Под капотом SAST: как инструменты анализа кода ищут дефекты безопасности

Сергей Васильев
PVS-Studio LLC


'; DROP TABLE PLATES;
Audi - Vorsprung durch Technik

Name of the blog | Welcome to

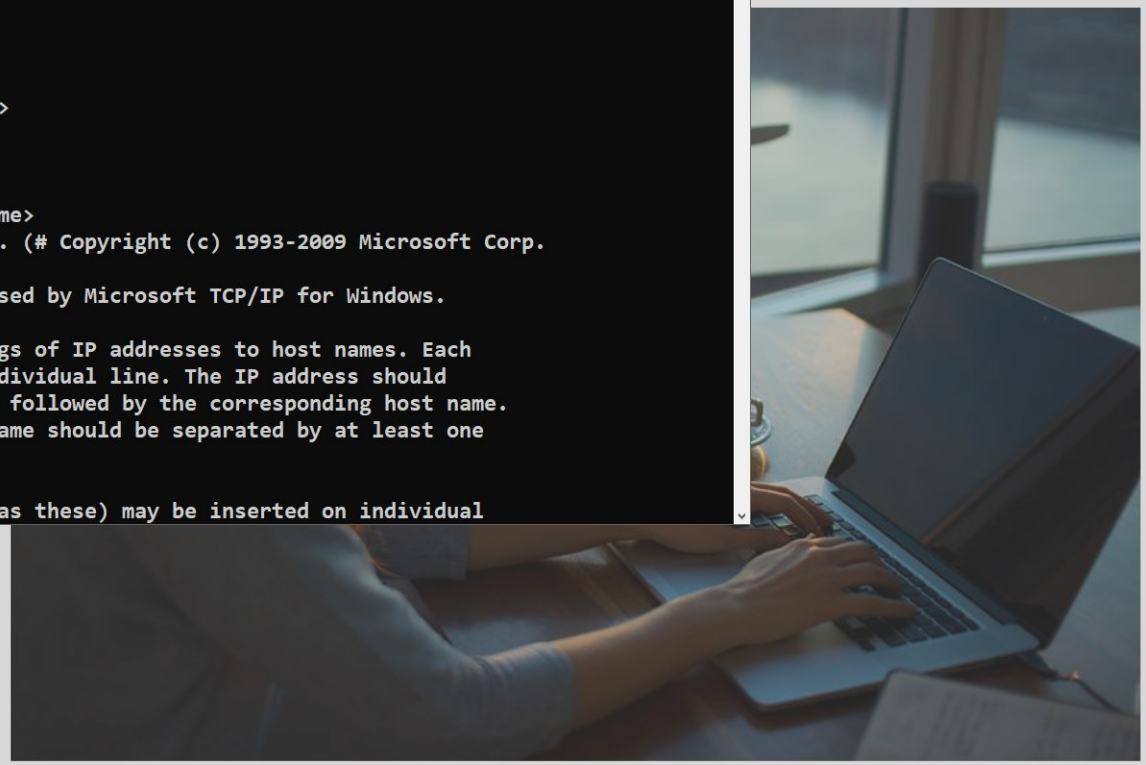
Not secure | vasiliev-pc:8081/post/welcome-to-blogengine-net

Welcome to BlogEngine.NET

C:\Windows\System32\cmd.exe

```
E:\XXE>curl -d "@xxe.xml" -X POST http://vasiliev-pc:8081/metaweblog.axd
 <?xml version="1.0" encoding="utf-8"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value>02</value>
        </member>
        <member>
          <name>faultString</name>
          <value>Unknown Method. (# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
```

share



If you see this post it means that BlogEngine.NET is running and the hard part of creating your own blog is done. There is only a few things left to do.

DOWNLOAD THEMES | OFFICIAL WEBSITE | DONATE

Write Permissions

Кто таков?

Сергей Васильев

Head of DevRel в PVS-Studio LLC

7 лет в статическом анализе

В прошлом:

- C++, C# developer
- Tools & DevOps Team Leader
- C# Analyzer Team Leader

Пишу на habr, выступаю

@_SergVasiliev_

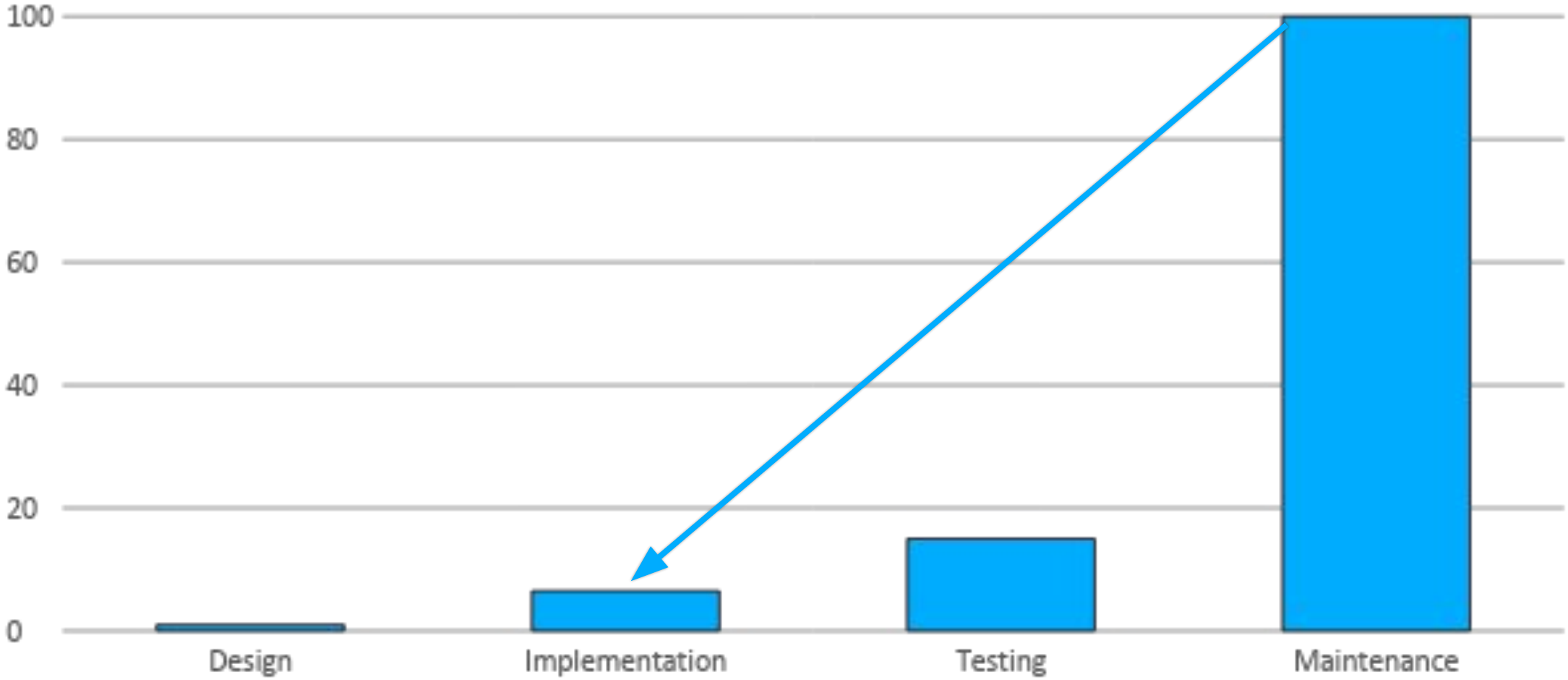
SAST

TechLead Conf 2022

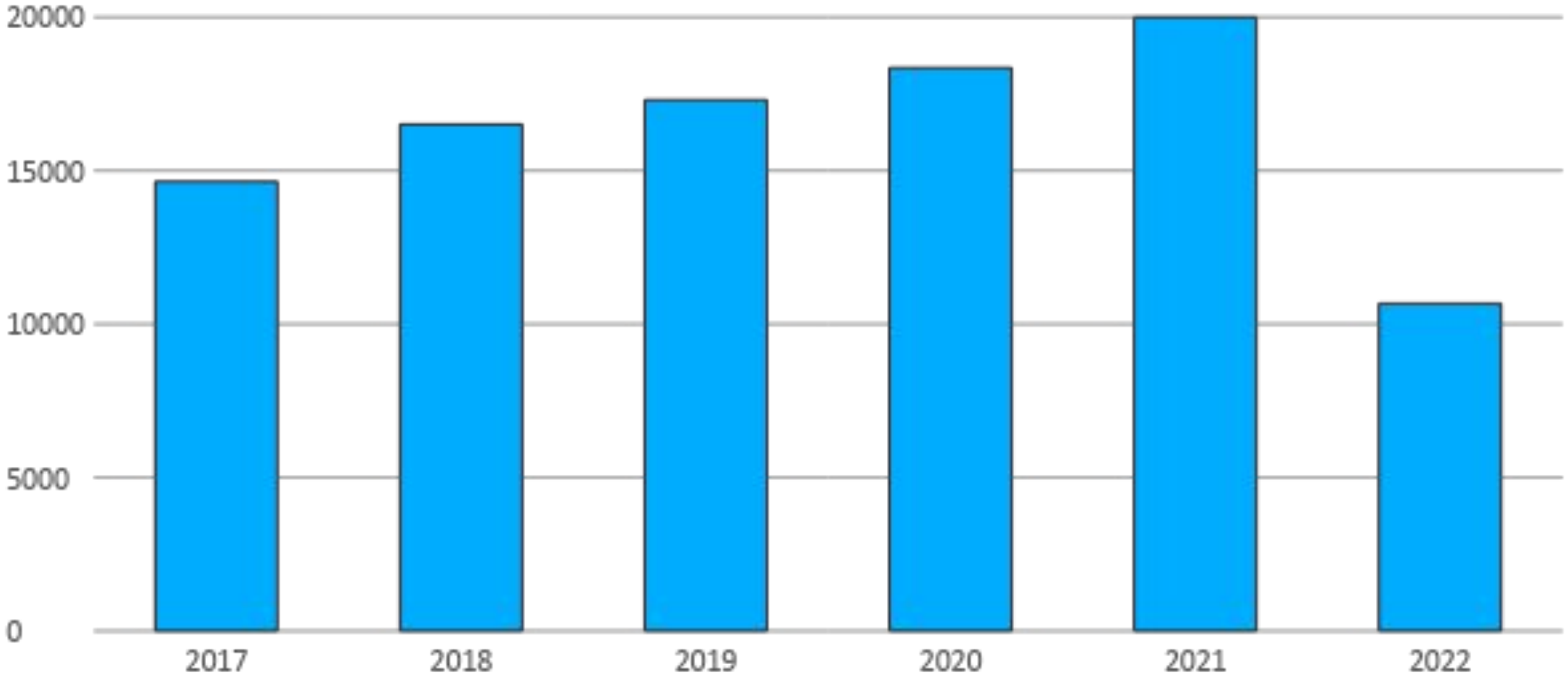
SAST (static application security testing)

- Анализ исходников на дефекты безопасности
- Shift-left
- Ищет:
  - SQL injection
  - Path traversal
  - XSS
  - ...

Стоимость исправления уязвимости



Количество уязвимостей




Sources
SAST
Report
(issues)

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Basis

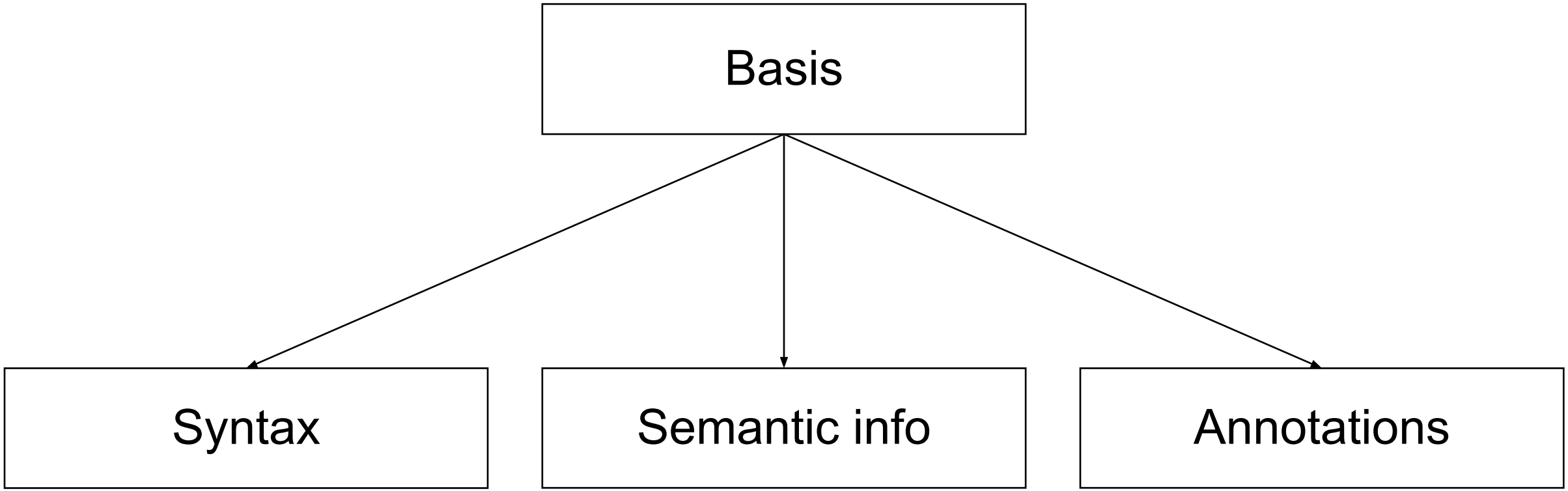
Pattern-based analysis

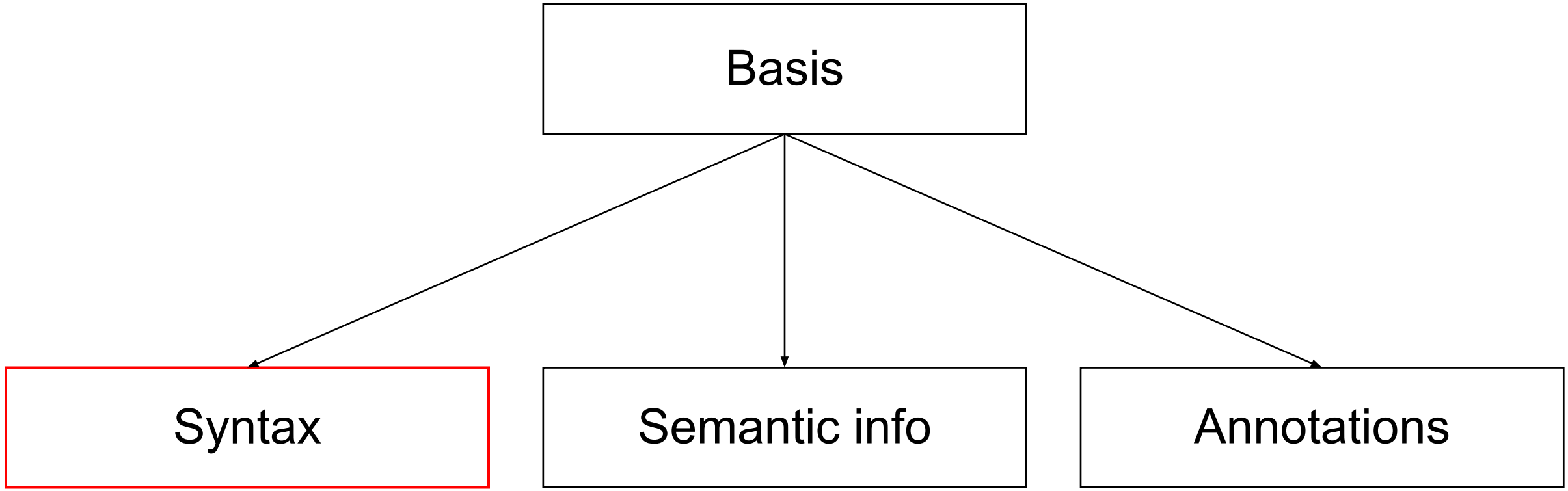
Data-flow analysis

Taint checking

Basis

TechLead Conf 2022


Basis
Syntax
Semantic info
Annotations


Basis
Syntax
Semantic info
Annotations

Syntax

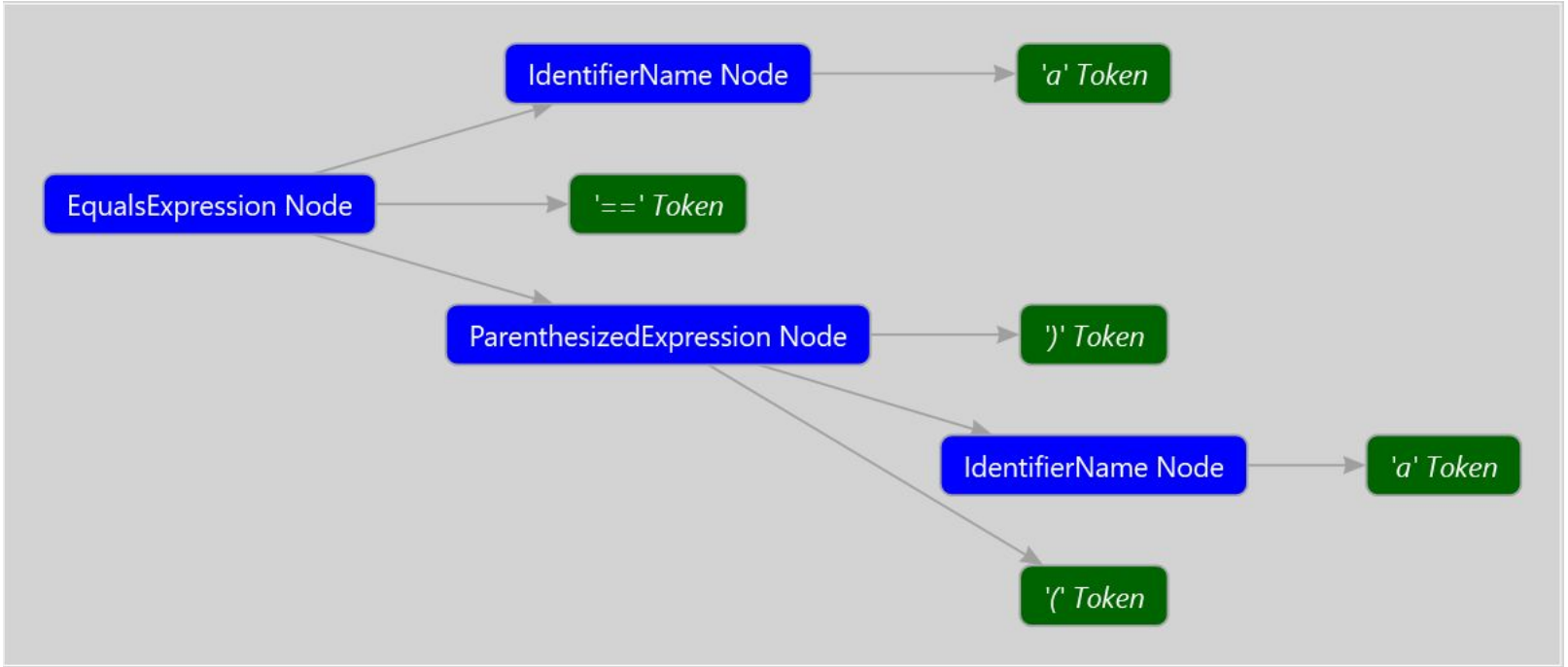
Syntax: A = A

```
if (A == A)
{ }

if (A == (A))
{ }

if (this.A == this.A)
{ }

if (A == this.A)
{ }
```

```
if (this.A == base.A)
if (this      base.A)
```

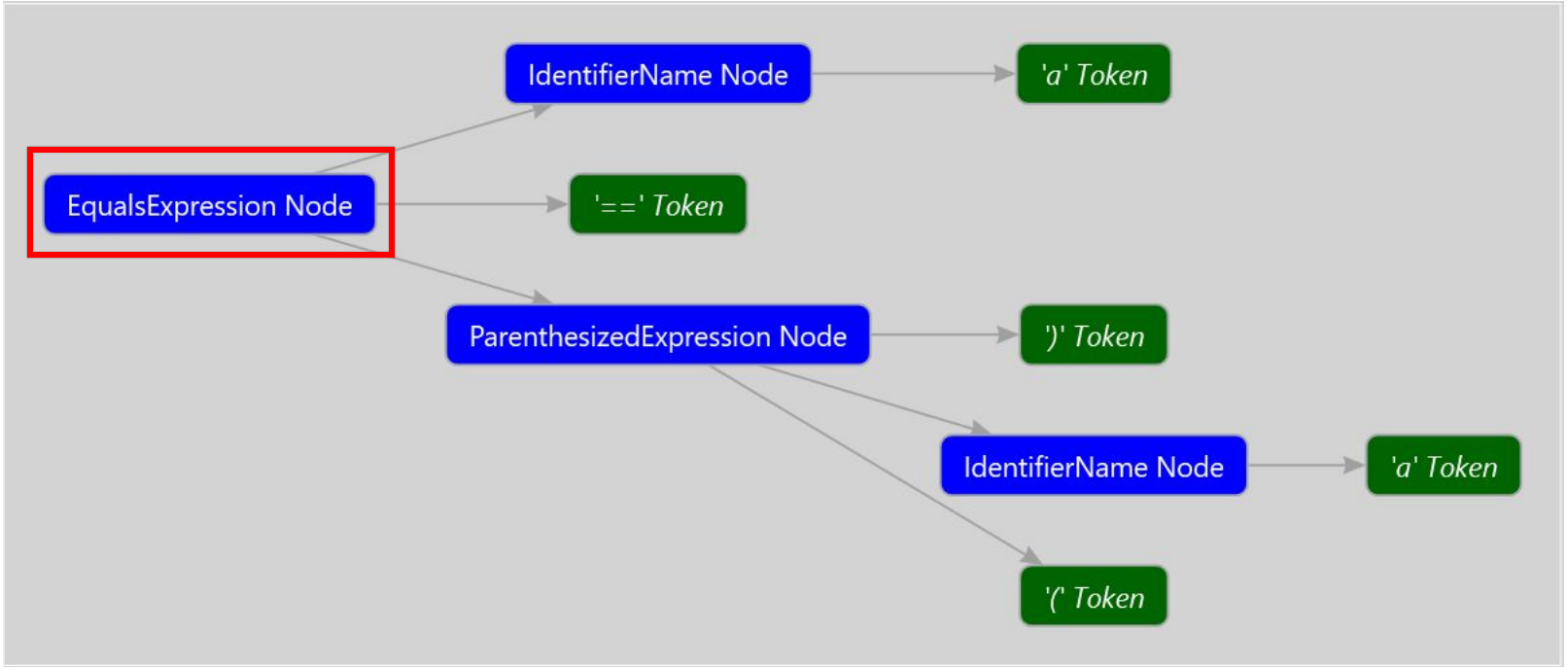
Syntax: A = A

```
if (a == (a))
{ }
```



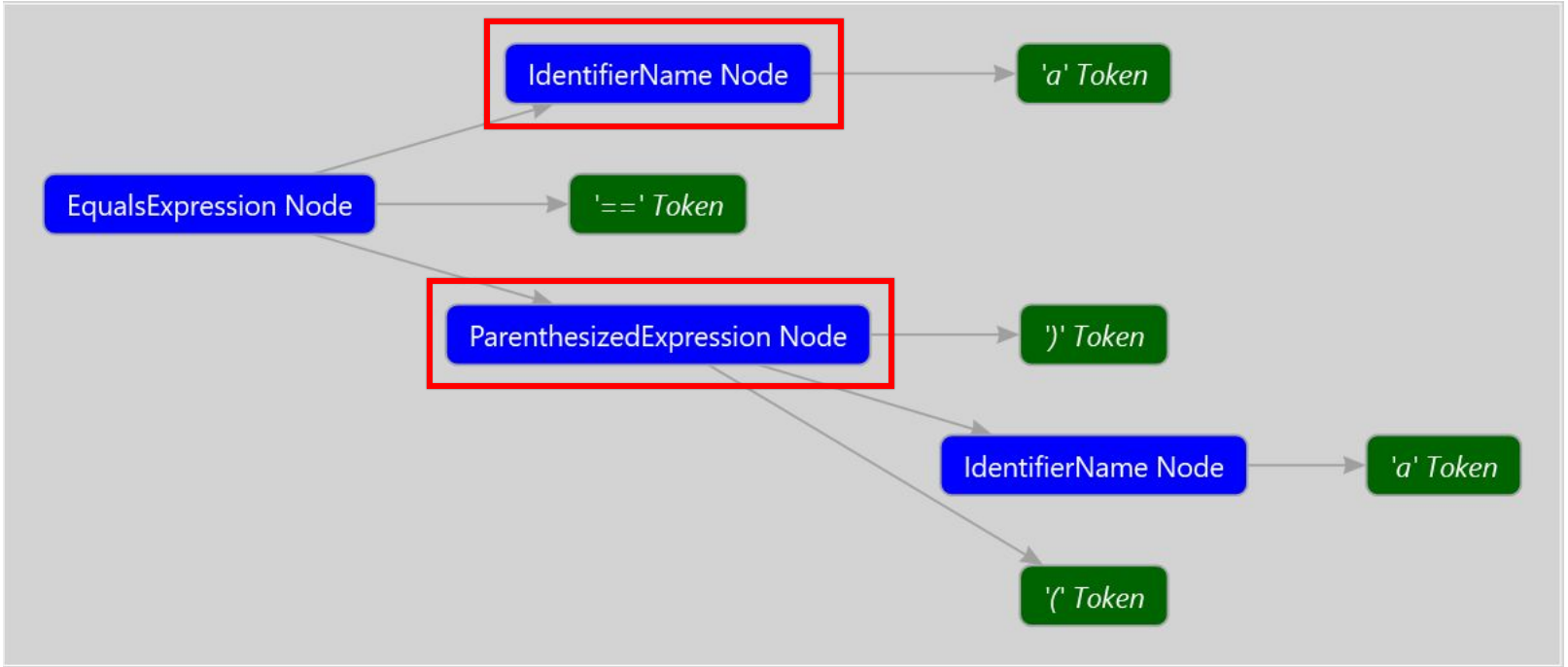
Syntax: A = A

```
if (a == (a))
{ }
```



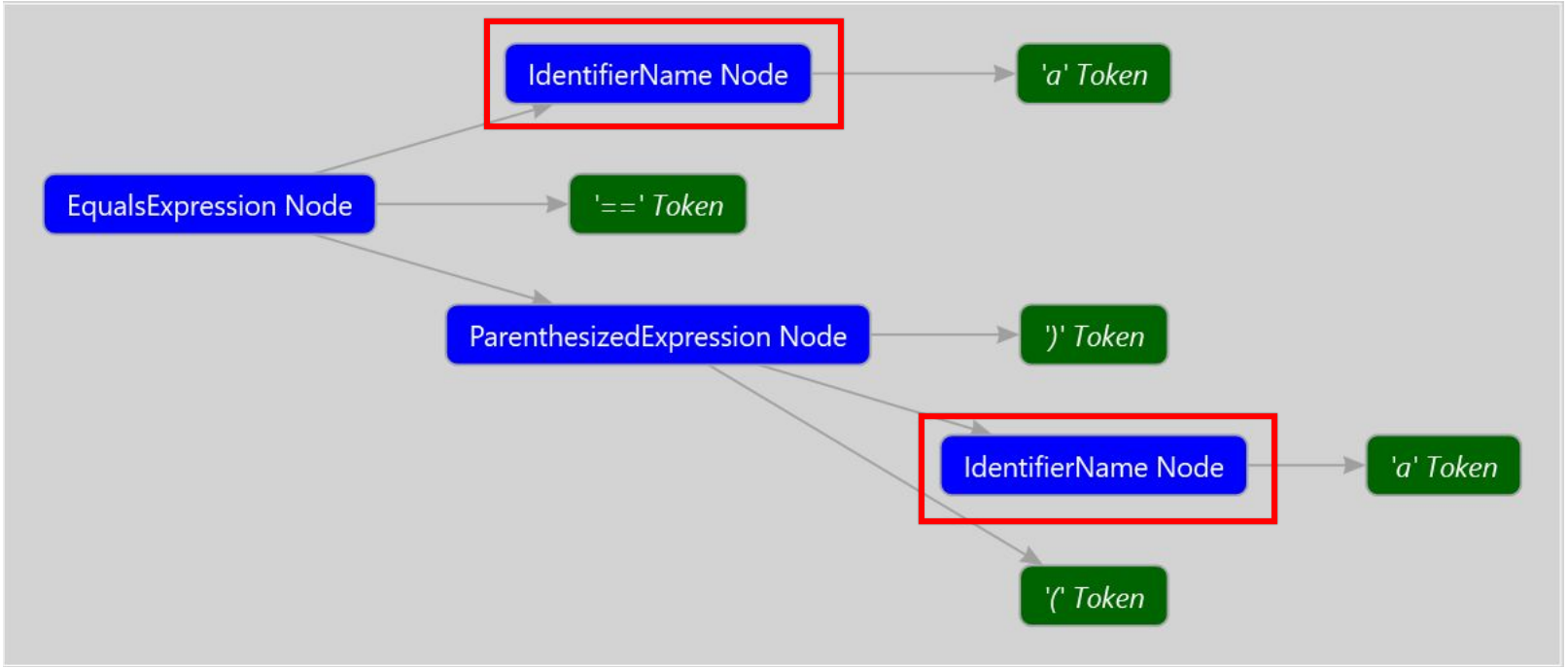
Syntax: A = A

```
if (a == (a))
{ }
```



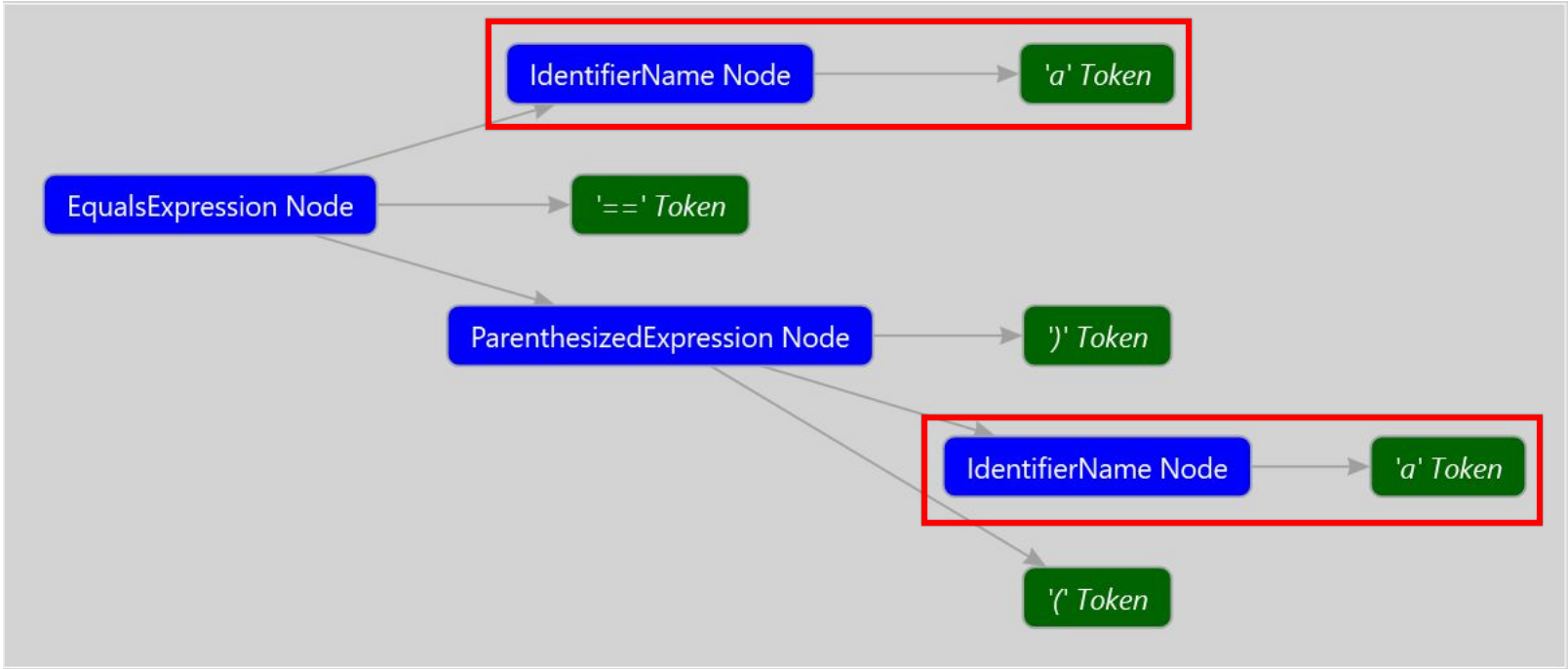
Syntax: A = A

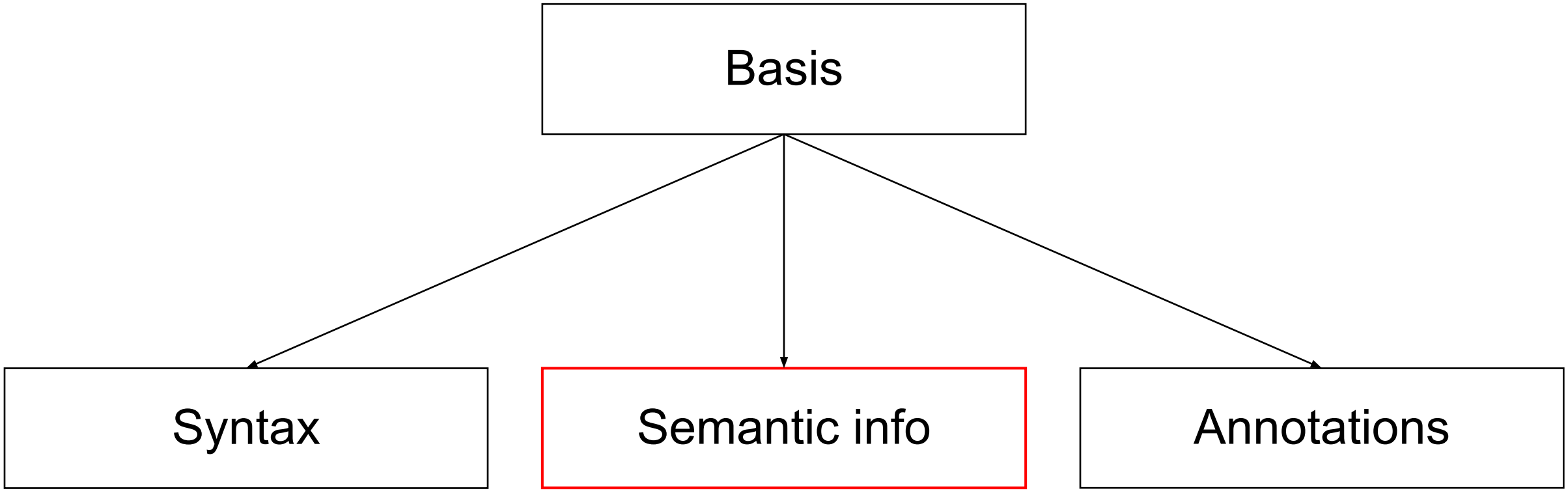
```
if (a == (a))
{ }
```



Syntax: A = A

```
if (a == (a))
{ }
```




Basis
Syntax
Semantic info
Annotations

Semantic info

Semantic info

```
if (lhsVar == rhsVar)
{
  // True
}
```

```
lhsVar: 0.5
rhsVar: 0.5
```


Semantic info

```
if (lhsVar == rhsVar)
{
  // False
}
```

```
lhsVar: 0.4999999999999999
rhsVar: 0.5
```


Semantic info

```
if (lhsVar == rhsVar)
{
 // False
}
```

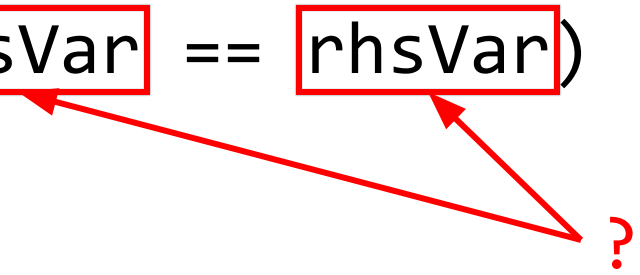
```
lhsVar: 62
rhsVar: 42
```


Semantic info

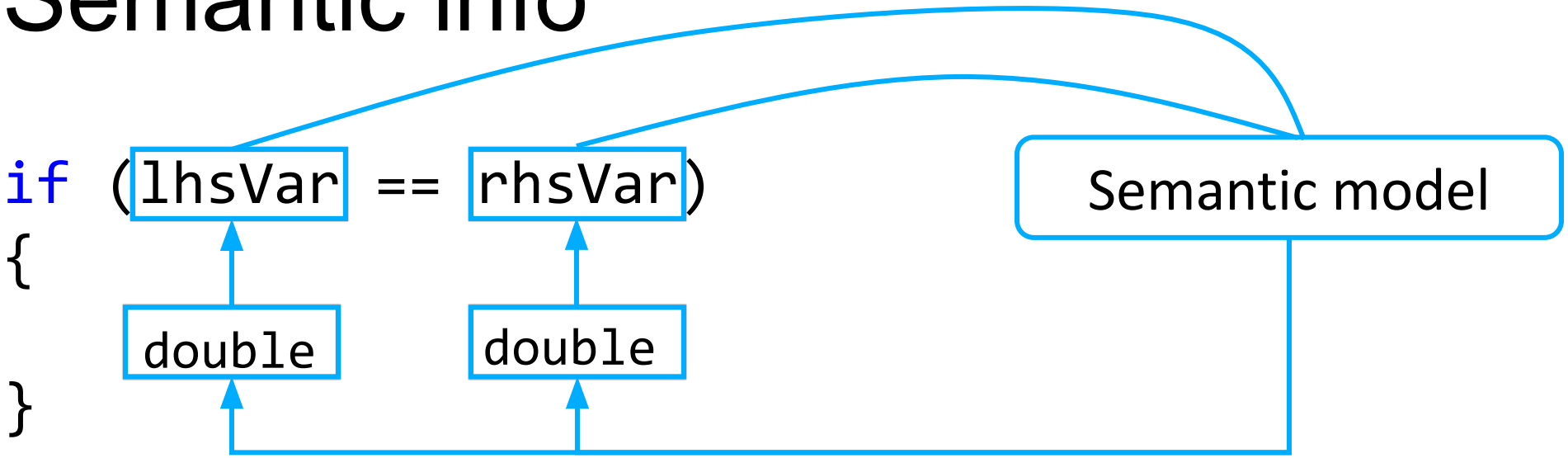
```
if (lhsVar == rhsVar)
{

}
```


Semantic info

```
if (lhsVar == rhsVar)
{

}
```



Semantic info



Semantic info

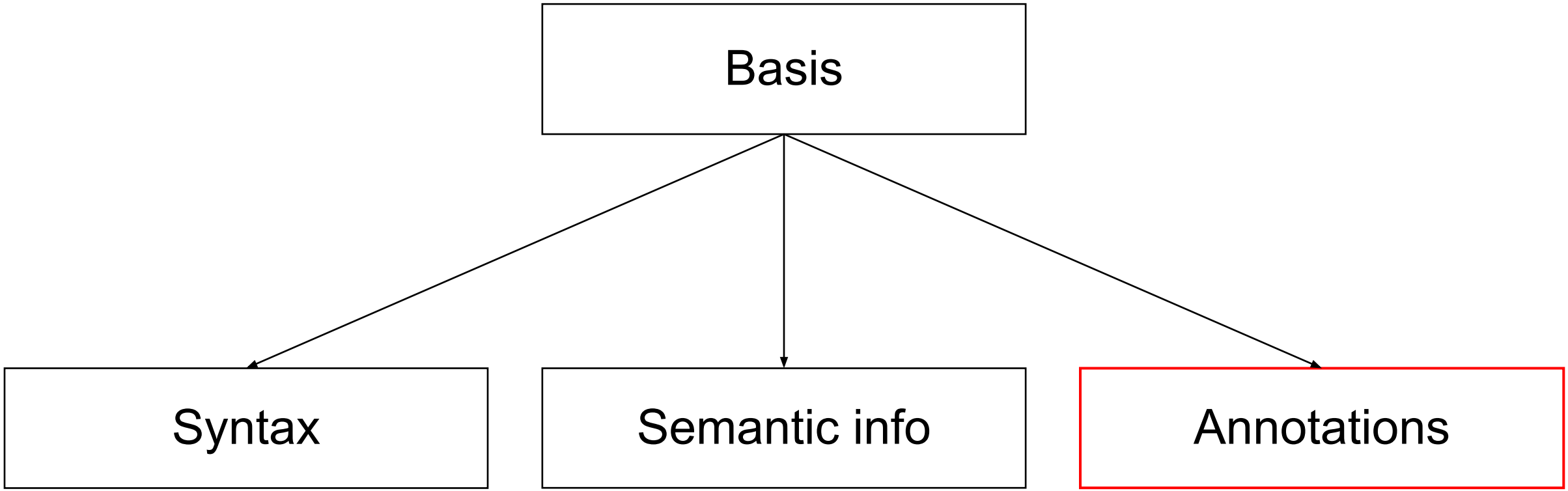
```
if (lhsVar == rhsVar)
{

}
```

Общая информация:

- сущность
- статический?
- абстрактный?
- виртуальный?
- ...

Тип:

- интерфейсы
- сборка
- значимый / ссылочный


Basis
Syntax
Semantic info
Annotations

Annotations

Annotations

```
var seq = null;
var list = Enumerable.ToList(seq);
if (list == null)
   ....
```


Annotations

```
var seq = null;
var list = Enumerable.ToList(seq);
if (list == null)
   ....
```


Annotations

```
var seq = null;
var list = Enumerable.ToList(seq);
if (list == null)
   ....
```


Annotations

```
var seq = null;
var list = Enumerable.ToList(seq);
if (list == null)
   ....
```


Annotations

```
List<TSource> Enumerable.ToList(arg)
```

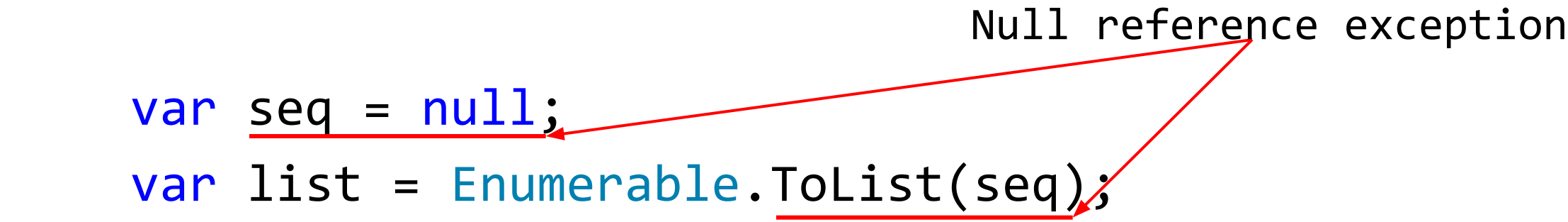

Annotations

```
List<TSource> Enumerable.ToList(arg)
```

Annotations

```
var seq = null;
var list = Enumerable.ToList(seq);
if (list == null)
   ....
```


Annotations



Annotations

```
var seq = null;
var list = Enumerable.ToList(seq);
if (list == null)
   ....
```

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Pattern-based analysis

TechLead Conf 2022

Pattern-based analysis

```
if (lhsVar == rhsVar)
{
  // False
}
```

```
lhsVar: 0.4999999999999999
rhsVar: 0.5
```


iOS: CVE-2014-1266

```
if ((err = SSLHashSHA1.update(
                &hashCtx, &signedParams)) != 0)
  goto fail;
  goto fail;
....
```


iOS: CVE-2014-1266

```
if ((err = SSLHashSHA1.update(
                &hashCtx, &signedParams)) != 0)
  goto fail;
  goto fail;
....
```

iOS: CVE-2014-1266

```
if ((err = SSLHashSHA1.update(
                &hashCtx, &signedParams)) != 0)
  goto fail;
goto fail;
....
```


iOS: CVE-2014-1266

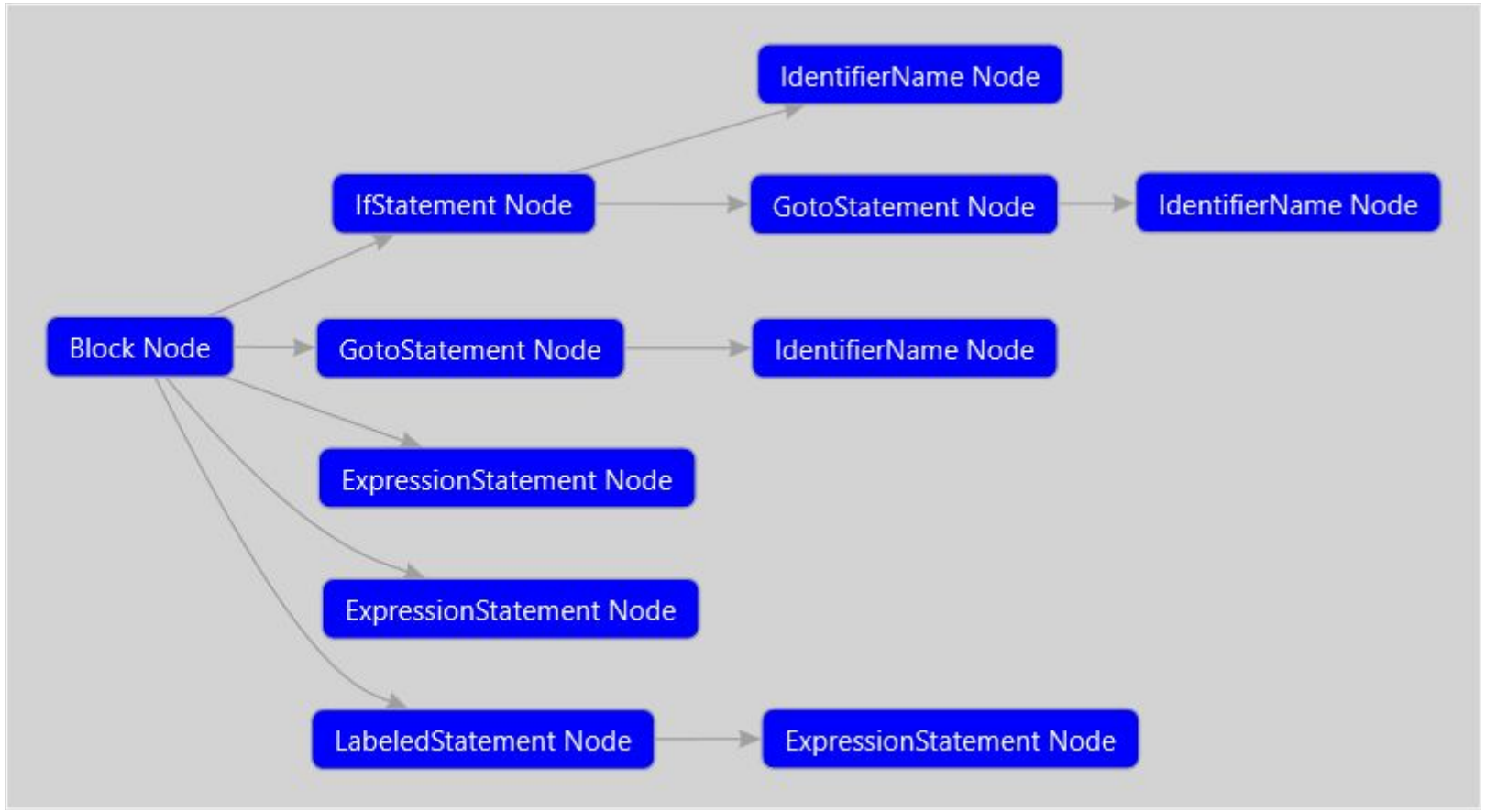
```
if ((err = SSLHashSHA1.update(
                &hashCtx, &signedParams)) != 0)
  goto fail;
  goto fail;
....
```

Способ #1	Способ #2
Безусловный goto	Форматирование vs логика исполнения

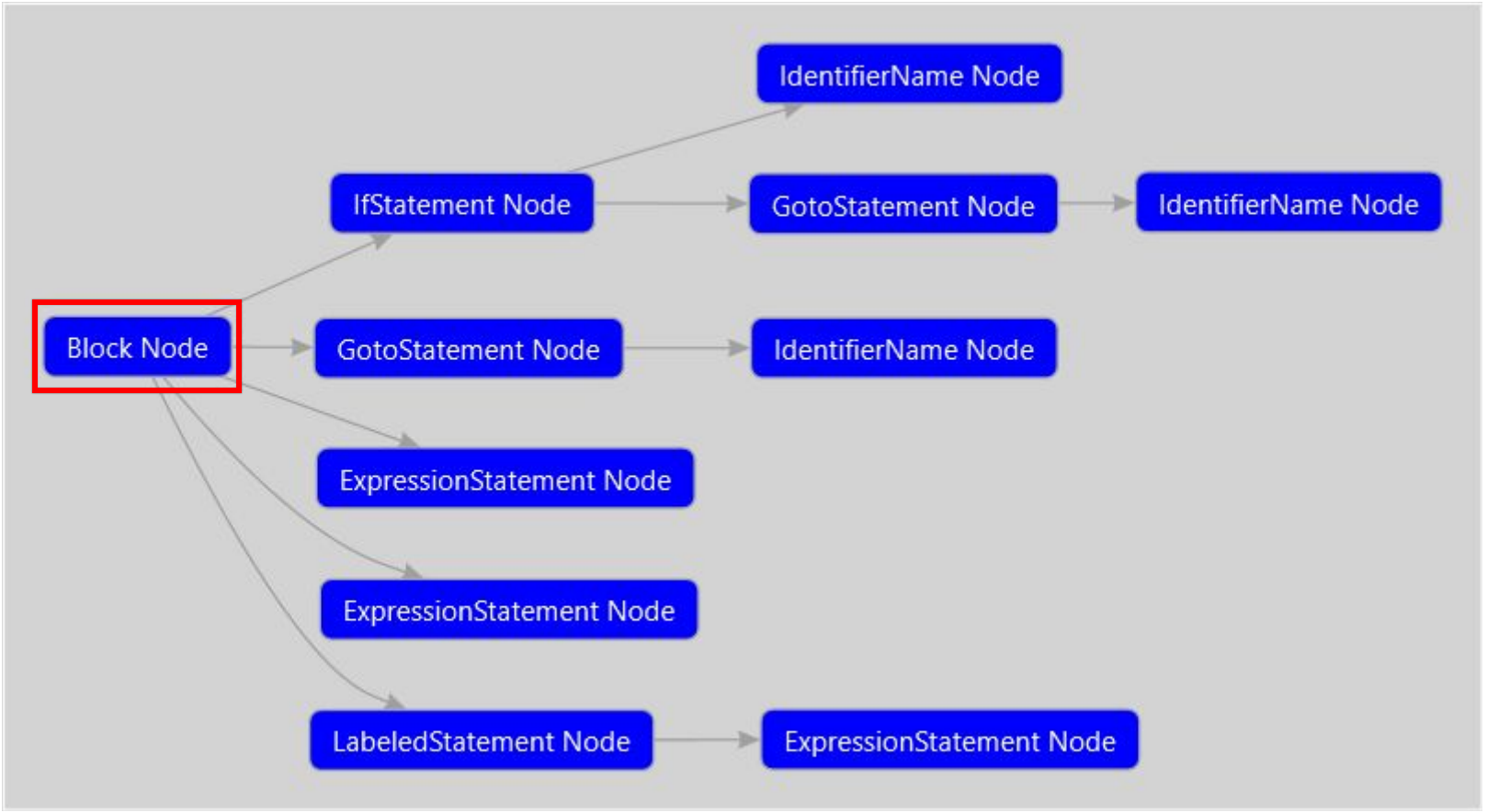
iOS: CVE-2014-1266

```
if ((err = SSLHashSHA1.update(
                &hashCtx, &signedParams)) != 0)
  goto fail;
  goto fail;
....
```

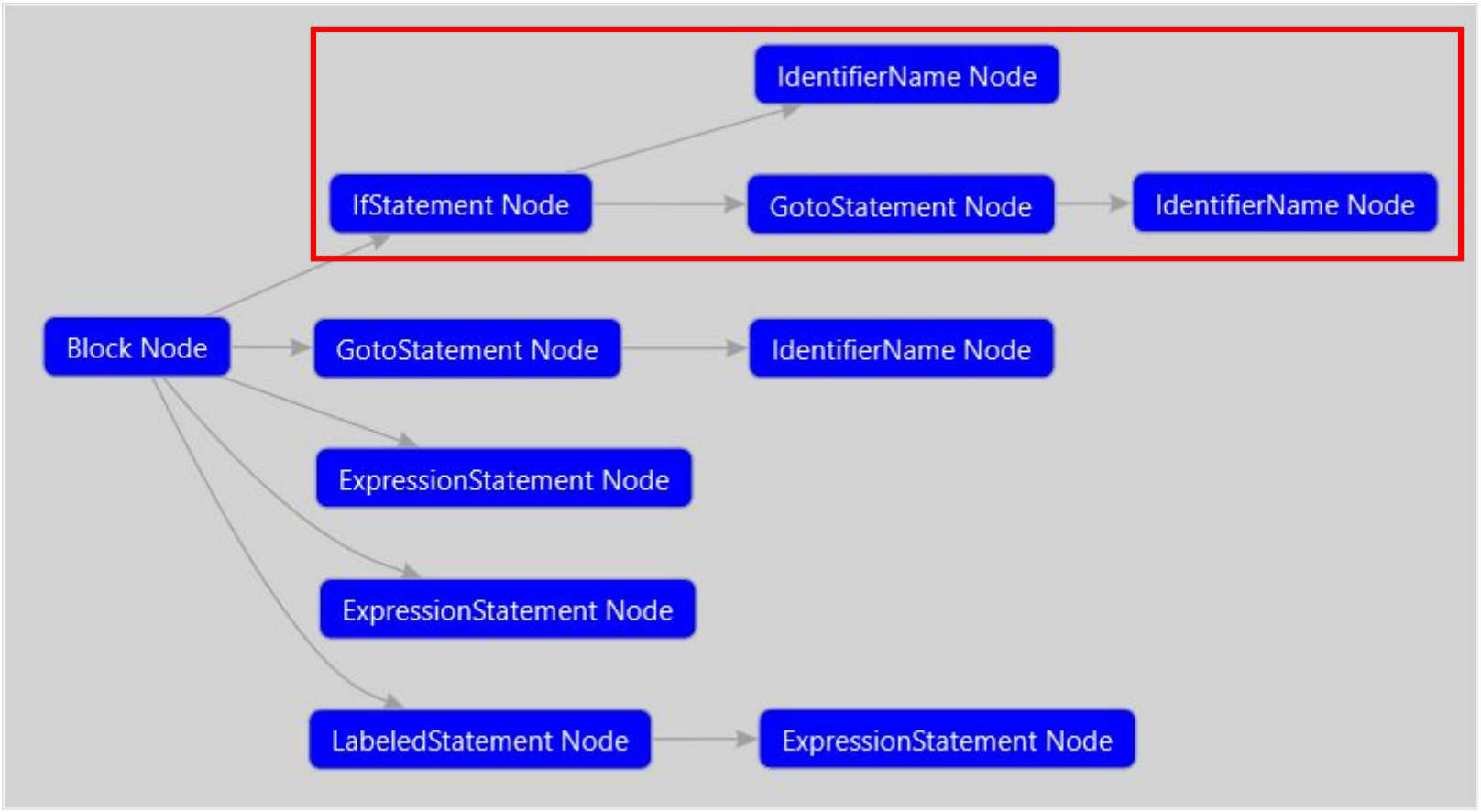

```
if (condition)
   goto fail;
   goto fail;
....
```



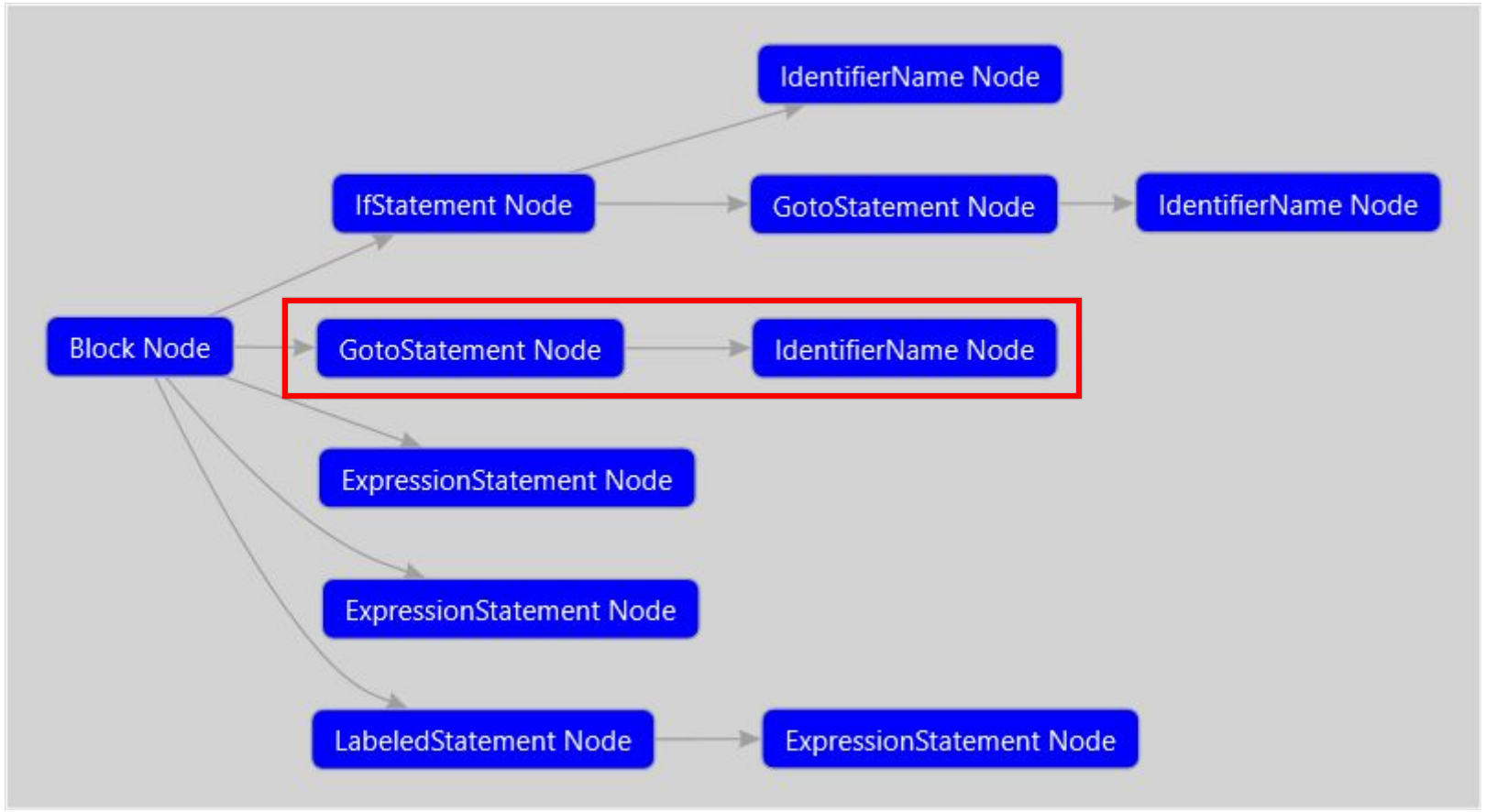
```
if (condition)
   goto fail;
   goto fail;
....
```



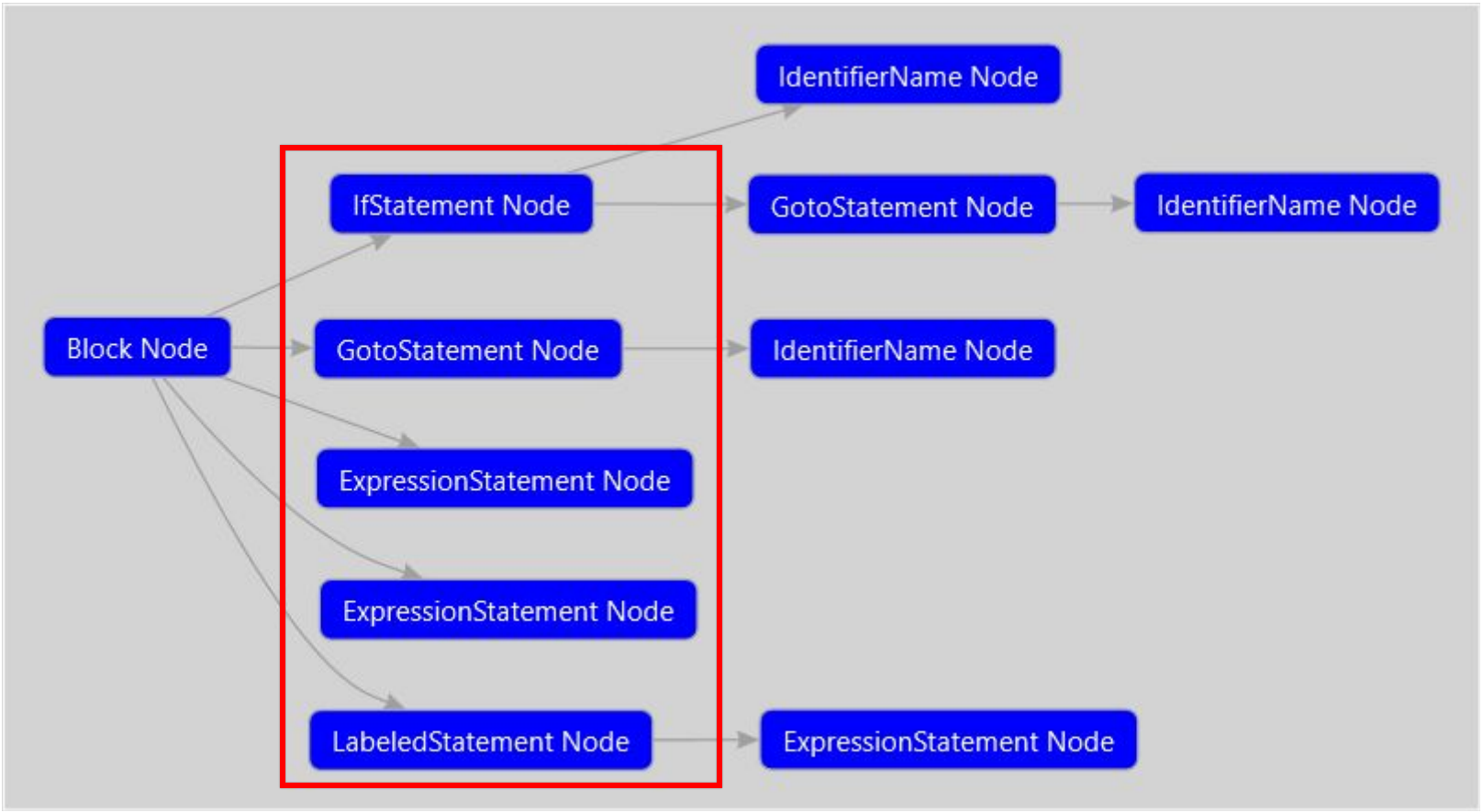
```
if (condition)
   goto fail;
   goto fail;
....
```



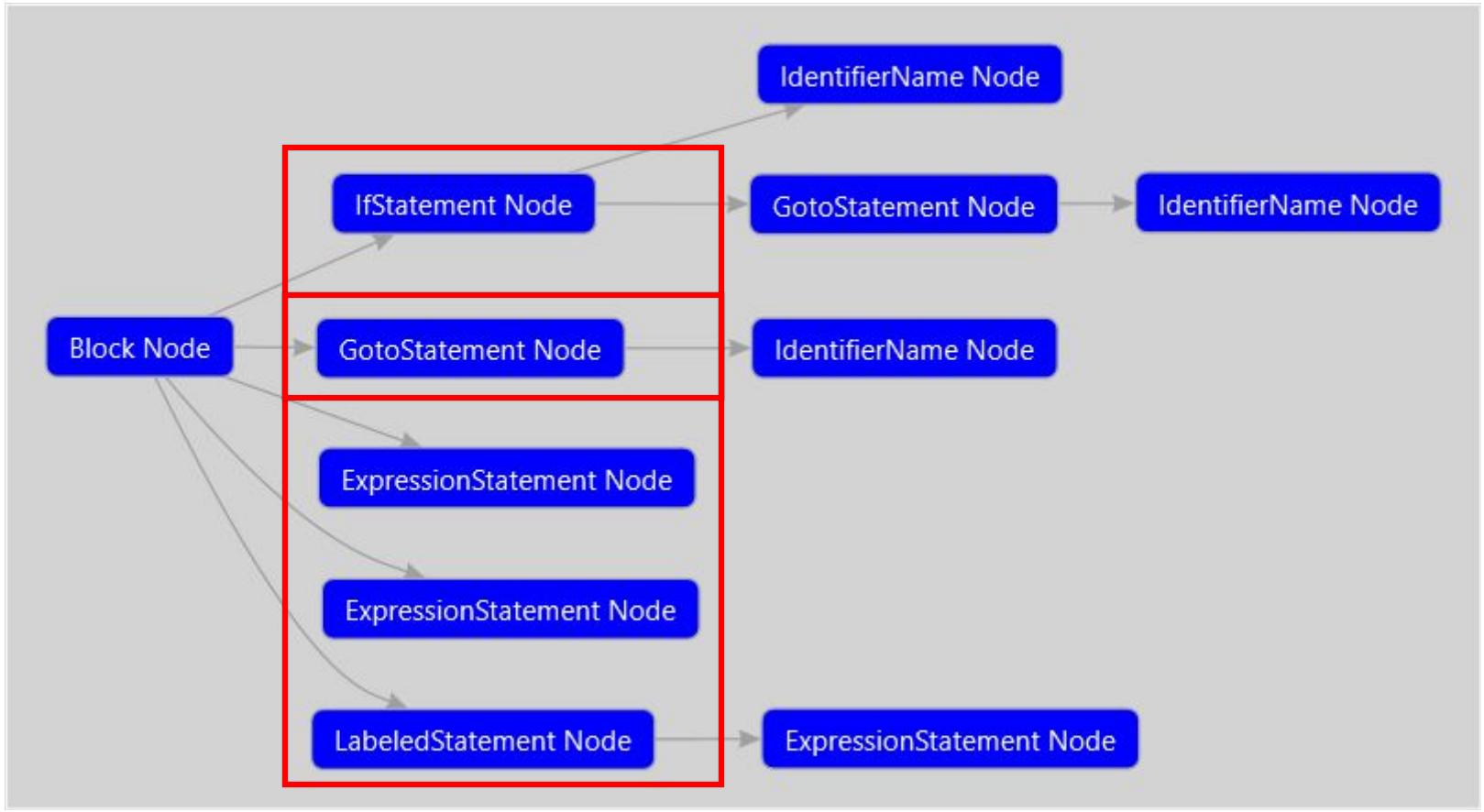
```
if (condition)
   goto fail;
   goto fail;
....
```



```
if (condition)
   goto fail;
   goto fail;
....
```



```
if (condition)
   goto fail;
   goto fail;
....
```



Форматирование vs логика исполнения

```
if ((err = SSLHashSHA1.update(
                &hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
....
```


MySQL: CVE-2012-2122

```
typedef char my_bool;
my_bool
check_scramble(const char  *scramble_arg,
               const char  *message,
               const uint8 *hash_stage2) {

  ....
  return memcmp(hash_stage2,
                hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```


MySQL: CVE-2012-2122

```
typedef char my_bool;
my_bool
check_scramble(const char  *scramble_arg,
               const char  *message,
               const uint8 *hash_stage2) {

  ....
  return memcmp(hash_stage2,
                hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```


memcmp

<cstring>

```
int memcmp ( const void * ptr1, const void * ptr2, size_t num );
```

Compare two blocks of memory

Compares the first *num* bytes of the block of memory pointed by *ptr1* to the first *num* bytes pointed by *ptr2*, returning zero if they all match or a value different from zero representing which is greater if they do not.

Notice that, unlike strcmp, the function does not stop comparing after finding a null character.

Parameters

`ptr1`
: Pointer to block of memory.

`ptr2`
: Pointer to block of memory.

`num`
: Number of bytes to compare.

Return Value

Returns an integral value indicating the relationship between the content of the memory blocks:

return value	indicates
`<0`	the first byte that does not match in both memory blocks has a lower value in *ptr1* than in *ptr2* (if evaluated as *unsigned char* values)
`0`	the contents of both memory blocks are equal
`>0`	the first byte that does not match in both memory blocks has a greater value in *ptr1* than in *ptr2* (if evaluated as *unsigned char* values)

MySQL: CVE-2012-2122

```
typedef char my_bool;
my_bool
check_scramble(const char  *scramble_arg,
               const char  *message,
               const uint8 *hash_stage2) {

  ....
  return memcmp(hash_stage2,
                hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```


MySQL: CVE-2012-2122

```
typedef char my_bool;
my_bool
check_scramble(const char  *scramble_arg,
               const char  *message,
               const uint8 *hash_stage2) {

  ....
  return memcmp(hash_stage2,
                hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```


MySQL: CVE-2012-2122

int -> char

```
typedef char my_bool;
my_bool
check_scramble(const char  *scramble_arg,
               const char  *message,
               const uint8 *hash_stage2) {

  ....
  return memcmp(hash_stage2,
                hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```


MySQL: CVE-2012-2122

```
typedef char my_bool;
my_bool
check_scramble(const char  *scramble_arg,
               const char  *message,
               const uint8 *hash_stage2) {

  ....
  return memcmp(hash_stage2,
                hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```


Как ловить?

```
resType variable = memcmp(....);
```

```
resType foo()
{
  return memcmp(....);
}
```

1. Находим вызов memcmp
2. Проверяем, что sizeof(resType) < sizeof(int)

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Data-flow analysis

TechLead Conf 2022

```
if (ptr || ptr->foo())
{
  // Do something
}
```


```
if (ptr || ptr->foo())
{
   // Do something
}
```


```
if (ptr || ptr->foo())
{
  // Do something
}
```


```
if (ptr)
{ .... }

auto test = ptr->foo();
```


```
if (ptr)
{ .... }

auto test = ptr->foo();
```


```
if (flag)
{
   ptr = nullptr;
}

auto test = ptr->foo();
```


```
if (flag)
{
  ptr = nullptr;
}

if (ptr) {
  auto test = ptr->foo();
}
```


Data-flow analysis

Вычисление значений выражений

- booleans: true / false
- integers: ranges
- pointers: null-state analysis

```
void DataFlowTest(short x) {
   if (x > int.MaxValue)
      ....
}
```


CWE-570: Expression is Always False

```
void DataFlowTest(short x) {
  if (x > int.MaxValue) // always false
    ....
}
```


CWE-570: Expression is Always False

```
void DataFlowTest(short x) {
   if (x > int.MaxValue) // always false
      ....
}
```


CWE-570: Expression is Always False

```
void DataFlowTest(short x) {
  if (x > int.MaxValue) // always false
    ....
}
```

x (short): [-32768; 32767]

CWE-570: Expression is Always False

```
void DataFlowTest(short x) {
  if (x > int.MaxValue) // always false
    ....
}
```

```
x (short):     [-32768; 32767]
int.MaxValue: [  2147483647 ]
```


CWE-570: Expression is Always False

```
void DataFlowTest(short x) {
  if (x > int.MaxValue) // always false
    ....
}
```

```
x (short):     [-32768; 32767]
int.MaxValue: [  2147483647 ]
```


CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```


CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```


CWE-570: Expression is Always False

```
void DataFlowTest(int x) { x: [-2147483648; 2147483647]
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```


CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```

x: [-2147483648; 2147483647]

x: [11; 2147483647]

CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```

x: [-2147483648; 2147483647]

x: [11; 2147483647]

y: [1; 2147483637]

CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```

```
x: [-2147483648; 2147483647]
x: [ 11; 2147483647]
y: [ 1; 2147483637]
```


CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0)
      ....
    if (y <= 1)
      ....
  }
}
```

x: [-2147483648; 2147483647]
x: [11; 2147483647]
y: [1; 2147483637]

CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0) // expression is always false
      ....
    if (y <= 1)
      ....
  }
}
```

```
x: [-2147483648; 2147483647]
x: [ 11; 2147483647]
y: [ 1; 2147483637]
```


CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0) // expression is always false
      ....
    if (y <= 1)
      ....
  }
}
```

x: [-2147483648; 2147483647]
x: [11; 2147483647]
y: [1; 2147483637]

CWE-570: Expression is Always False

```
void DataFlowTest(int x) {
  if (x > 10) {
    var y = x - 10;
    if (y < 0) // expression is always false
      ....
    if (y <= 1) // ok
      ....
  }
}
```

x: [-2147483648; 2147483647]
x: [11; 2147483647]
y: [1; 2147483637]

CWE-476: NULL Pointer Dereference

```
void PtrDeref(....) {
  const Wrapper* ptr = flag ? &wrapper : nullptr;
  // ....
  if (anotherFlag) {
    ptr->foo();
  }
}
```


CWE-476: NULL Pointer Dereference

```
void PtrDeref(....) {
  const Wrapper* ptr = flag ? &wrapper : nullptr;
  // ....
  if (anotherFlag) {
    ptr->foo();       ptr: [potential null]
  }
}
```


ytnef: CVE-2017-6298

```
....
TNEF->subject.data = calloc(size, sizeof(BYTE));
TNEF->subject.size = vl->size;
memcpy(TNEF->subject.data, vl->data, vl->size);
....
```


ytnef: CVE-2017-6298

```
....
TNEF->subject.data = calloc(size, sizeof(BYTE));
TNEF->subject.size = vl->size;
memcpy(TNEF->subject.data, vl->data, vl->size);
....
```


std::memcpy

Defined in header <cstring>

```
void* memcpy( void* dest, const void* src, std::size_t count );
```

Copies count bytes from the object pointed to by src to the object pointed to by dest. Both objects are reinterpreted as arrays of `unsigned char`.

If the objects overlap, the behavior is undefined.

If either dest or src is an invalid or null pointer, the behavior is undefined, even if count is zero.

If the objects are potentially-overlapping or not *TriviallyCopyable*, the behavior of memcpy is not specified and may be undefined.

Parameters

dest	-	pointer to the memory location to copy to
src	-	pointer to the memory location to copy from
count	-	number of bytes to copy

std::memcpy

Defined in header <cstring>

```
void* memcpy( void* dest, const void* src, std::size_t count );
```

Copies count bytes from the object pointed to by src to the object pointed to by dest. Both objects are reinterpreted as arrays of `unsigned char`.

If the objects overlap, the behavior is undefined.

If either dest or src is an invalid or null pointer, the behavior is undefined, even if count is zero.

If the objects are potentially-overlapping or not *TriviallyCopyable*, the behavior of memcpy is not specified and may be undefined.

Parameters

dest	-	pointer to the memory location to copy to
src	-	pointer to the memory location to copy from
count	-	number of bytes to copy

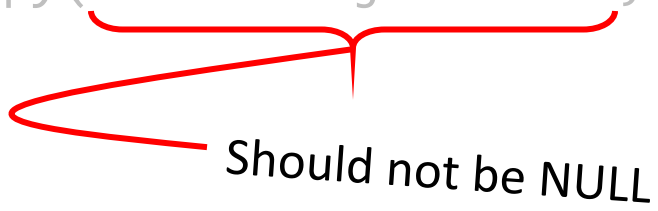
ytnef: CVE-2017-6298

```
....
TNEF->subject.data = calloc(size, sizeof(BYTE));
TNEF->subject.size = vl->size;
memcpy(TNEF->subject.data, vl->data, vl->size);
....
```

Should not be NULL

ytnef: CVE-2017-6298

```
....
TNEF->subject.data = calloc(size, sizeof(BYTE));
TNEF->subject.size = vl->size;
memcpy(TNEF->subject.data, vl->data, vl->size);
....
```



Defined in header <stdlib.h>

```
void* calloc( size_t num, size_t size );
```

Allocates memory for an array of `num` objects of `size` and initializes all bytes in the allocated storage to zero.

If allocation succeeds, returns a pointer to the lowest (first) byte in the allocated memory block that is suitably aligned for any object type.

If `size` is zero, the behavior is implementation defined (null pointer may be returned, or some non-null pointer may be returned that may not be used to access storage)

`calloc` is thread-safe: it behaves as though only accessing the memory locations visible through its argument, and not any static storage.

A previous call to `free` or `realloc` that deallocates a region of memory *synchronizes-with* a call to **`calloc`** that allocates the same or a part of the same region of memory. This synchronization occurs after any access to the memory by the deallocating function and before any access to the memory by `calloc`. There is a single total order of all allocation and deallocation functions operating on each particular region of memory. (since C11)

Parameters

`num` - number of objects

`size` - size of each object

Return value

On success, returns the pointer to the beginning of newly allocated memory. To avoid a memory leak, the returned pointer must be deallocated with `free()` or `realloc()`.

On failure, returns a null pointer.

Defined in header <stdlib.h>

```
void* calloc( size_t num, size_t size );
```

Allocates memory for an array of `num` objects of `size` and initializes all bytes in the allocated storage to zero.

If allocation succeeds, returns a pointer to the lowest (first) byte in the allocated memory block that is suitably aligned for any object type.

If `size` is zero, the behavior is implementation defined (null pointer may be returned, or some non-null pointer may be returned that may not be used to access storage)

`calloc` is thread-safe: it behaves as though only accessing the memory locations visible through its argument, and not any static storage.

A previous call to `free` or `realloc` that deallocates a region of memory *synchronizes-with* a call to **`calloc`** that allocates the same or a part of the same region of memory. This synchronization occurs after any access to the memory by the deallocating function and before any access to the memory by `calloc`. There is a single total order of all allocation and deallocation functions operating on each particular region of memory. (since C11)

Parameters

`num` - number of objects

`size` - size of each object

Return value

On success, returns the pointer to the beginning of newly allocated memory. To avoid a memory leak, the returned pointer must be deallocated with `free()` or `realloc()`.

On failure, returns a null pointer.

ytnef: CVE-2017-6298

```
....
TNEF->subject.data = calloc(size, sizeof(BYTE));
TNEF->subject.size = vl->size;
memcpy(TNEF->subject.data, vl->data, vl->size);
....
```

Potential NULL

Should not be NULL

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Basis

Pattern-based analysis

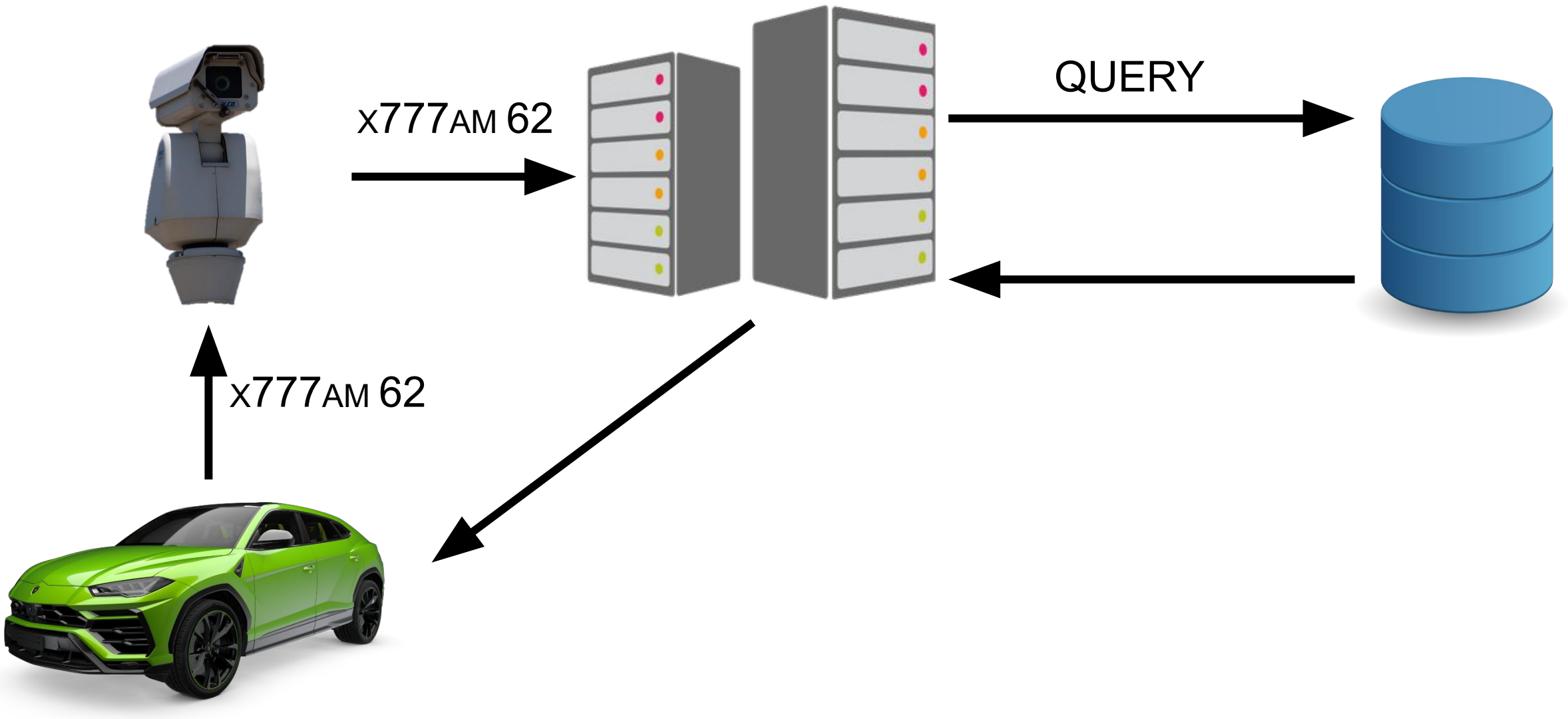
Data-flow analysis

Taint checking

Taint analysis

TechLead Conf 2022


'; DROP TABLE PLATES;
Audi - Vorsprung durch Technik

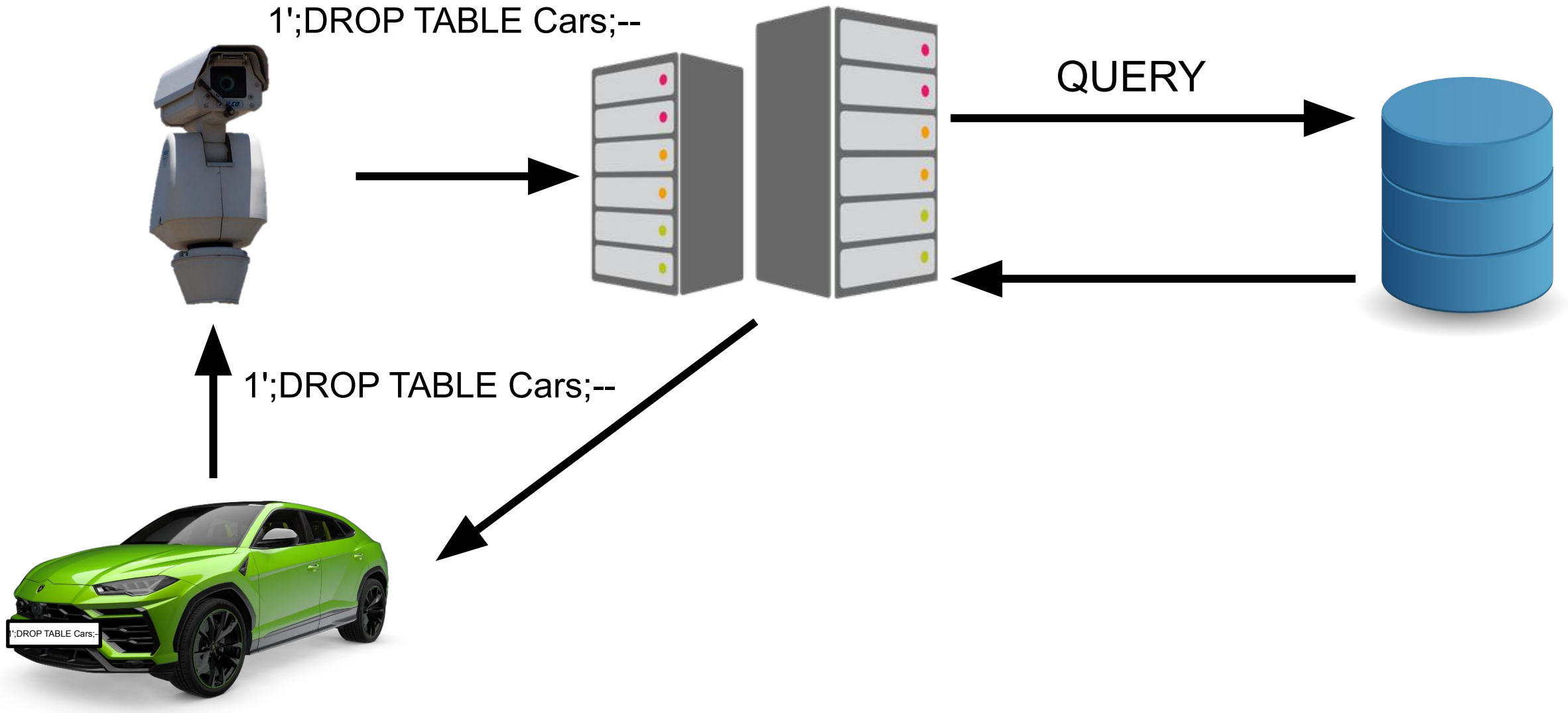

x777AM 62
QUERY
x777AM 62

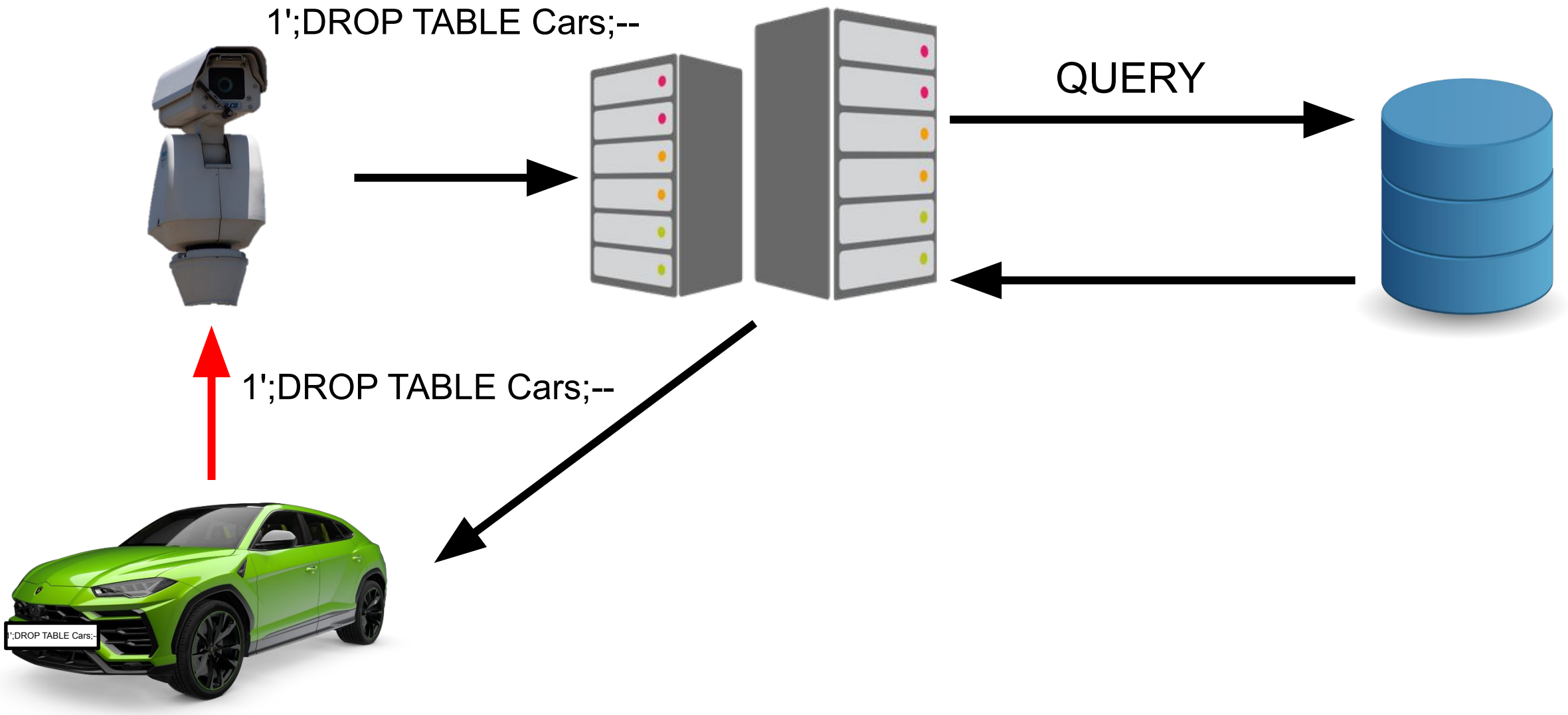
SQL injection

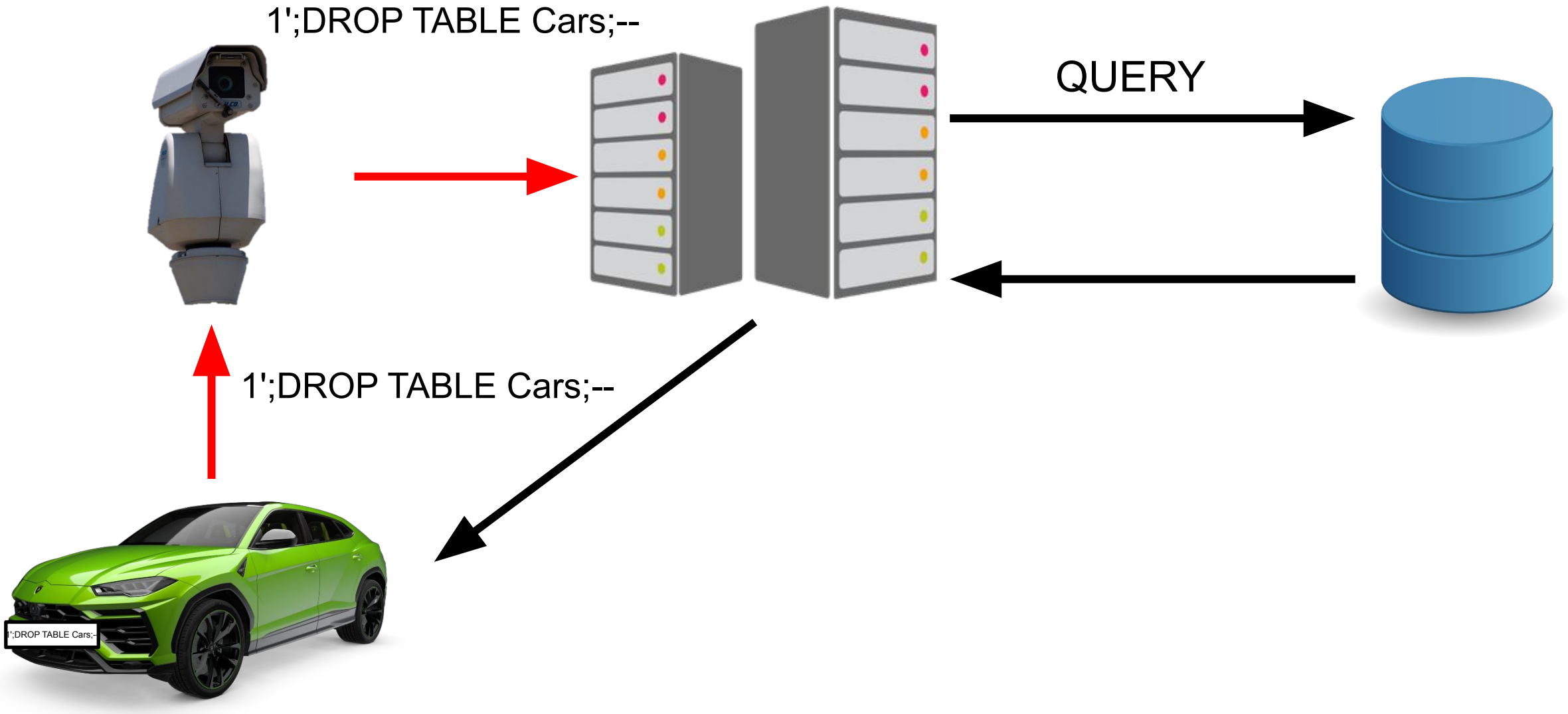
```
SELECT * FROM Cars WHERE PlateNumber = ' + PlateNumber + '
```

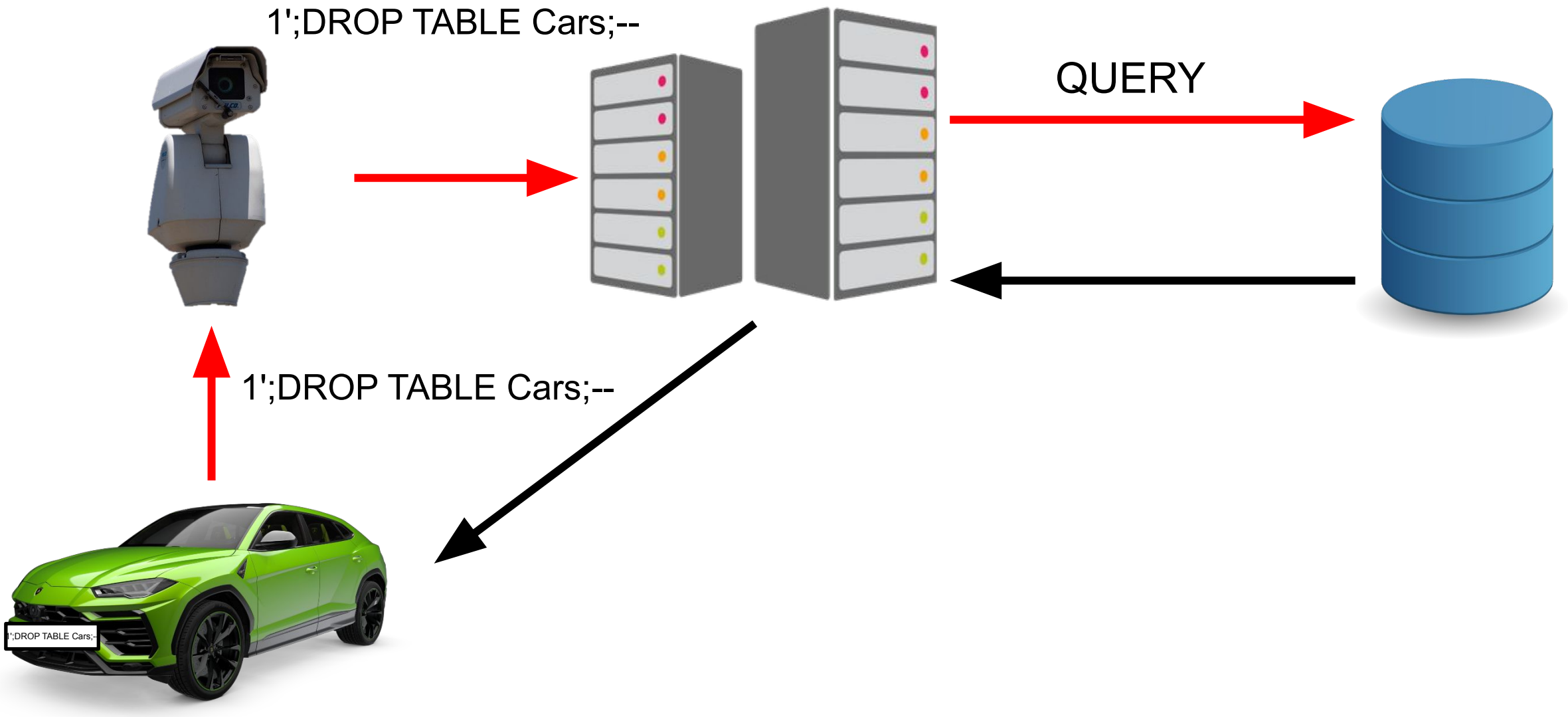

SQL injection

```
SELECT * FROM Cars WHERE PlateNumber = ' + PlateNumber + '

// x777am62
SELECT * FROM Cars WHERE PlateNumber = 'x777am62'
```



1';DROP TABLE Cars;--
QUERY
1';DROP TABLE Cars;--
1';DROP TABLE Cars;--


1';DROP TABLE Cars;--
QUERY
1';DROP TABLE Cars;--
1';DROP TABLE Cars;--


1';DROP TABLE Cars;--
QUERY
1';DROP TABLE Cars;--
1';DROP TABLE Cars;--


1';DROP TABLE Cars;--
QUERY
1';DROP TABLE Cars;--
1';DROP TABLE Cars;--

SQL injection

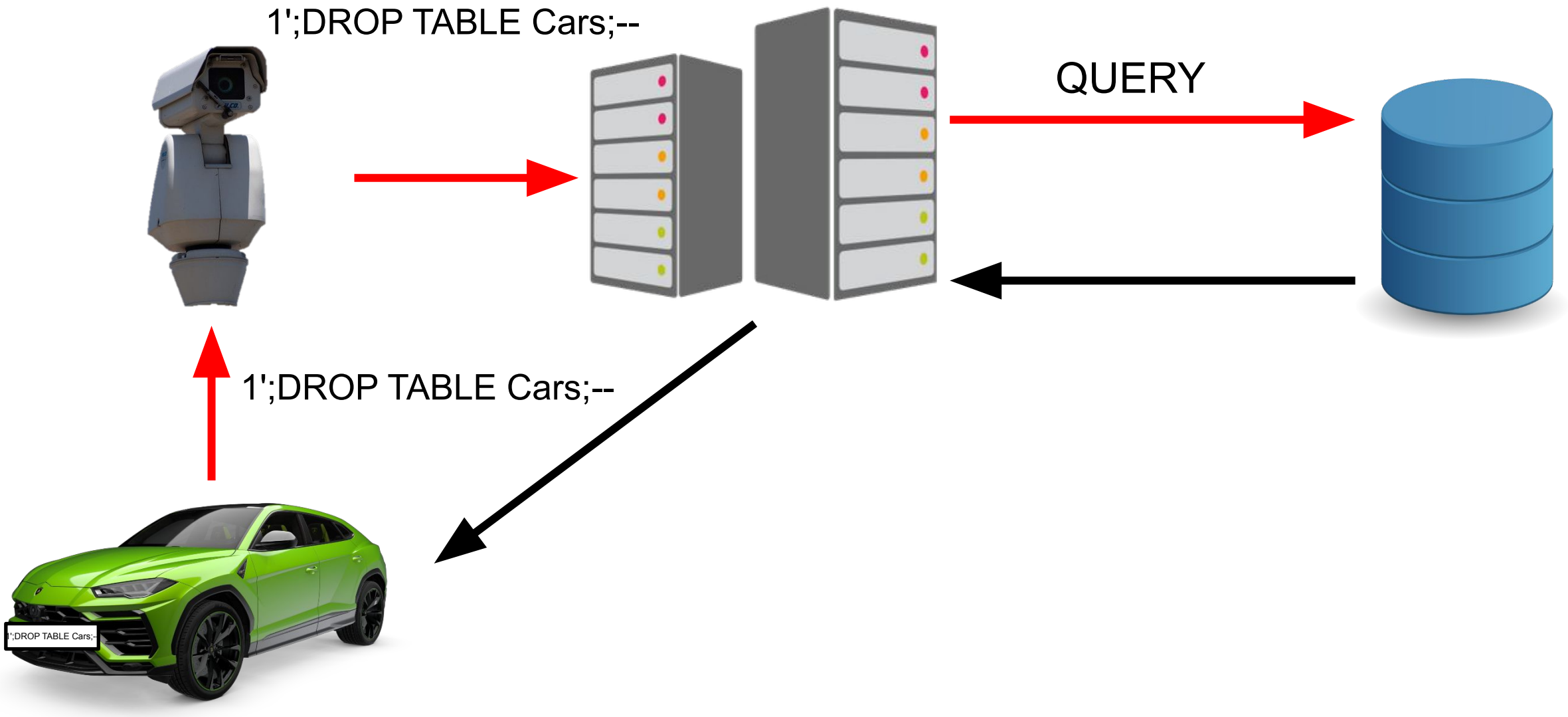
```
SELECT * FROM Cars WHERE PlateNumber = ' + PlateNumber + '

// x777am62
SELECT * FROM Cars WHERE PlateNumber = 'x777am62'
```

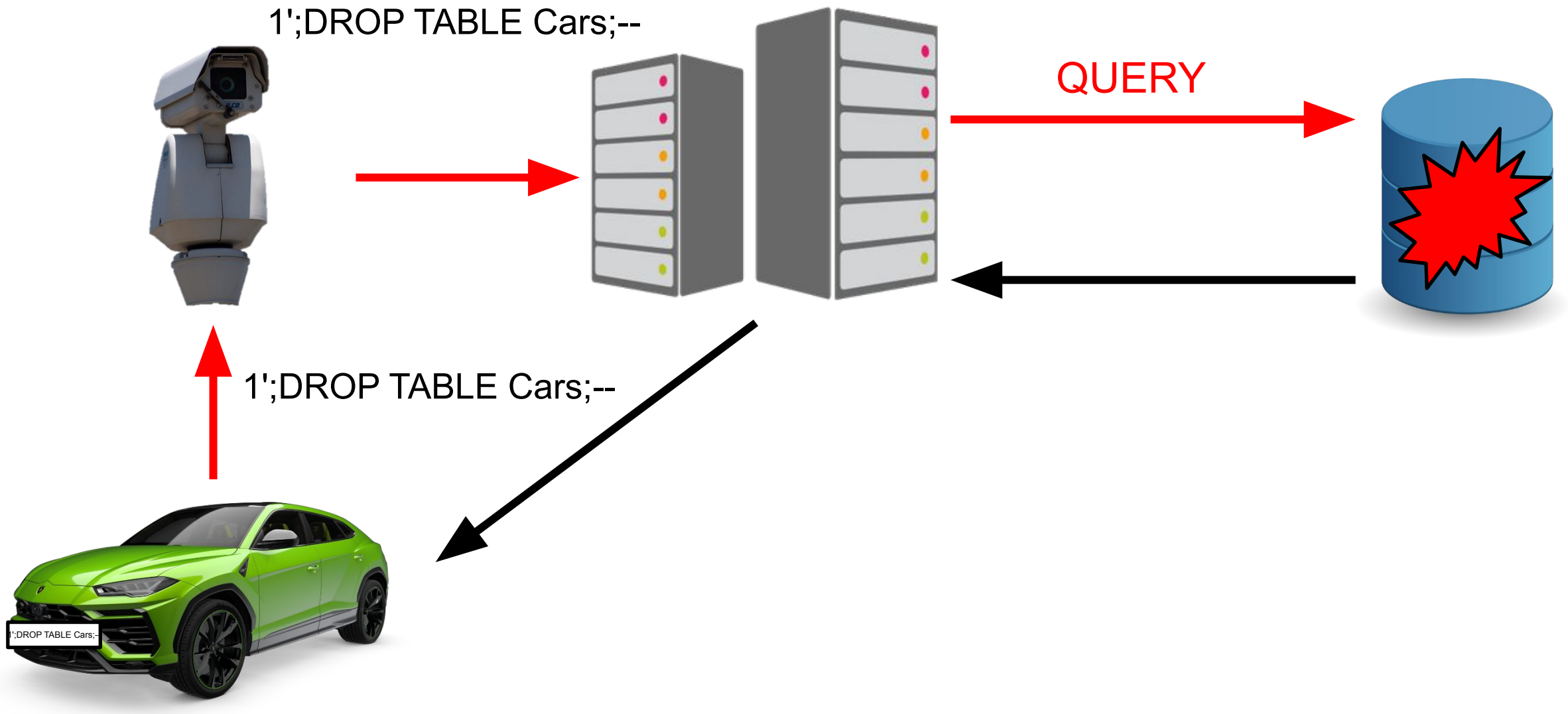

SQL injection

```
SELECT * FROM Cars WHERE PlateNumber = ' + PlateNumber + '

// x777am62
SELECT * FROM Cars WHERE PlateNumber = 'x777am62'

// 1';DROP TABLE Cars;--
SELECT * FROM Cars WHERE PlateNumber = '1';DROP TABLE Cars;--
```


SQL injection

```
SELECT * FROM Cars WHERE PlateNumber = ' + PlateNumber + '

// x777am62
SELECT * FROM Cars WHERE PlateNumber = 'x777am62'

// 1';DROP TABLE Cars;--
SELECT * FROM Cars WHERE PlateNumber = '1';DROP TABLE Cars;--
```



1';DROP TABLE Cars;--
QUERY
1';DROP TABLE Cars;--
1';DROP TABLE Cars;--


1';DROP TABLE Cars;--
QUERY
1';DROP TABLE Cars;--
1';DROP TABLE Cars;--

Хакер, когда кто-то проектирует по оптимистичному сценарию

SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```


SQL injection

```
using (SqlConnection connection = new SqlConnection(_connectionString)) {
  String userName = Request.Form["userName"];
  using (var command = new SqlCommand() {
    Connection = connection,
    CommandText = $"SELECT * FROM Users WHERE UserName = '" + userName + "'",
    CommandType = System.Data.CommandType.Text
  }) {
    using (var reader = command.ExecuteReader())
      // Data processing
  }
}
```

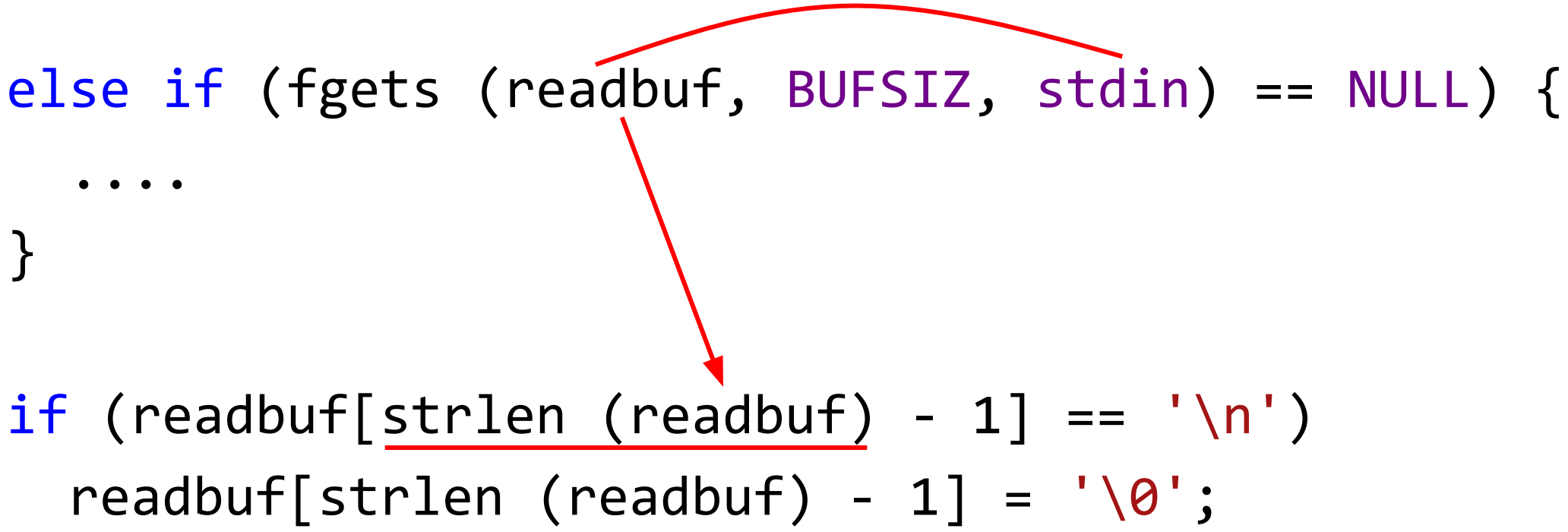

Taint analysis

- Не требует точных значений
- Ищет трассы “заражённых” данных
- Хорош для поиска инъекций

Taint analysis

Источники	Передатчики	Валидаторы	Приёмники
Откуда приходят?	Как передаются?	Как проверяются?	Куда не должны попасть?

libidn: CVE-2015-8948

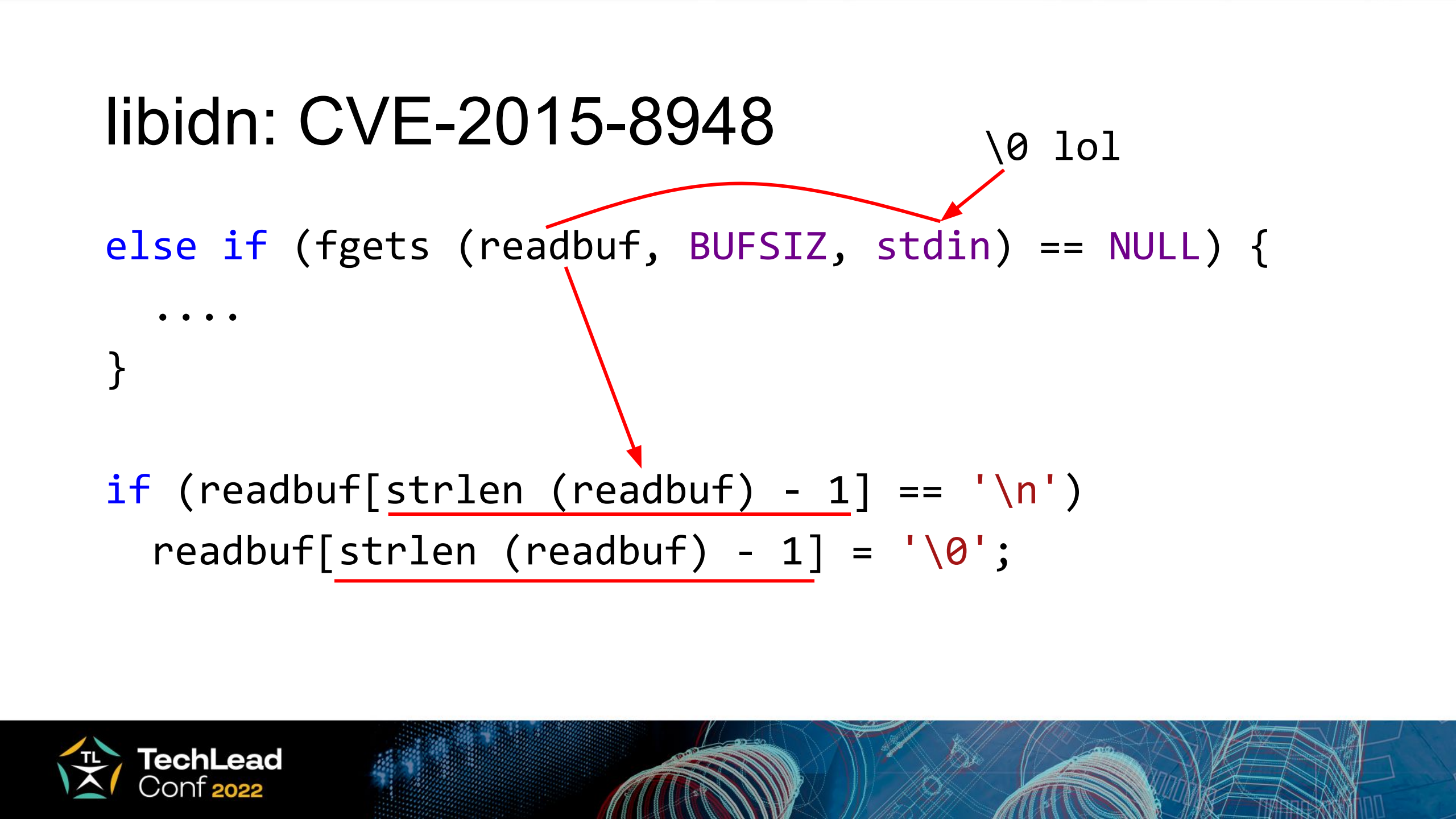


```
else if (fgets (readbuf, BUFSIZ, stdin) == NULL) {
   ....
}

if (readbuf[strlen (readbuf) - 1] == '\n')
  readbuf[strlen (readbuf) - 1] = '\0';
```

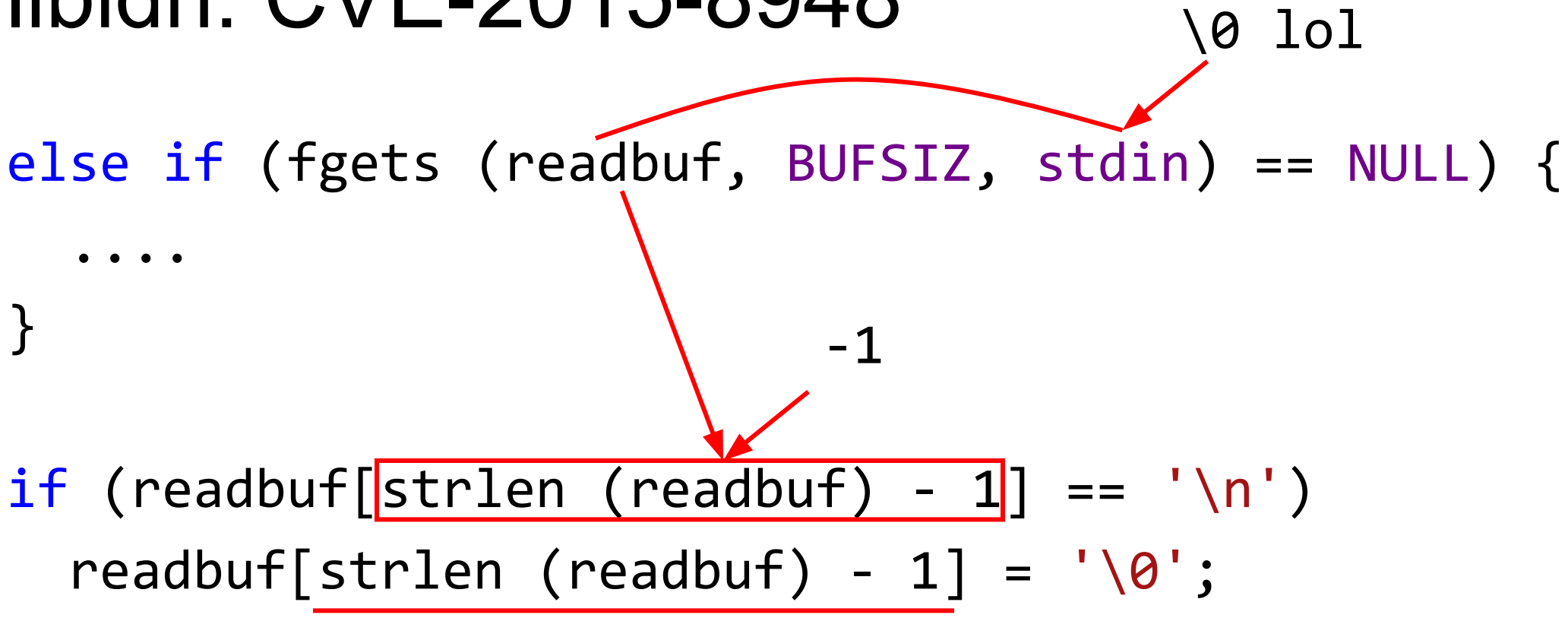
libidn: CVE-2015-8948

```
else if (fgets (readbuf, BUFSIZ, stdin) == NULL) {
   ....
}

if (readbuf[strlen (readbuf) - 1] == '\n')
  readbuf[strlen (readbuf) - 1] = '\0';
```

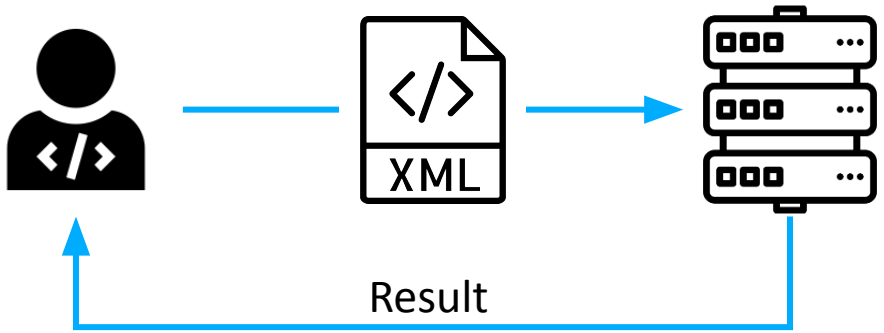
libidn: CVE-2015-8948

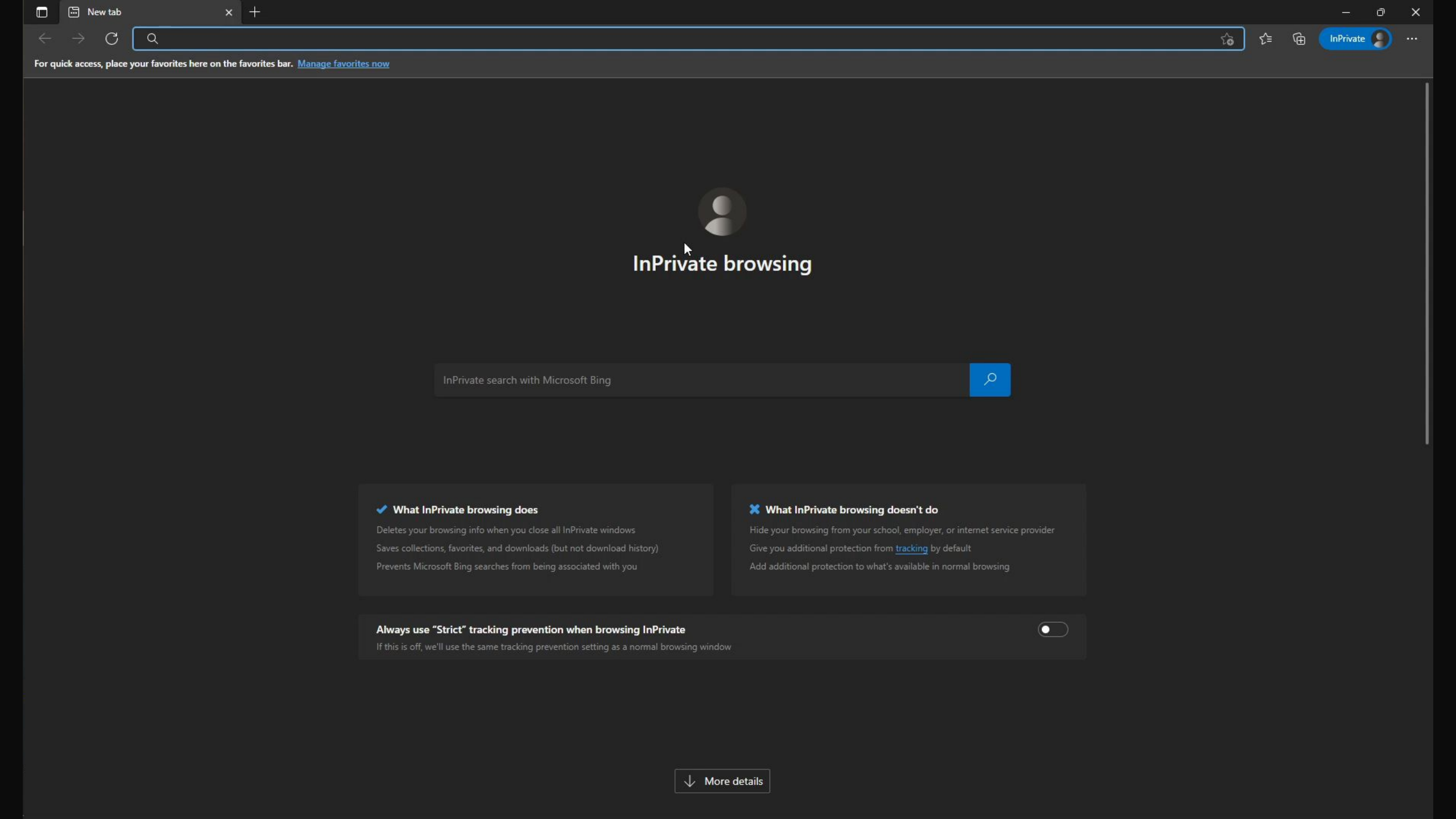


libidn: CVE-2015-8948

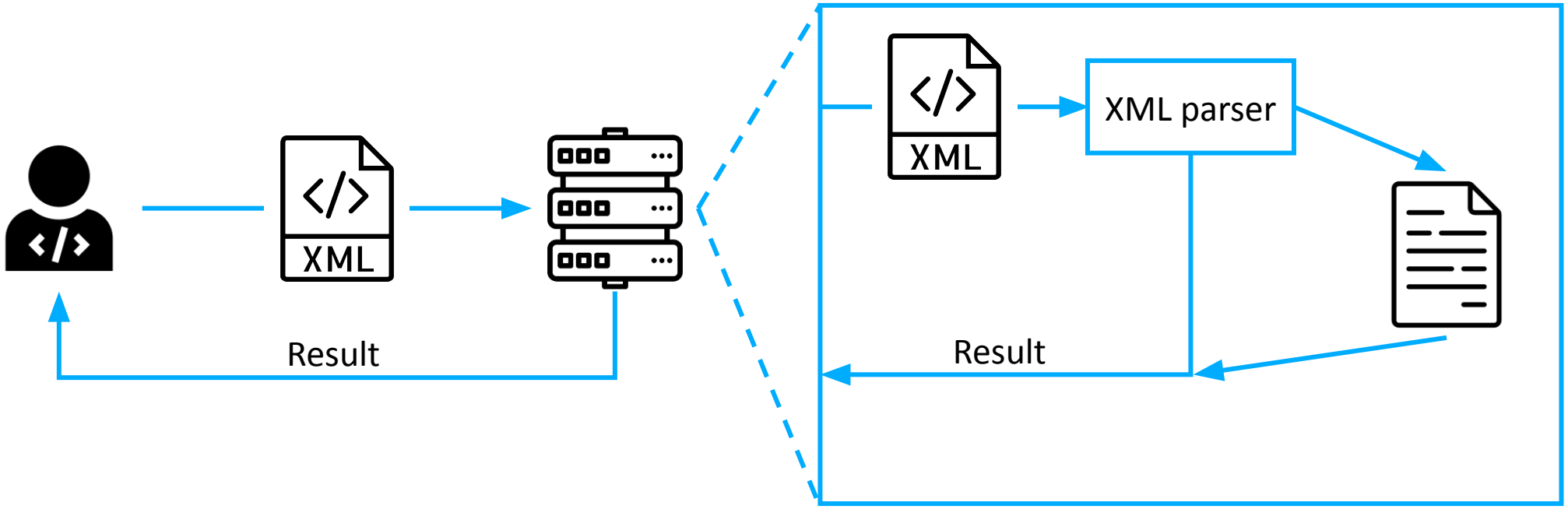


XXE (XML eXternal Entities)

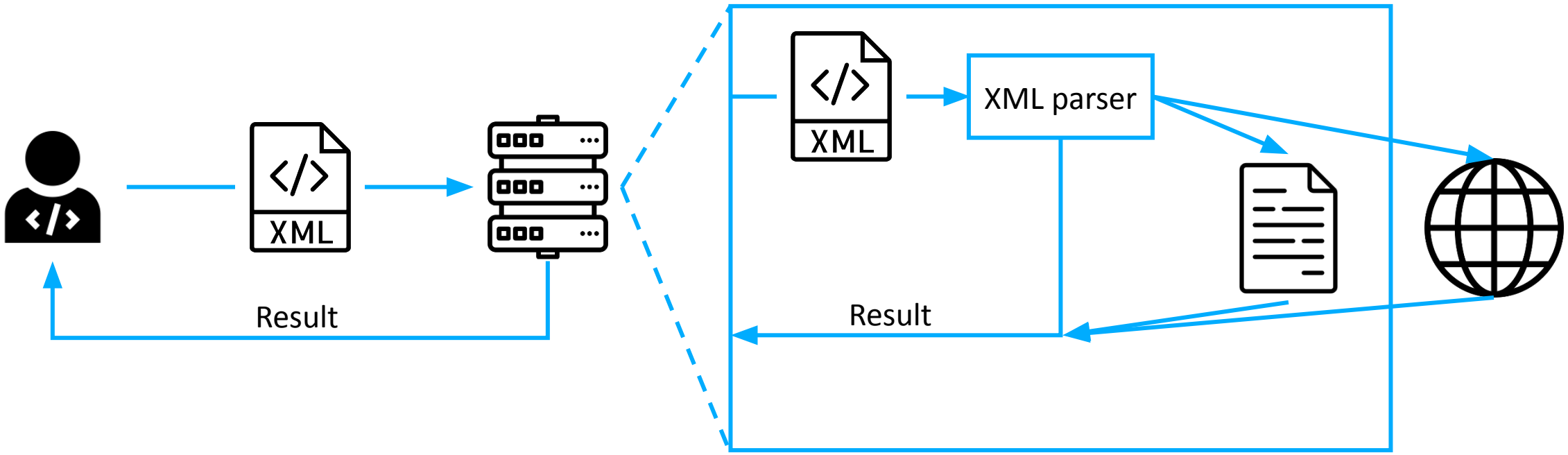



New tab
InPrivate
For quick access, place your favorites here on the favorites bar. Manage favorites now
InPrivate browsing
InPrivate search with Microsoft Bing
What InPrivate browsing does
Deletes your browsing info when you close all InPrivate windows
Saves collections, favorites, and downloads (but not download history)
Prevents Microsoft Bing searches from being associated with you
What InPrivate browsing doesn't do
Hide your browsing from your school, employer, or internet service provider
Give you additional protection from tracking by default
Add additional protection to what's available in normal browsing
Always use "Strict" tracking prevention when browsing InPrivate
If this is off, we'll use the same tracking prevention setting as a normal browsing window
More details

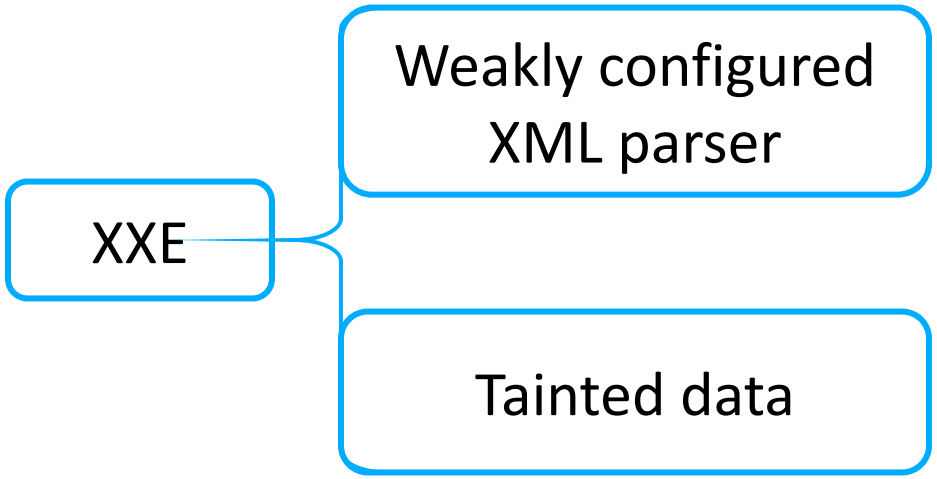
XXE (XML eXternal Entities)



XXE (XML eXternal Entities)



XXE



Способ #1
Ищем только опасный парсер

Способ #2
Ищем опасный парсер и данные от пользователя

```
private static string
ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request
                                .InputStream
                                .Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(
    buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```


```
private static string
ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request
                                .InputStream
                                .Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(
    buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```


BlogEngine.NET: CVE-2018-14485

```
private static string ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request.InputStream.Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```

```
private static string
ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request
                              .InputStream
                              .Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(
    buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```


BlogEngine.NET: CVE-2018-14485

```
public XMLRPCRequest(HttpContext input)
{
    var inputXml = ParseRequest(input);

    // LogMetaWeblogCall(inputXml);
    this.LoadXmlRequest(inputXml); // Loads Method Call
                                   // and Associated Variables
}
```


```
private static string
ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request
                              .InputStream
                              .Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(
    buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```


BlogEngine.NET: CVE-2018-14485

```
private void LoadXmlRequest(string xml) {
  var request = new XmlDocument();
  try {
    if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method"))) {
      xml = xml.Substring(xml.IndexOf("<?xml"));
    }
    request.LoadXml(xml);
  }
  ....
}
```


BlogEngine.NET: CVE-2018-14485

```
private void LoadXmlRequest(string xml) { [inputXml -> xml]
  var request = new XmlDocument();
  try {
    if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method"))) {
      xml = xml.Substring(xml.IndexOf("<?xml"));
    }
    request.LoadXml(xml);
  }
  ....
}
```


BlogEngine.NET: CVE-2018-14485

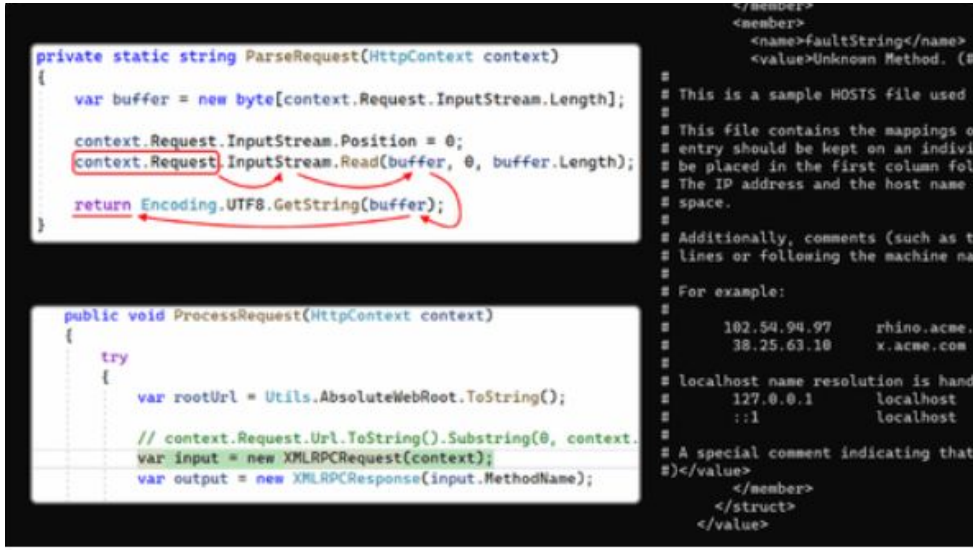
```
private void LoadXmlRequest(string xml) {
  var request = new XmlDocument();
  try {
    if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method"))) {
      xml = xml.Substring(xml.IndexOf("<?xml"));
    }
    request.LoadXml(xml);
  }
  ....
}
```


```
private static string
ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request
                                .InputStream
                                .Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(
    buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```


```
private static string
ParseRequest(HttpContext context)
{
  var buffer = new byte[context.Request
                                .InputStream
                                .Length];

  context.Request.InputStream.Position = 0;
  context.Request.InputStream.Read(
    buffer, 0, buffer.Length);

  return Encoding.UTF8.GetString(buffer);
}
```

BlogEngine.NET

SCAN ME



#CSharp #Knowledge #Security 11 Фев 2022

Уязвимости из-за обработки XML-файлов: XXE в C# приложениях в теории и на практике

Сергей Васильев

Как простая обработка XML-файлов может стать дефектом безопасности? Каким образом блог, развёрнутый на вашей машине, может стать причиной утечки данных? Сегодня мы ответим на эти вопросы и разберём, что такое XXE и как эта уязвимость выглядит в теории и на практике.

...

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Basis

Pattern-based analysis

Data-flow analysis

Taint checking

Итоги

TechLead Conf 2022

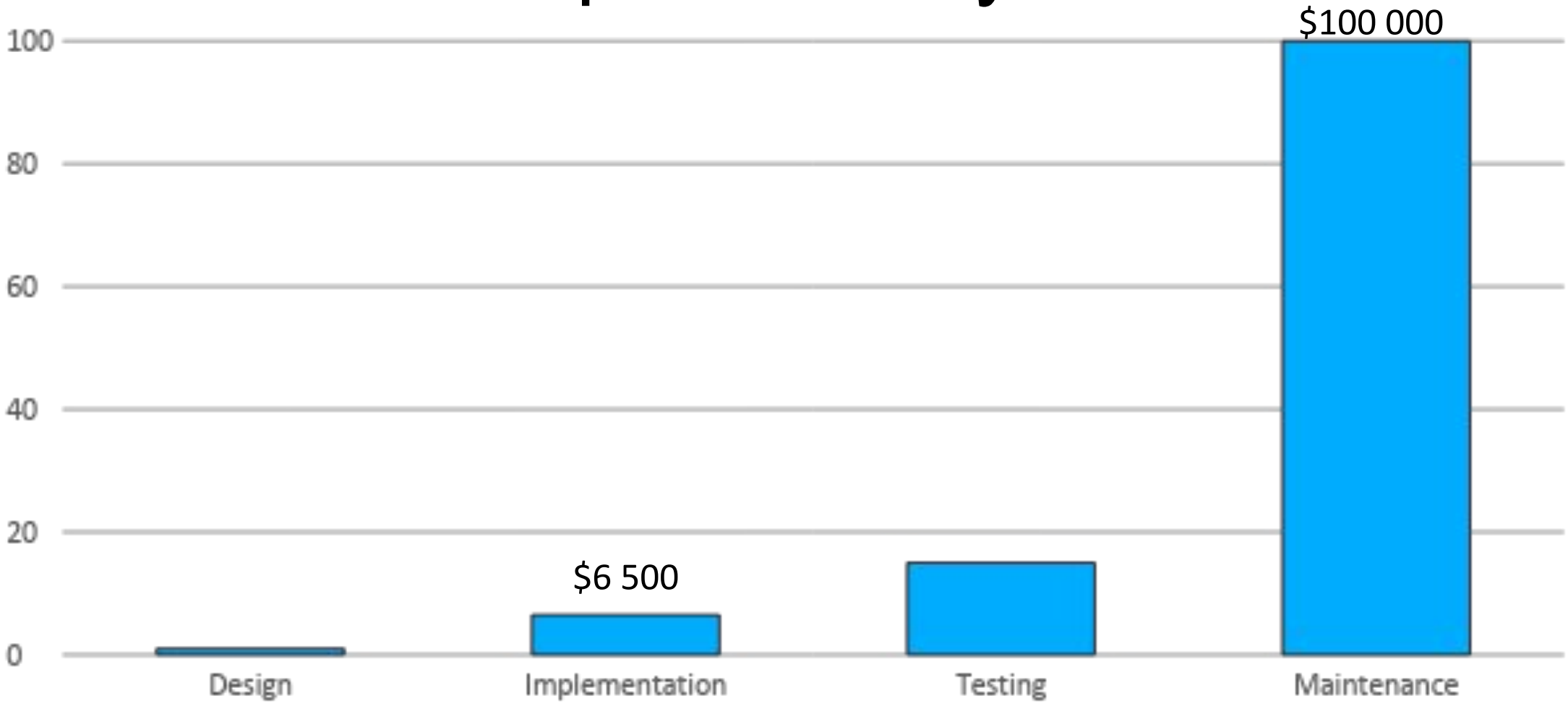
Basis

Pattern-based analysis

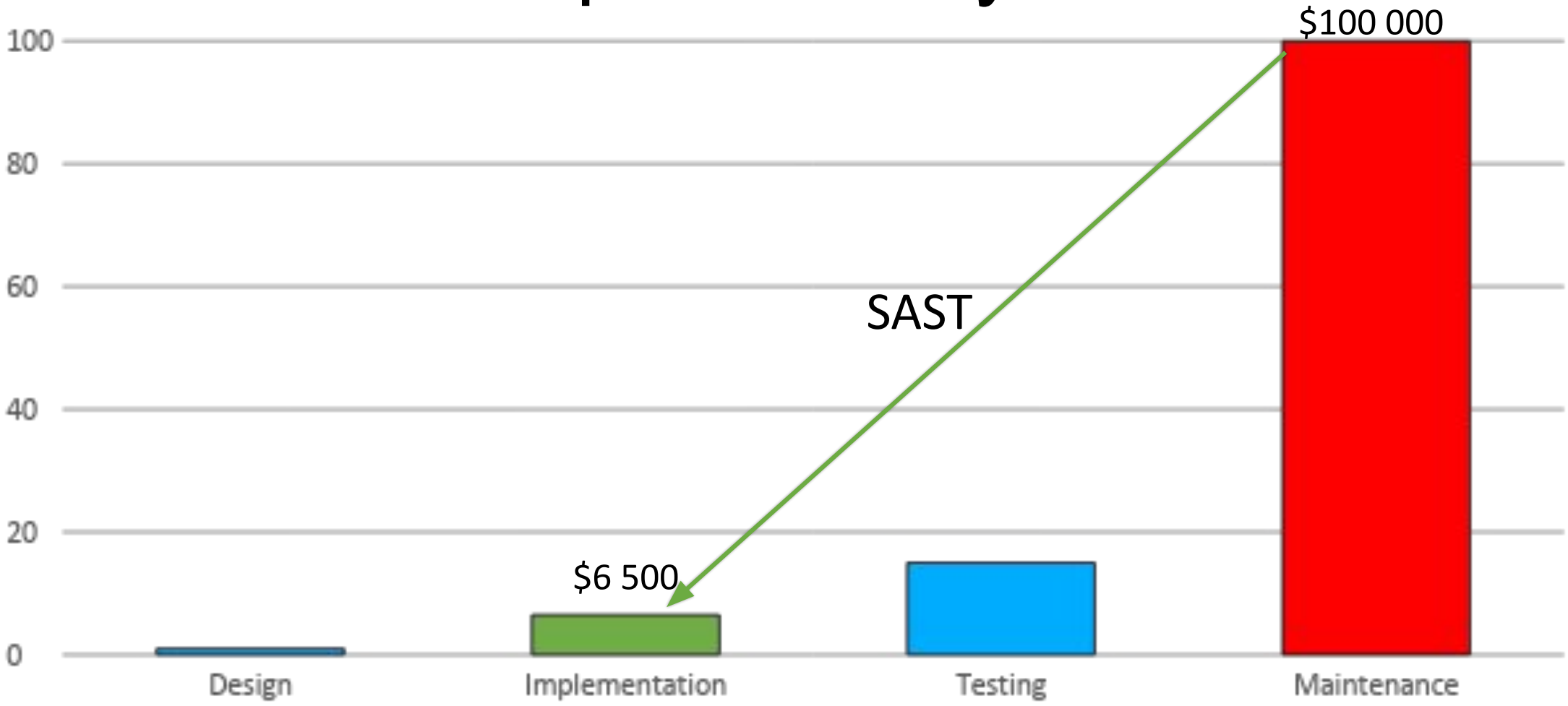
Data-flow analysis

Taint checking

Стоимость исправления уязвимости



Стоимость исправления уязвимости



Знак, что пора внедрять SAST :)

Сергей Васильев
PVS-Studio LLC

OC

TechLead Conf 2022