name: Entry
Message:
<script>alert(document.cookie)</script>

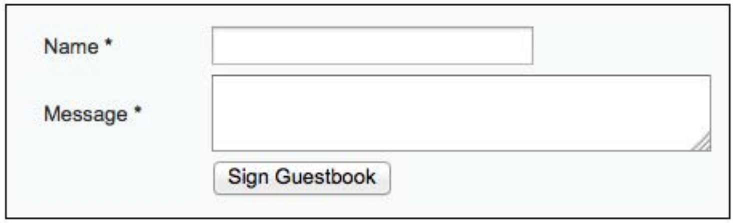
Vulnerability Detection - XSS

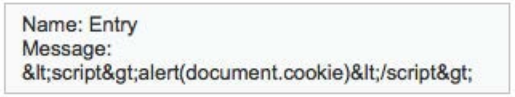
input: `<script>alert(document.cookie)</script>`

reflected



stored





Name: Entry
Message:
<script>alert(document.cookie)</script>

> Failure!

```
<pre>Hello &lt;script&gt;alert(document.cookie)&lt;/script&gt;</pre>
```

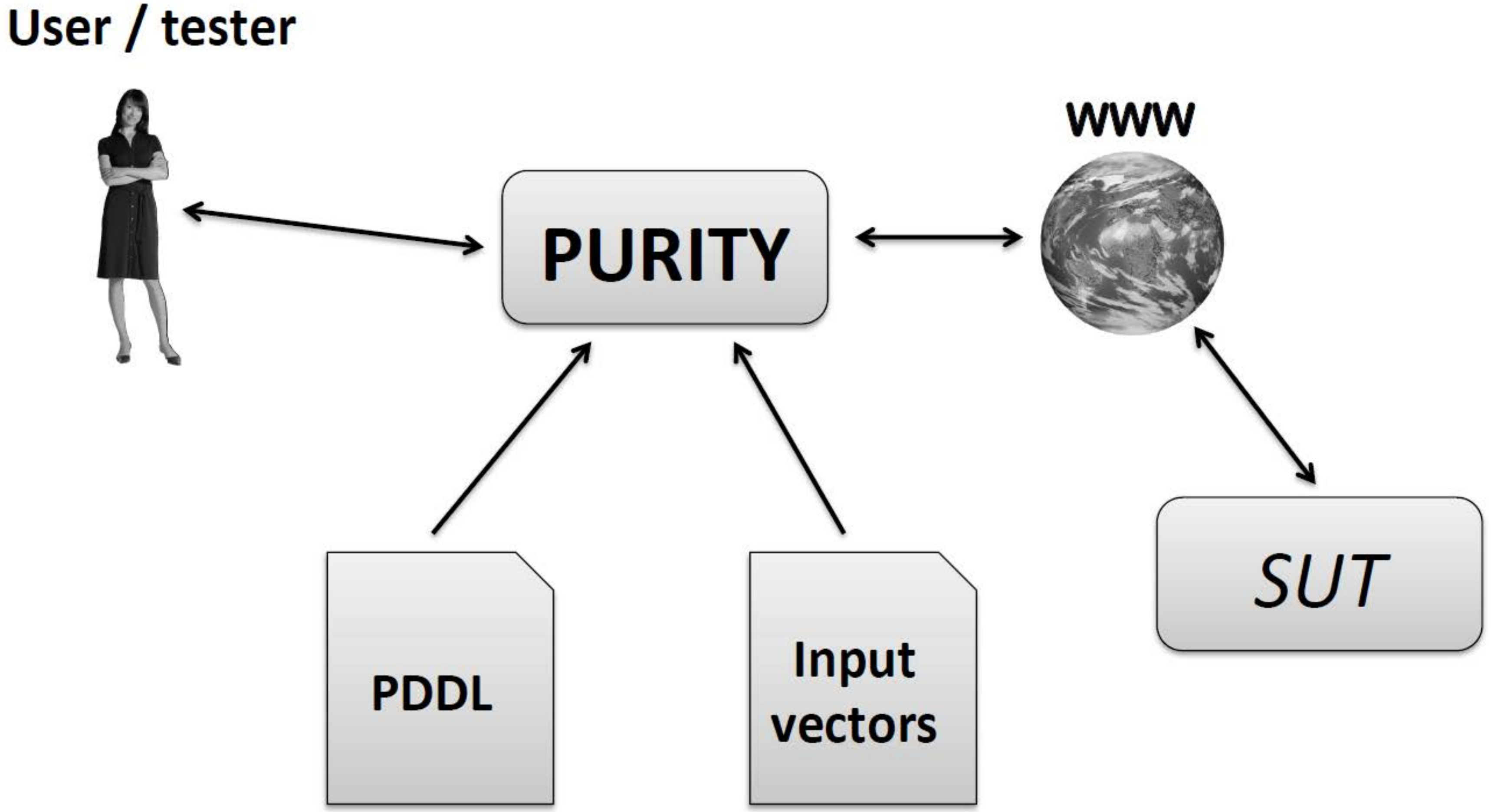
PURITY

- In order to make security testing of web applications easier, we propose the penetration testing tool *Planning-based secURITY testing tool*.
- Meant for testing for SQLI and XSS.
- Manually or automated testing (or something in-between).
- Easy to use but also provides high configurability and offers extendibility.
- Encompasses several elements that interact with each other as well as with the user.

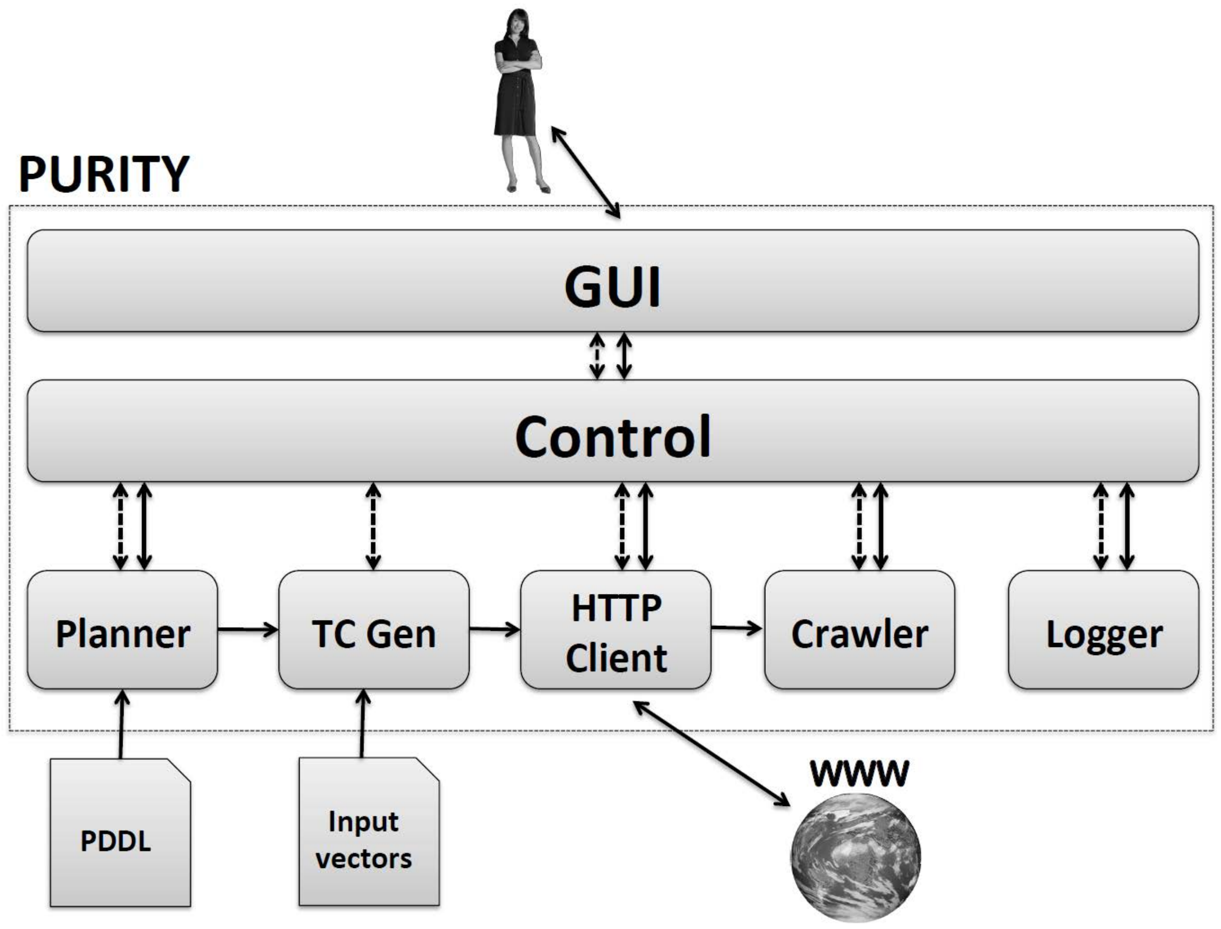
PURITY

- The tester can interact with the program on a minimum scale, i.e. by setting only the initial configuration like URL address.
- A test can be carried out completely manually by assigning specific values to selected parts of the website.

PURITY



Key components



Key components

- All interaction between tester and PURITY proceeds over the GUI.
- Implementation handles data flow between individual components and tester.
- The web application is accessed either over the World Wide Web or locally per URL.
- The communication between PURITY and SUT is handled dynamically over HTTP. Here the attack vectors are submitted and the response is parsed.

Key components

- The Web crawler browses the SUT and identifies hyperlinks in websites that are connected to the initial URL.
- Two initial test sets, one with SQL injections and the other containing XSS vectors. New ones can be obtained externally by attaching them to our tool.
- The planner generates abstract test cases from PDDL's.
- The test case generator reads the abstract actions and searches for their concrete counterpart in the implementation.

Key components

- For one abstract object from an action we can apply a dozen of concrete attack vectors.
- The logger collects all relevant data produced during the execution.

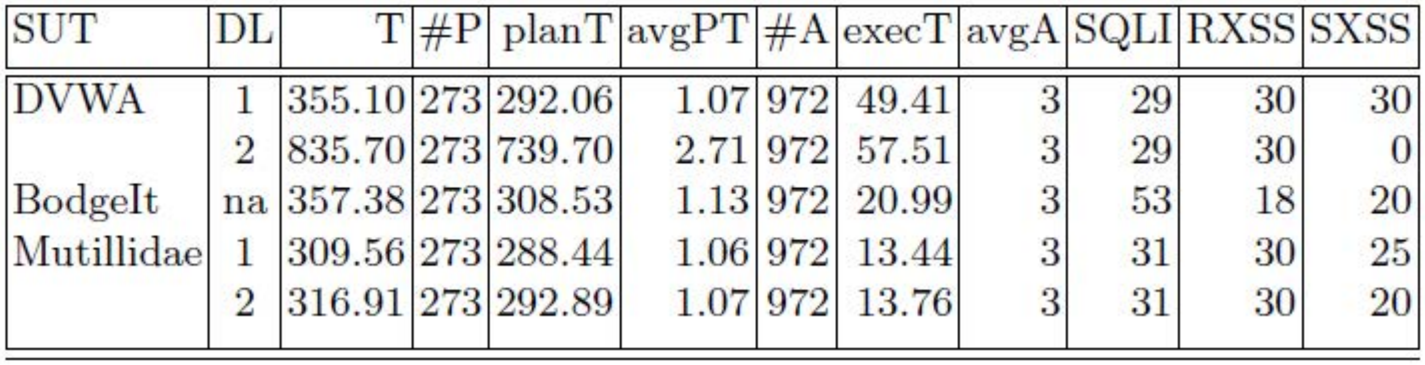
Modes of Use

- ***Completely Automatic***: Performs the execution in a completely automated manner. Of all modes, this one covers most of the functionality of PURITY.
- ***Partly Automatic***: Just one plan is generated. The tester can make experiments by deleting and adding actions or changing the order of their appearance. Only one execution is carried out per attack vector.
- ***Completely Manual***: Test a single website by manually writing values for all its user input fields.
- ***Partly Manual***: Test one specific element against a list of vectors in an automated manner.

Evaluation

- $2\times 3+1$ tested applications.
- 1/SQLI and 1/XSS inputs.
- Harmless input: detection of a potentially good starting point for attacks.
- 19 action definitions and several predicates and intial values.
- Changing values: active state, attack type, login information and HTTP methods.
- Higher security levels have more sophisticated input filtering mechanisms.

Evaluation



SUT	DL	T	#P	planT	avgPT	#A	execT	avgA	SQLI	RXSS	SXSS
DVWA	1	355.10	273	292.06	1.07	972	49.41	3	29	30	30
	2	835.70	273	739.70	2.71	972	57.51	3	29	30	0
BodgeIt	na	357.38	273	308.53	1.13	972	20.99	3	53	18	20
Mutillidae	1	309.56	273	288.44	1.06	972	13.44	3	31	30	25
	2	316.91	273	292.89	1.07	972	13.76	3	31	30	20

- Third security level remained impervious.
- Better filtering mechanisms for stored XSS.
- Acceptable execution time.
- Minor individual adaptation: URL address, expected value for SQLI.

Evaluation

- Unchanged domain specification: same number of generated plans for every SUT.
- Higher number of potential combinations of different initial parameters causes more plans but also more successful tests.
- Size of plan also depends on number of actions and preconditions.
- Planning takes more time than execution.

Conclusion and Future Work

- Novel approach to security testing based on planning.
- Formalization of the test case generation problem as a planning problem.
- Algorithm for security testing of web applications.
- Research prototype: PURITY
- The tester is also offered the possibility to execute the tool in a manual manner.

Conclusion and Future Work

- Further improvements: more data in PDDL, more user configurability.
- Use different test case generation techniques.
- The execution time is relatively low.

References

[1] "OWASP Top 10," https://www.owasp.org/index.php/Top_10.

[2] J. Bozic and F. Wotawa, "Security Testing Based on Attack Patterns," in Proceedings of the 5th International Workshop on Security Testing (SECTEST'14), 2014.

[3] J. Bozic, D. E. Simos and F. Wotawa, "Attack Pattern-Based Combinatorial Testing," in Proceedings of the 9th International Workshop on Automation of Software Test (AST'14), 2014.

[4] J. Bozic and F. Wotawa, "Plan it! Automated Security Testing Based on Planning," in Proceedings of the 26th IFIP WG 6.1 International Conference (ICTSS'14), September 2014, pp. 48–62.

[5] "Java FF," http://www.inf.kcl.ac.uk/staff/andrew/JavaFF/.

[6] "HttpClient," http://hc.apache.org/httpcomponents-client-ga/.

[7] "jsoup: Java HTML Parser," http://jsoup.org/.

[8] "Crawler4j," An open source Web crawler for Java.

[9] "METRIC-FF," http://fai.cs.uni-saarland.de/hoffmann/metric-ff.html.

[10] "Fast-Forward," http://fai.cs.uni-saarland.de/hoffmann/ff.html.

THANK YOU FOR YOUR ATTENTION!

QUESTIONS?