# Подходы к сигнатурному статическому анализу

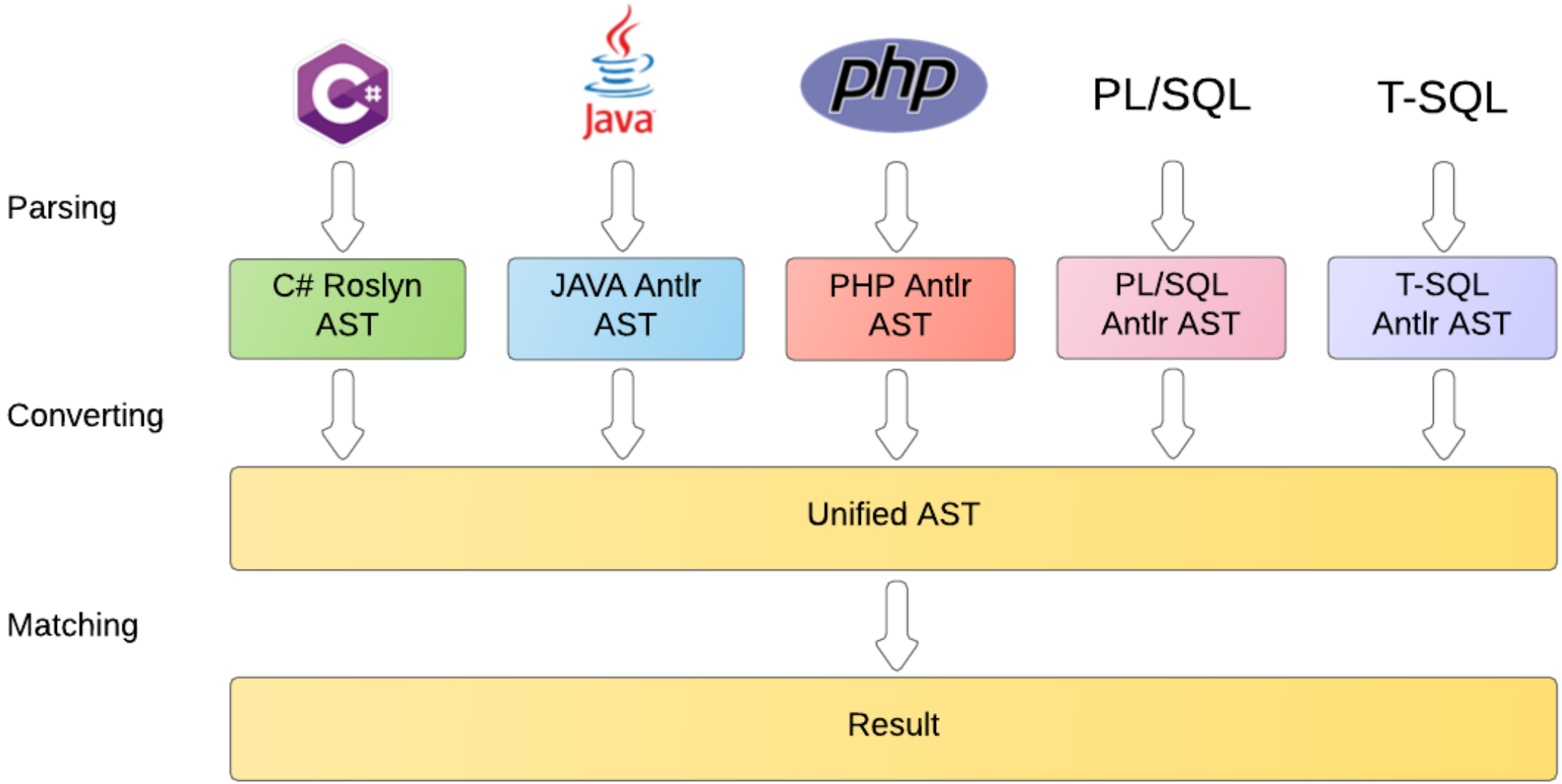
Иван Кочуркин

POSITIVE TECHNOLOGIES

ptsecurity.com

# Этапы работы анализатора Pattern Matching

POSITIVE TECHNOLOGIES



# Обоснование разработки анализатора Pattern Matching

POSITIVE TECHNOLOGIES

| Преимущества | Недостатки |
| --- | --- |
| • Простая разработка, поддержка и тестирование<br>• Быстрая масштабируемость на другие языки<br>• Обработка файлов с синтаксическими ошибками | • Невозможность анализировать семантику кода<br>• Покрытие частных случаев вместо общих<br>• Невозможность учитывать конфигурацию и правила бизнес-логики |

# Задачи парсинга

POSITIVE TECHNOLOGIES

- Разбор контекстно-зависимых конструкций
- Разрешение неоднозначностей
- Анализ комментариев
- Обработка ошибок
- Производительность и потребление памяти
- Использование Visitor или Listener для обхода деревьев

Теория и практика парсинга исходников с помощью ANTLR и Roslyn
Обработка древовидных структур и унифицированное AST

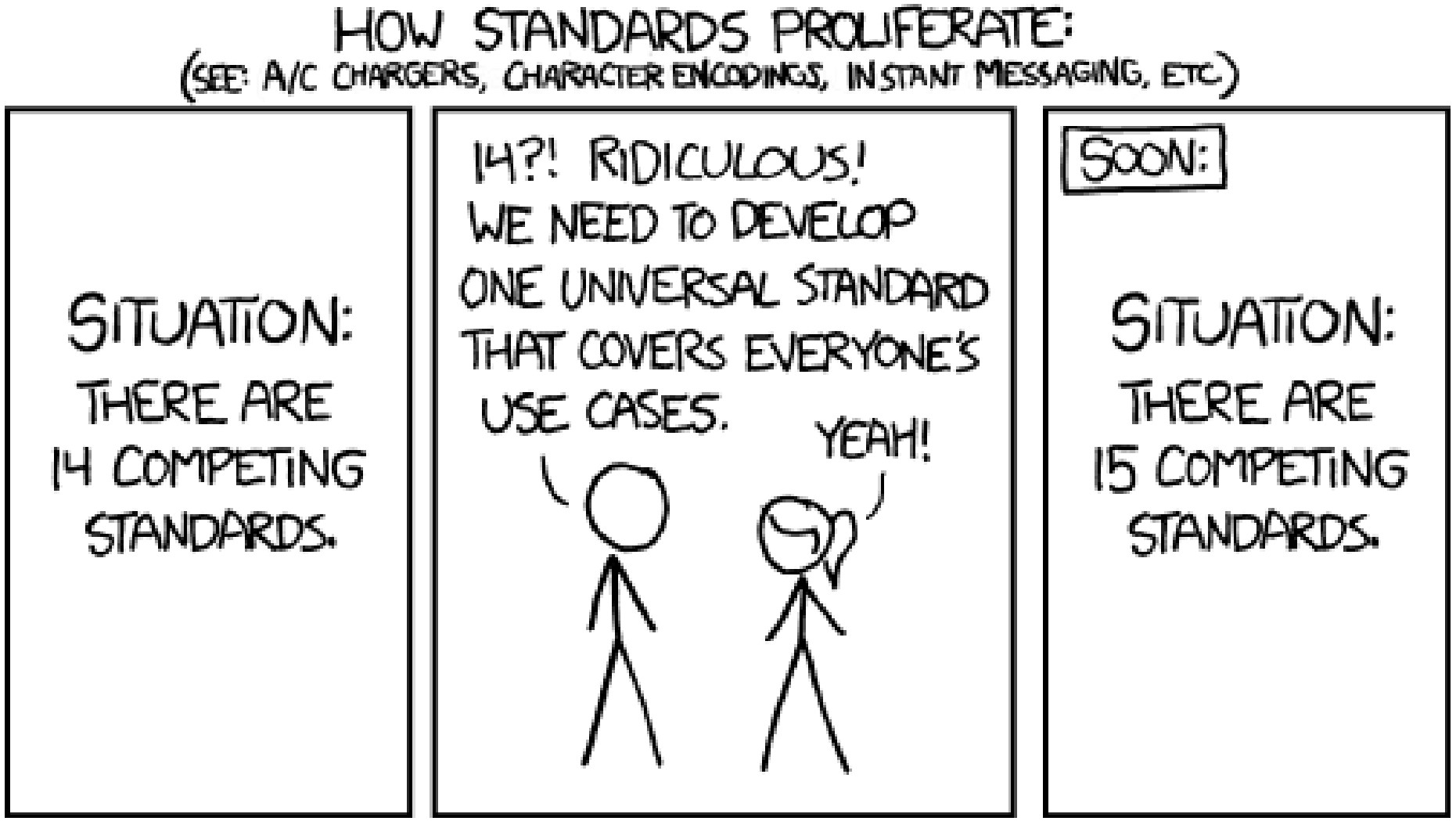
# Способы описания шаблонов



- Жесткое задание в коде
  - + Отсутствие какого-либо парсинга
  - − Громоздкий и нечитаемый синтаксис
  - − Невозможность добавления своих шаблонов без перекомпиляции
- Запись в формате обмена данных (JSON, YAML, XML)
  - + Простая реализация
  - − Громоздкий и трудночитаемый синтаксис
- Запись в формате предметно-ориентированного языка (DSL)
  - + Краткий и лаконичный синтаксис
  - − Необходимость разрабатывать грамматику и конвертер для нее

# Целесообразность DSL?





# DSL ≠ Regex

POSITIVE TECHNOLOGIES

DSL – это синтаксические конструкции
и регулярные выражения.

Регулярные выражения:

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-
\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-
\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

DSL – не нужно больше думать о ~~проблемах~~ пробелах:

```
"private_key_bits" = <[0..2048]>
```

# Синтаксические конструкции в DSL



`expr(args)` – вызов метода

`expr.Id` – обращение к члену объекта (поле, метод)

`Id expr = expr` – инициализация переменной

`expr + expr` – конкатенация

`new Id(args)` – создание объекта

`expr[expr]` – обращение по индексу или ключу

# Синтаксические конструкции в DSL

POSITIVE TECHNOLOGIES

- Примитивные типы в DSL

`Id, String, Int, Bool, Null`

- Синтаксис DSL

<[]> – оператор расширенного выражения (<[md5|sha1]> или <[0..2048]>)

# или <[expr]> – любое Expression

... или <[args]> – произвольное количество любых аргументов

(expr.)?expr – эквивалентно expr.expr или просто expr

<[~]>expr – отрицание условия

expr (<[||]> expr)* – объединение нескольких условий (ИЛИ)

Comment: "regex" – поиск по комментариям

# Примеры шаблонов на DSL

POSITIVE TECHNOLOGIES

- Hardcoded Password (All: C#, Java, PHP, PL/SQL, T-SQL)

```
(#.)?<[(?i)password(?-i)]> = <["\w*"]>
```

- Static Random Number Generator (C#, Java)

```
new Random(<[..]>)
```

- Debug Information Leak (PHP)

```
Configure.<[(?i)^write$(?-i)]>("debug", <[1..9]>)
```

- Insecure SSL connection (Java)

```
new AllowAllHostnameVerifier(...) <[||]> SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER
```

# Примеры шаблонов на DSL



- Password In Comment (All: C#, Java, PHP, PL/SQL, T-SQL)

```
Comment: <[ "(?i)password(?-i)\s*\=" ]>
```

- SQL Injection (All: C#, Java, PHP, PL/SQL, T-SQL)

```
<["(?i)select(?-i)\s\w*"]> + <[~"\w*"]>
```

- Cookie Without Secure Attribute (PHP)

```
session_set_cookie_params(#,#,#) // без четвертого аргумента
```

- Incorrect Permission Assignment For Critical Resource (PHP)

```
chmod(#, <[0777 || 02777 || 04777]>)
```

# Примеры шаблонов на DSL



- Empty try-catch block (All: C#, Java, PHP, PL/SQL, T-SQL)

```
try {...} catch { }
```

- Weak Cryptographic Algorithm (PHP)

```
mcrypt_encrypt(<[MCRYPT_DES || "des"]>, #, #, <[MCRYPT_MODE_ECB || "ecb"]>, #)
```

- Insecure Cookie (Java)

```
Cookie <[@cookie]> = new Cookie(...);
...
<[~]><[@cookie]>.setSecure(true);
...
response.addCookie(<[@cookie]>);
```

# Примеры шаблонов под PL/SQL и T-SQL

POSITIVE TECHNOLOGIES

- Erroneous Null Comparison (PL/SQL, T-SQL)

```
# == null <[||]> # != null
```

- Overly Broad Grant (PL/SQL, T-SQL)

```
grant_all(...)
```

Пример:

```
GRANT ALL ON employees TO john_doe;
```

# Незакрытый курсор в PL/SQL

POSITIVE TECHNOLOGIES

```
<[@cursor]> = DBMS_SQL.OPEN_CURSOR;
<[~]>DBMS_SQL.CLOSE_CURSOR(<[@cursor]>);
```

```
F1 := UTL_FILE.FOPEN('user_dir','u12345.tmp','R',256);
UTL_FILE.GET_LINE(F1,V1,32767);
-- UTL_FILE.FCLOSE(F1); is missing
```

# Незакрытый курсор в T-SQL

POSITIVE TECHNOLOGIES

```
declare_cursor(<[@cursor]>, ...);
<[~]>deallocate(<[@cursor]>);
```

```
DECLARE Employee_Cursor CURSOR FOR SELECT EmployeeID, Title FROM
AdventureWorks2012.HumanResources.Employee;
OPEN Employee_Cursor;
FETCH NEXT FROM Employee_Cursor;
--CLOSE Employee_Cursor; is missing
--DEALLOCATE Employee_Cursor; is missing
```

# Taint



- Taint > PM
- Анализ не только AST, но и CFG, DFG
- Учет семантики из других анализаторов (Roslyn для C#)
- Шаблоны Pattern Matching также будут валидны в Taint

# Шаблон Cross-Site Scripting (XSS) в Taint

POSITIVE TECHNOLOGIES

```
// TDE (Taint Data Entry)
<[@a]> = Request.<[Params|QueryString]>[#];
...
// TF (Transform function)
<[~]> <[@b]> =
<[System.Net.WebUtility.HtmlEncode]>(<[@a]>);
...
// PVF (Potential Vulnerability Function)
Response.Write(<[@b]>);
```

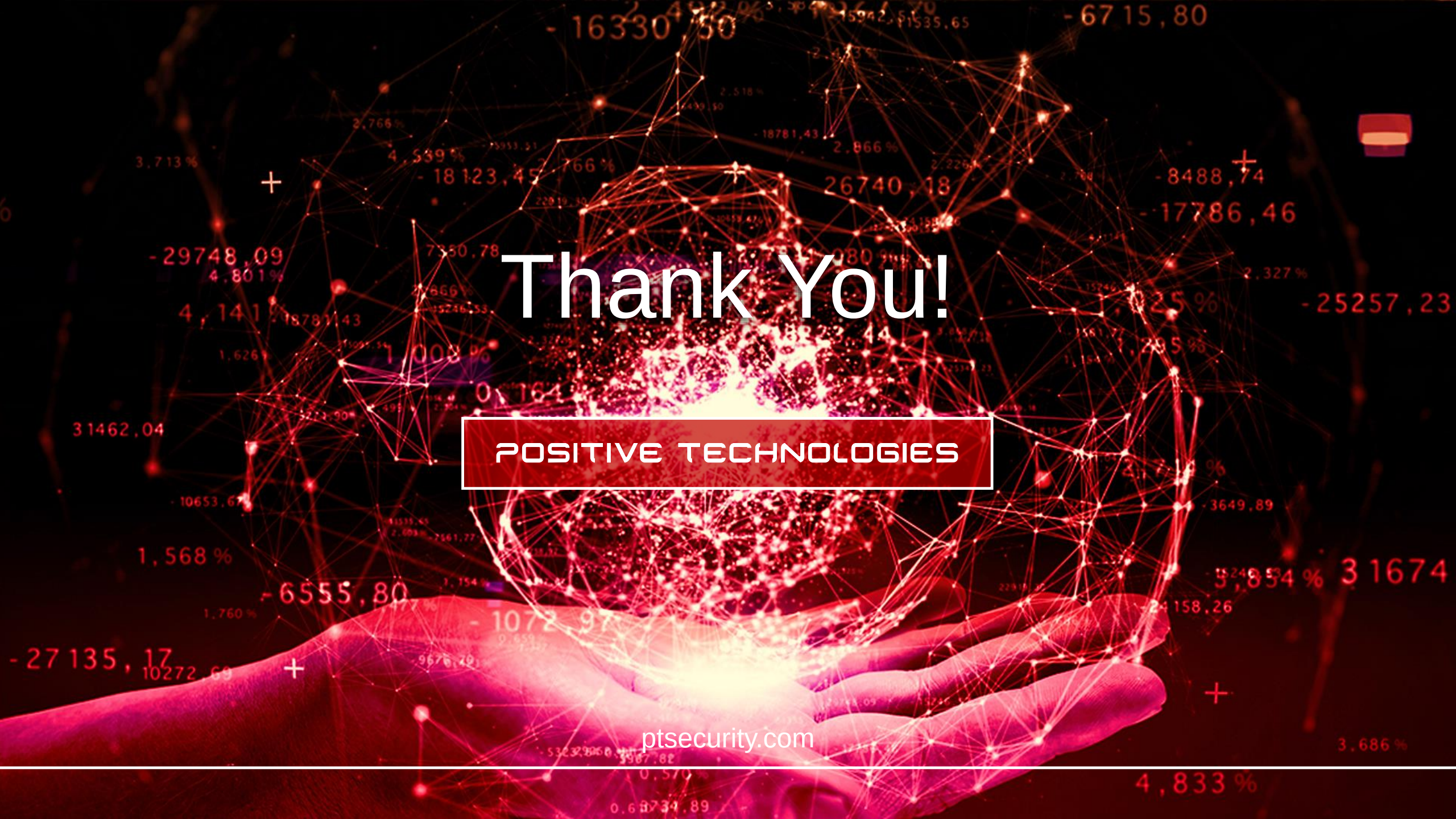
# Заключение

POSITIVE TECHNOLOGIES

Статьи с тегом [Application Inspector] на «Хабрахабре»

Грамматики PL/SQL, T-SQL, PHP на github.com/antlr/grammars-v4

Pattern Matching планируется включить в AppRoof


Thank You!
POSITIVE TECHNOLOGIES
ptsecurity.com