KCon

TABBY:
Java Code Review like a pro

王柏柱（wh1t3p1g）

+1 进阶，
护航未来

KCon

+1

CONTENTS

目录

KCon

+1

Background

KCon

代码审计发展回顾

+1

痛点：

1. 漏报率高，审计不全面；
2. 工具输出结果繁多，误报率高

人工审计阶段

依靠专家经验人工审计
辅以正则匹配工具

标志性工具：
Seay代码审计工具[1]

痛点：

1. 分析成本高，中间的分析结果不可重用
2. 可定制化能力差

初期半自动化阶段

过程内分析
AST流分析/token流分析/
简单数据流分析

标志性工具：
rips[2]，cobra[3]

后期半自动化阶段

过程间分析
跨函数的污点数据流分析。

标志性工具：
GadgetInspector[4]、
fortify[5]

现阶段最优解

代码数据化阶段

代码数据化
依靠程序分析，生成代码属性图。

标志性工具：CodeQL[6]

[1] https://github.com/f1tz/cnseay
[2] https://github.com/ripsscanner/rips
[3] https://github.com/FeeiCN/Cobra
[4] https://github.com/JackOfMostTrades/gadgetinspector
[5] https://www.microfocus.com/en-us/cyberres/application-security/static-code-analyzer
[6] https://github.com/github/codeql

KCon

为什么要有tabby？

面向的场景：

CodeQL面向的是甲方场景，可以直接更根据源码进行分析

Tabby面向的是安全研究人员，可以对编译后的项目进行分析

支持的漏洞类型：

CodeQL很难支持Java反序列化利用链的挖掘

Tabby可以对项目、三方组件、jdk组件进行利用链的挖掘

时间：

Tabby方案实现时间在2020年左右，CodeQL当时只提供了线上体验，

又没有好用的同类工具，那就自己造个轮子！

定位：Java安全研究人员代码审计的“辅助”工具

1. 定位图空间中的对象、函数
2. 聚焦可能存在问题的漏洞链路
3. 枚举类似路径的漏洞

tabby 构架

目标提取器
Target Extractor

War、Jar、Jsp、Class

语义提取器
Semantic Extractor

内存数据

1. 类空间
2. 函数空间
3. 污点分析中间数据

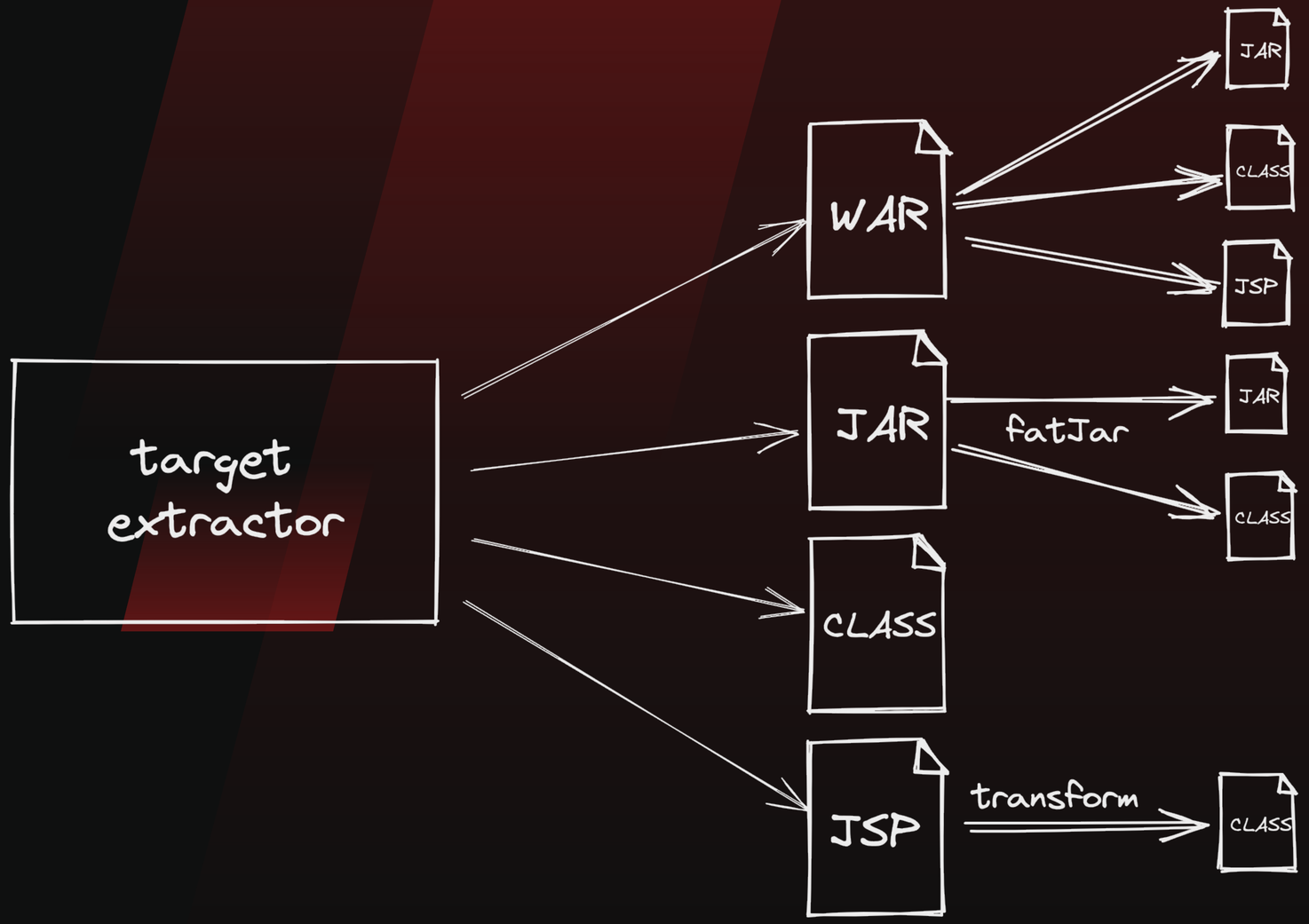
污点分析引擎
Taint Analysis Engine

代码属性图构建器
CPG Constructor

面向Java语言的代码属性图

贯穿构建流程

目标提取器 Target Extractor



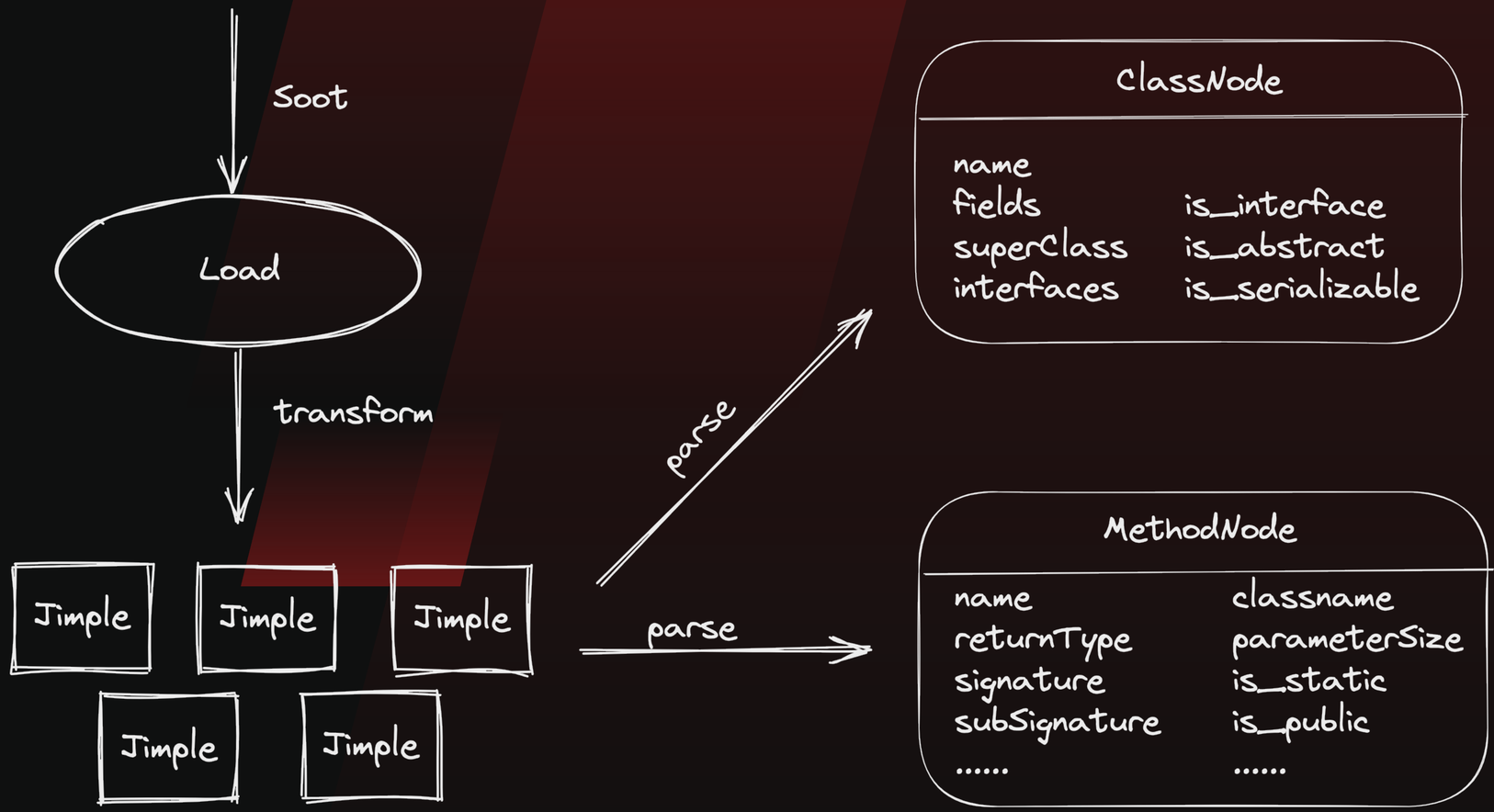
目标提取器

针对不同情况的目标文件，完整抽取所有待分析的对象：

1. Jsp文件采用tomcat-jasper动态编译
2. War文件采用解压缩的方式抽取
3. fatJar文件采用解压缩方式抽取

另外，jdk依赖可额外添加到分析目标中

语义提取器 Semantic Extractor



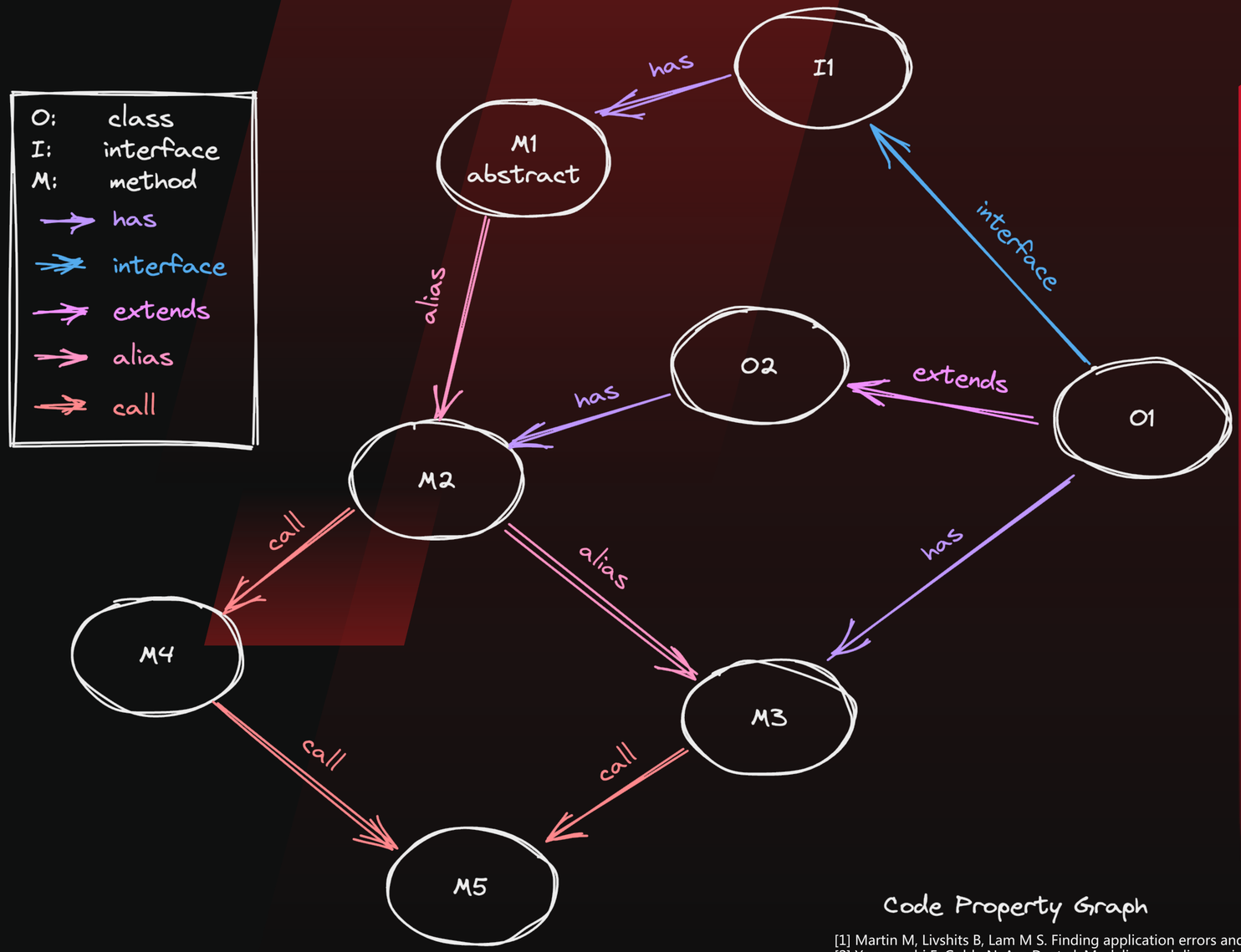
语义提取器

在进行语义分析前，语义提取器将待分析目标的语义信息抽取成语义空间：

1. 类空间：包含全量对象语义节点
2. 函数空间：包含全量函数语义节点

至此，我们获得了包含全量节点的语义空间，但每个节点之间仍是孤立状态

代码属性图构建器 CPG Constructor



Code Property Graph

代码属性图构建器

代码属性图构建器主要用于连接语义空间中各个孤立的节点，将其转化为一张具备分析能力的语义图。

面向Java语言的代码属性图共包含：

1. 类关系图 ORG
2. 函数别名图 MAG
3. 函数调用图 MCG
4. 精确的函数调用图 PCG（可选）

存在实体节点：

1. Class 节点
2. Method 节点

存在5种实体边：

1. Has边
2. Interface边
3. Extends边
4. Alias边
5. Call边

[1] Martin M, Livshits B, Lam M S. Finding application errors and security flaws using PQL: a program query language[J]. Acm Sigplan Notices, 2005, 40(10): 365-383.
[2] Yamaguchi F, Golde N, Arp D, et al. Modeling and discovering vulnerabilities with code property graphs[C]//2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 590-604.
[3] Backes M, Rieck K, Skoruppa M, et al. Efficient and flexible discovery of php application vulnerabilities[C]//2017 IEEE european symposium on security and privacy (EuroS&P). IEEE, 2017: 334

代码属性图构建器 | 类关系图 ORG

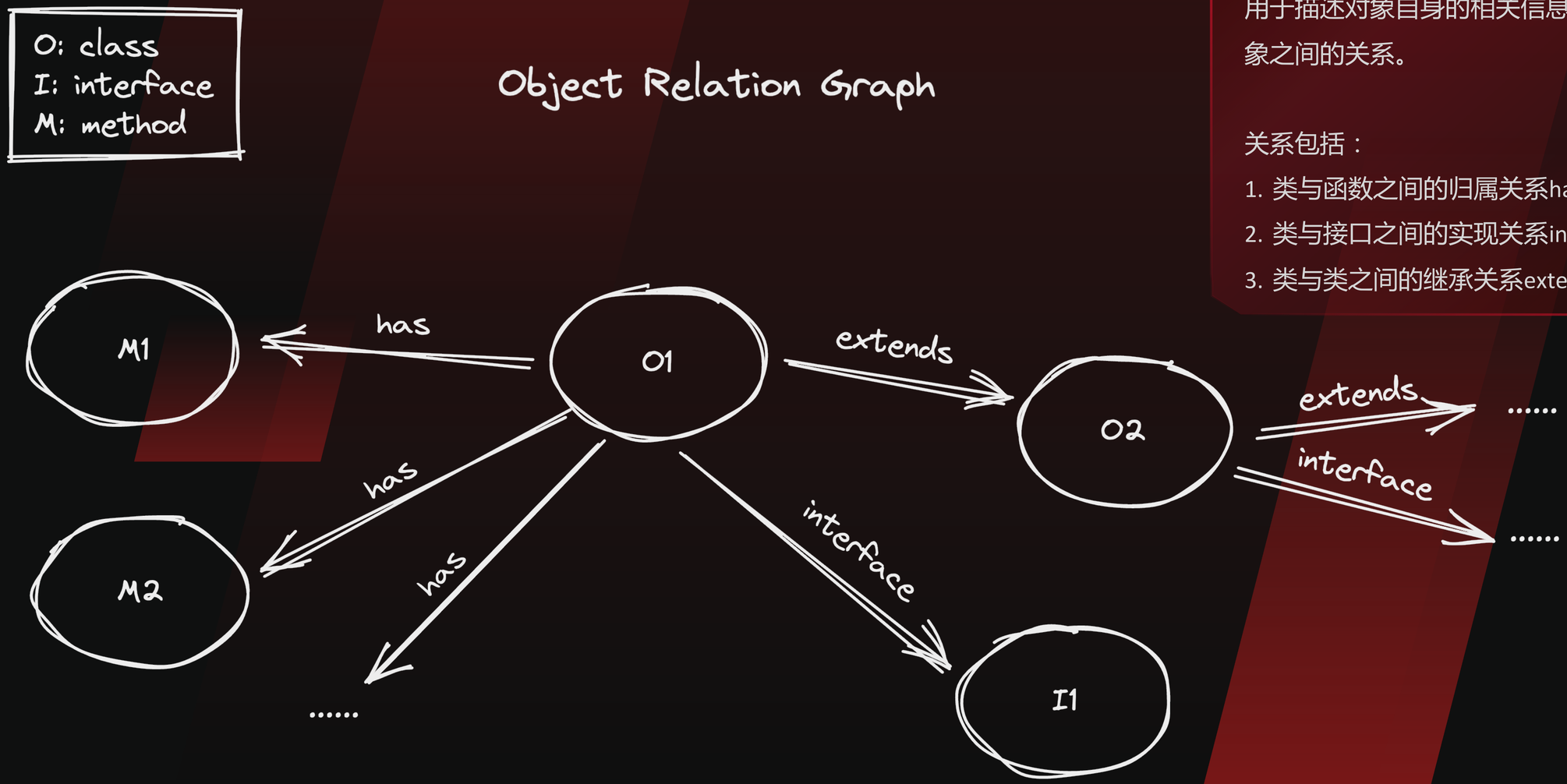
类关系图

Object Relation Graph

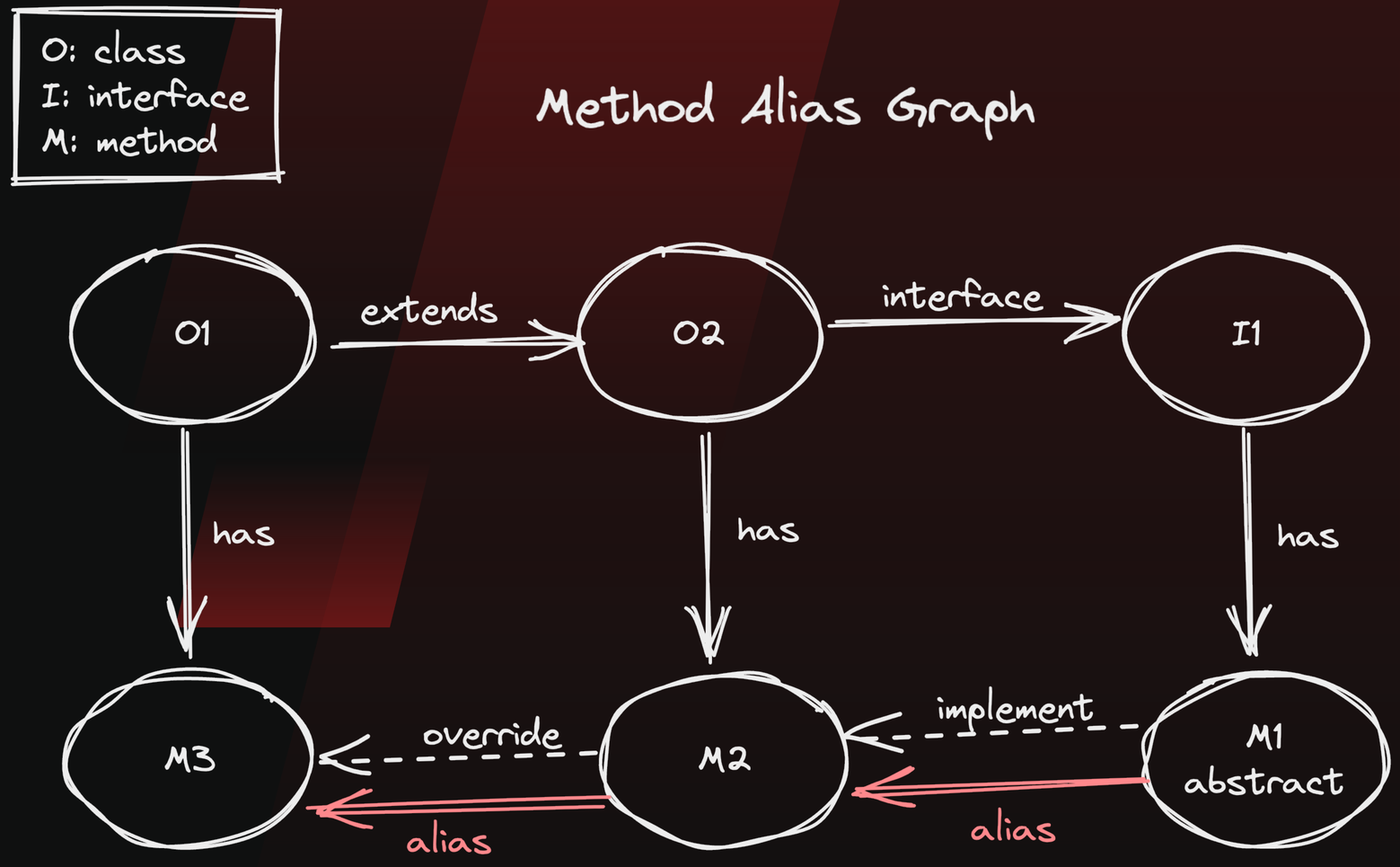
用于描述对象自身的相关信息以及同其他对象之间的关系。

关系包括：

1. 类与函数之间的归属关系has
2. 类与接口之间的实现关系interface
3. 类与类之间的继承关系extends



代码属性图构建器 | 函数别名图 MAG



函数别名图

Method Alias Graph

描述某一函数所有具体实现的语义图。函数别名图主要用于Java语言多态特性的分析场景。多态的特性导致了调用过程的断裂，而MAG做的就是修补断裂点，使其能完整分析所有链路。

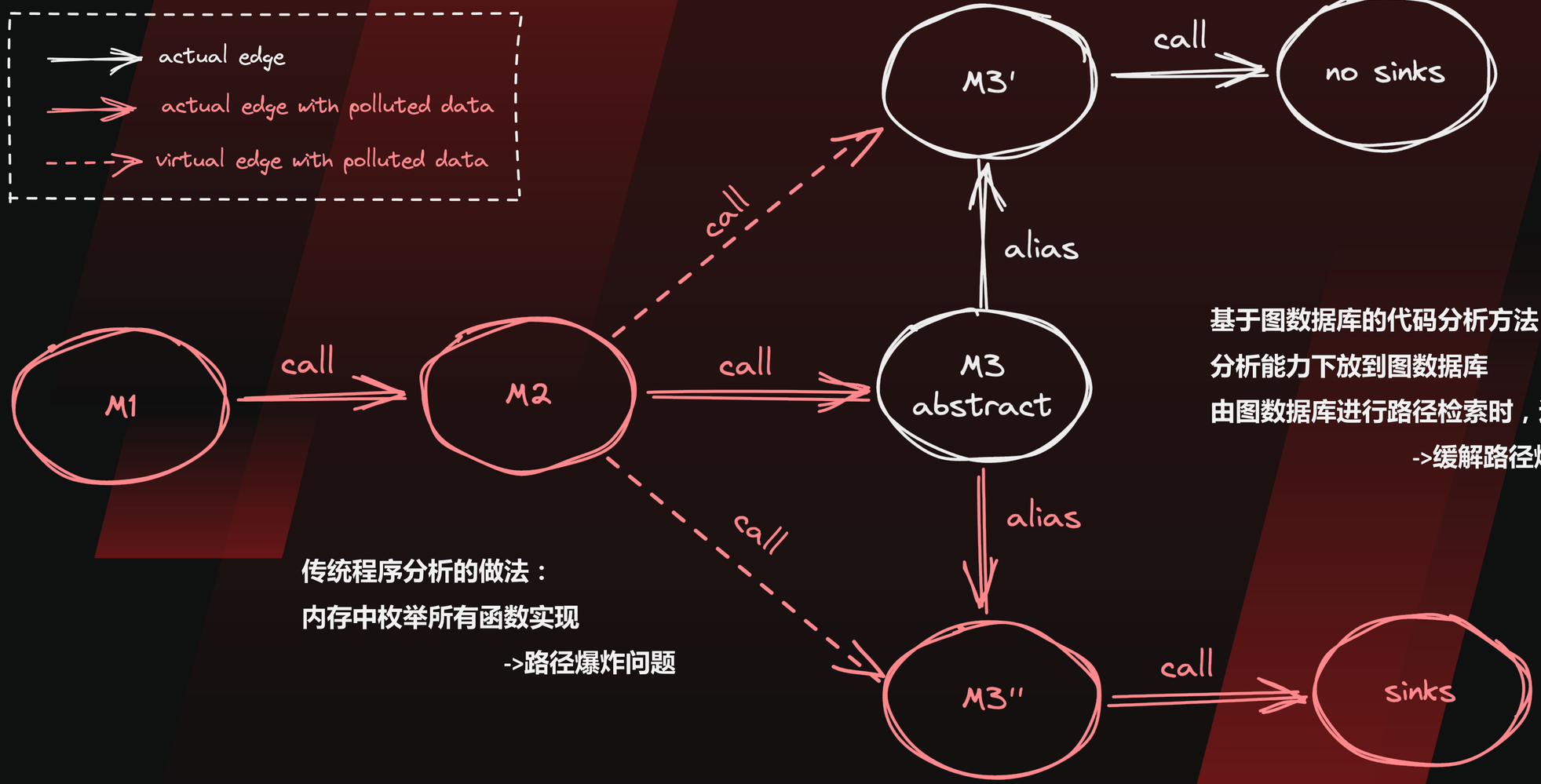
关系包括：

1. 函数与函数之间的别名关系alias

Alias边的集合是一个树状结构，树顶为interface或顶层类型；树枝为当前树顶函数的具体实现。

KCon

代码属性图构建器 | 函数别名图 MAG



传统程序分析的做法：

内存中枚举所有函数实现

->路径爆炸问题

基于图数据库的代码分析方法：

分析能力下放到图数据库

由图数据库进行路径检索时，边剪枝边枚举所有实现

->缓解路径爆炸问题

代码属性图构建器 | 函数调用图MCG

函数调用图

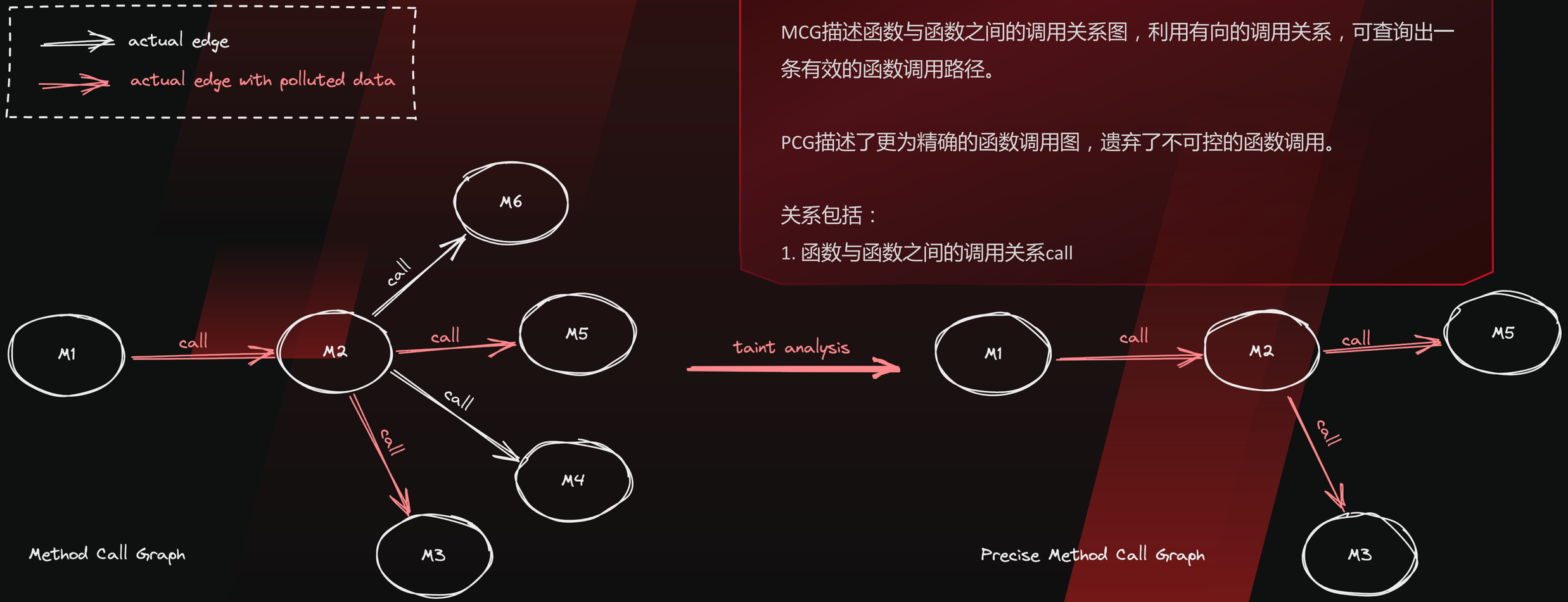
Method Call Graph

MCG描述函数与函数之间的调用关系图，利用有向的调用关系，可查询出一条有效的函数调用路径。

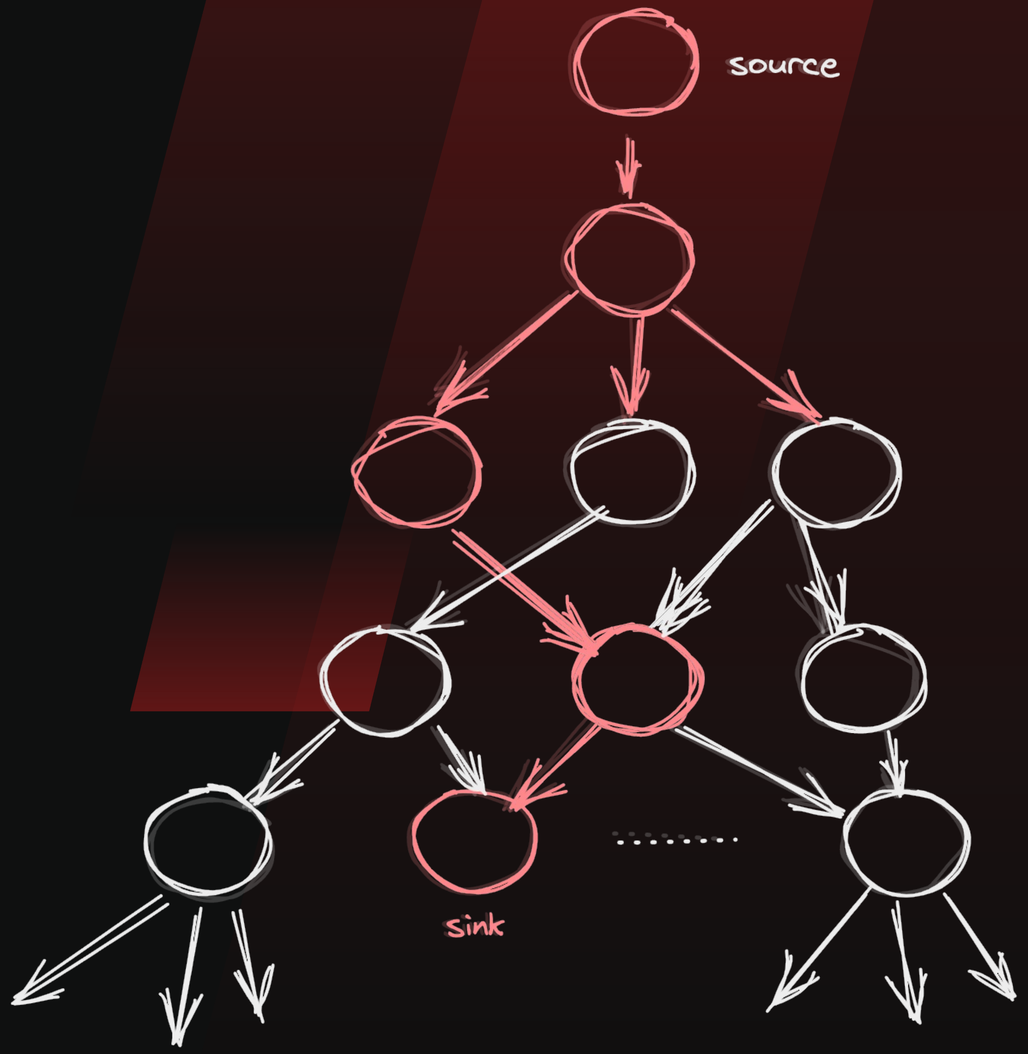
PCG描述了更为精确的函数调用图，遗弃了不可控的函数调用。

关系包括：

1. 函数与函数之间的调用关系call



污点分析引擎 Taint Analysis Engine



重复计算问题

路径爆炸问题

污点分析引擎

Taint Analysis Engine

污点分析引擎是tabby实现最核心的部分
它实现了从“代码属性图”至“带语义信息的代码属性图”的跨越，使得图数据库具备程序分析的基础。

Tabby重新设计了适合图数据库的污点分析算法：

化整为零

1. 单函数过程内分析算法，生成相应污点信息
2. 跨函数过程间分析算法，应用污点信息避免重复计算

化零为整

1. 依据图数据库路径检索能力，重建调用链路
2. 利用调用边污点信息，边剪枝边构建链路，缓解路径爆炸问题

污点分析引擎 | 过程内分析案例

类型	语句stmt	规则
简单赋值	a = b;	b变量的可控性传递给a变量
新建变量	a = new statement	a变量原有的可控性消除，并生成新的实体值
类属性赋值	a.f = b;	b变量的可控性传递给a变量的f属性
类属性载入	a = b.f;	b变量的f属性的可控性传递给a变量，创建f属性的实体值
静态类属性赋值	Class.field = b;	b变量的可控性传递给静态变量Class.field
静态类属性载入	a = Class.field;	静态变量Class.field的可控性传递给a变量，创建新的实体值
数组赋值	a[i] = b;	b变量的可控性传递给a变量的第i个元素
数组载入	a = b[i];	b变量的第i个元素的可控性传递给a变量
强制转化	a = (T) b;	同简单赋值一样
赋值函数调用	a = b.func(c);	b.func函数返回值的可控性传递给a变量，通常与b变量和入参c变量的可控性有关。
函数调用	b.func(c)	返回值无关的函数调用，func函数内容决定入参c和b变量本身的可控性。
函数返回	return stmt	当前函数的返回值，其可控性依赖于stmt所返回变量的可控性。

```
A example( A arg0 )
{
    A a = arg0;
    arg0 = new A();
    arg0.f = a;
    return arg0;
}
```

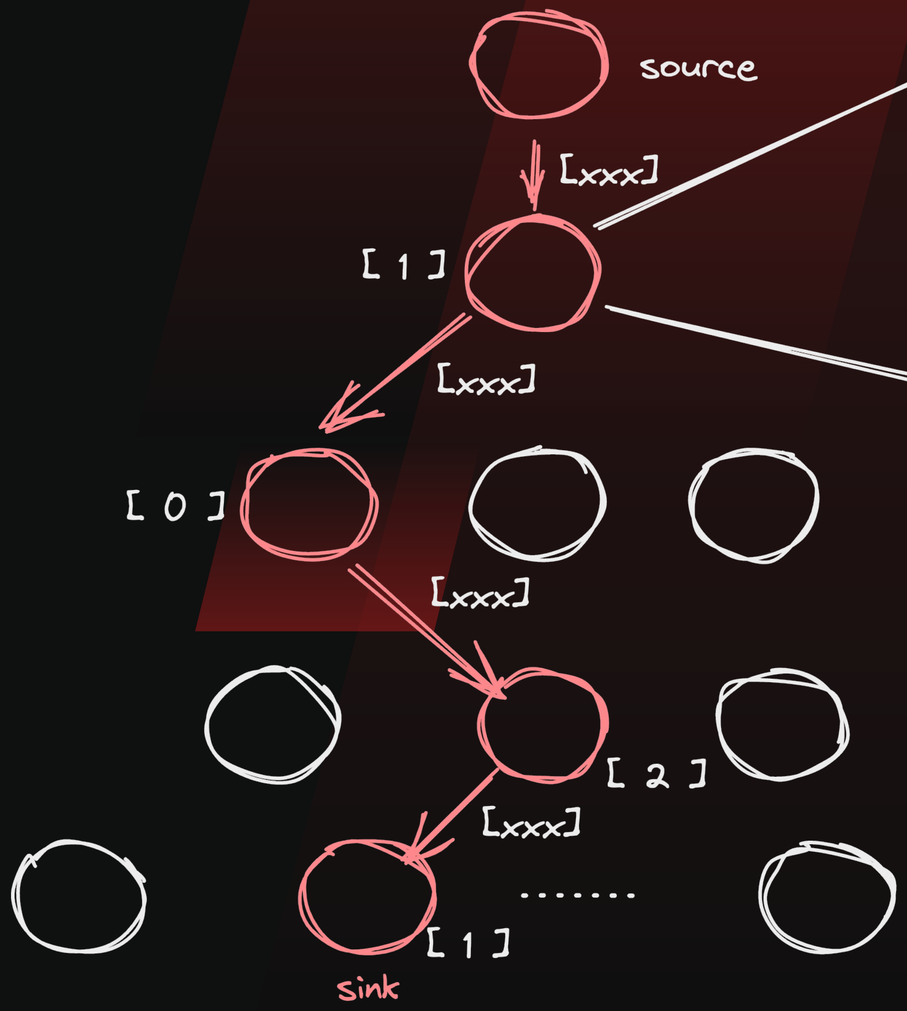
objects	values
arg0	v0
a	v0
arg0.f	v0
return	v1, v0

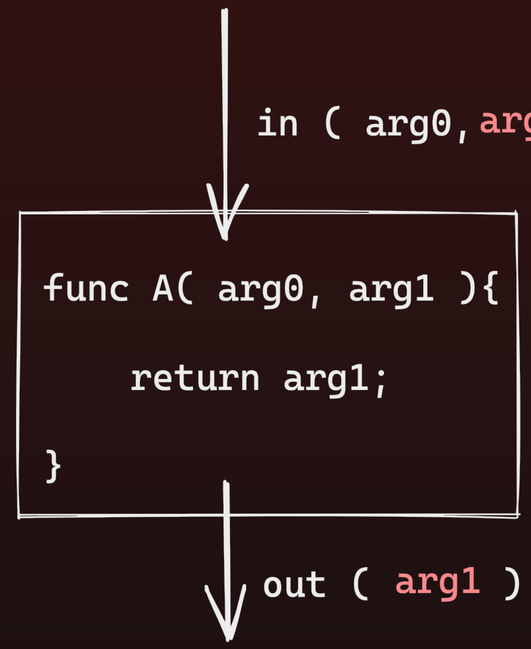
KCon

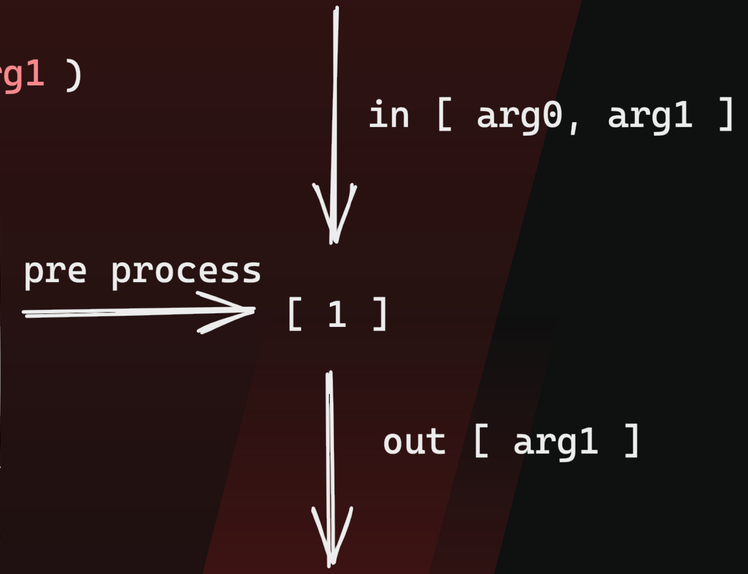
污点分析引擎 | 化整为零

+1

gadgetinspector







跨函数的过程间分析转化为

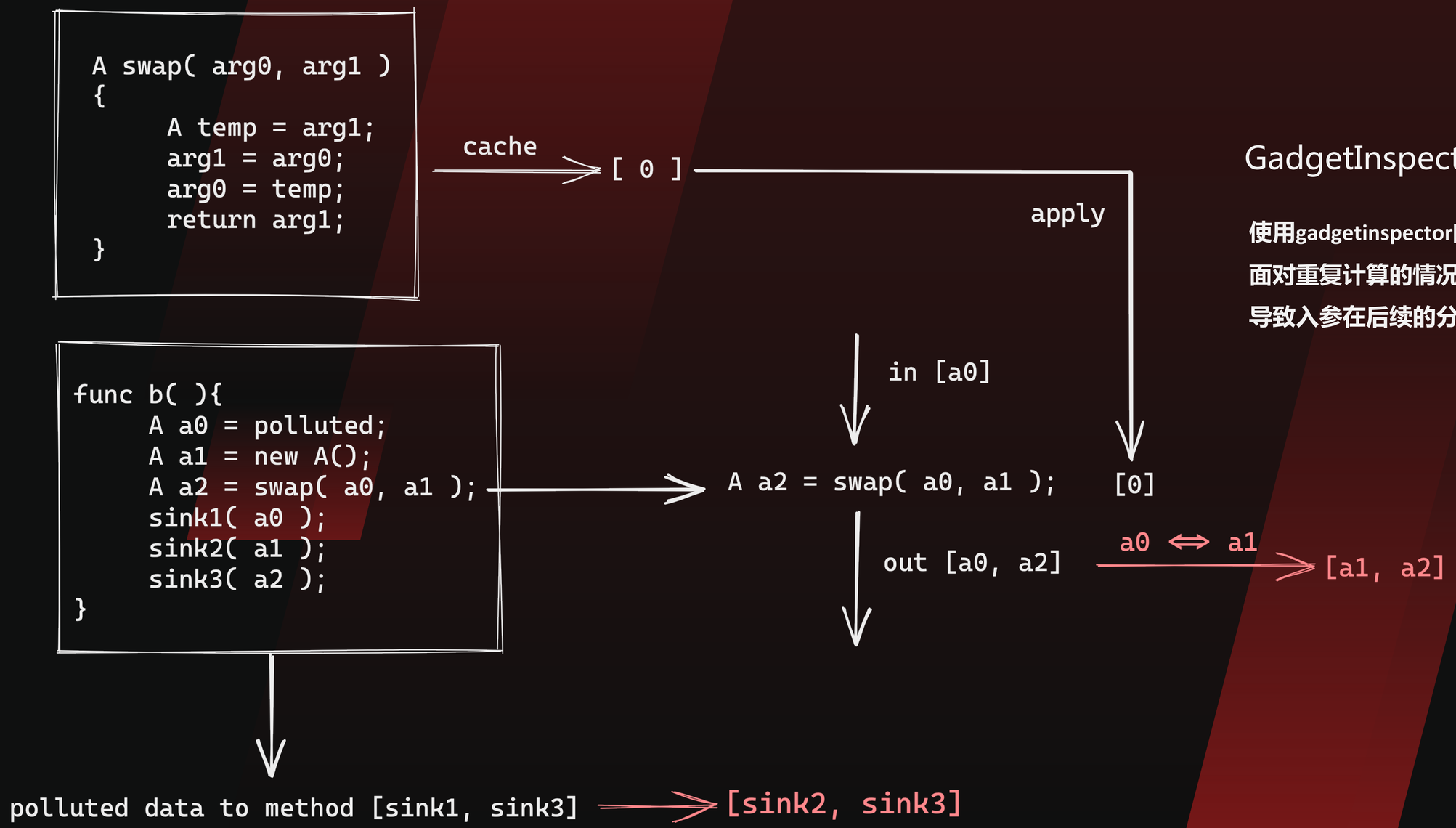
“调用边上污点信息” 同 “调用函数的语义缓存” 比较

利用逆拓扑排序算法，巧妙地将过程间分析“转化”为类过程内分析

同时也解决了重复计算分析的问题

分而治之的方式，也在一定程度上缓解了程序分析过程中路径爆炸的问题

污点分析引擎 | 化整为零



GadgetInspector分析案例

使用gadgetinspector的方案时

面对重复计算的情况，忽略了入参所发生的变化

导致入参在后续的分析中将产生 误报 或 漏报

KCon

污点分析引擎 | 化整为零

```
A swap( arg0, arg1 )
{
    A temp = arg1;
    arg1 = arg0;
    arg0 = temp;
    return arg1;
}
```

cache →

```
actions {
    "param-0"   : ["param-1"],
    "param-1"   : ["param-0"],
    "return"    : ["param-0"]
}
```

tabby semantic caching system

```
func b( ){
    A a0 = polluted;
    A a1 = new A();
    A a2 = swap( a0, a1 );
    sink1( a0 );
    sink2( a1 );
    sink3( a2 );
}
```

apply ~~gadgetinspector~~ polluted data tabby translate

```
init
copy
"param-0": ["param-1"]
"param-1": ["param-0"]
"return" : ["param-0"]
clean temp data
```

```
in [ a0 ]
[a0, a0']
[a0']
[a1, a0']
}a1, a2, a0']
out [a1, a2]
```

```
A a2 = swap( a0, a1 );
```

```
out [ a1, a2 ]
```

```
a0' = a0; a1' = a1;
a0 = a1';
a1 = a0';
a2 = a0';
a0' = null; a1' = null;
```

```
"return": "param-0"
"return<f>data": "param-1"
1. apply actions
2. cache call edge
    b --[0,-3]-> swap
```

```
assert a0 is polluted; // false
assert a1 is polluted; // true
assert a2 is polluted; // true
```

polluted data to method sink2, sink3

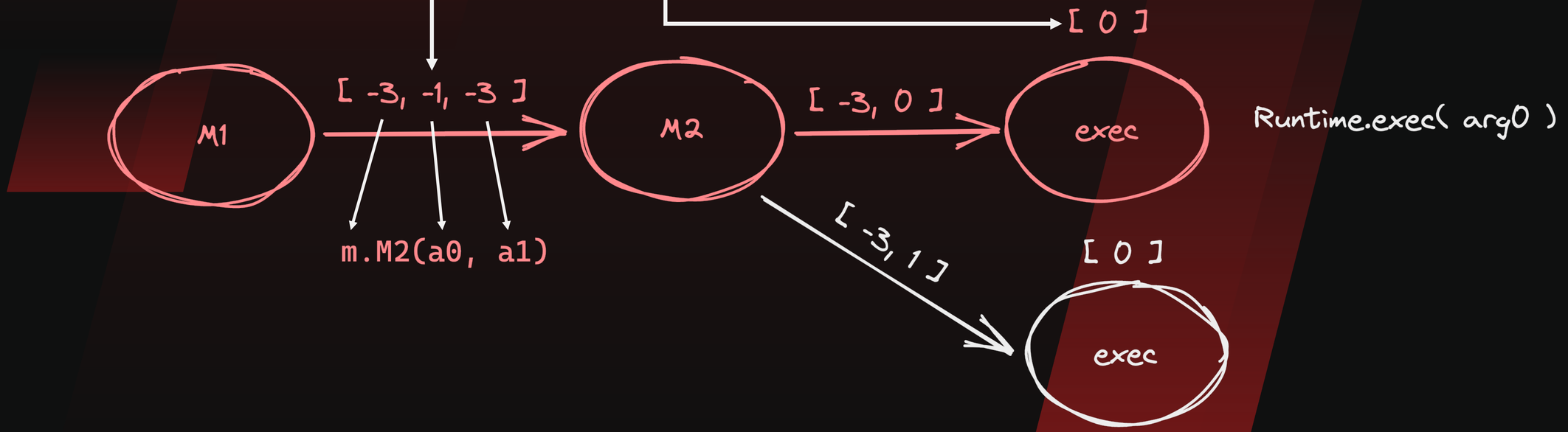
污点分析引擎 | 化零为整

污点推算规则

Tabby的污点推算规则共依靠两块语义信息：

1. 调用边上的污点信息
2. 预置sink函数的污点信息

可控性标识	意义
-3	表示变量不可控
-2	表示变量来源于sources
-1	表示变量来源于调用者本身
0-n	表示变量来源于函数参数列表

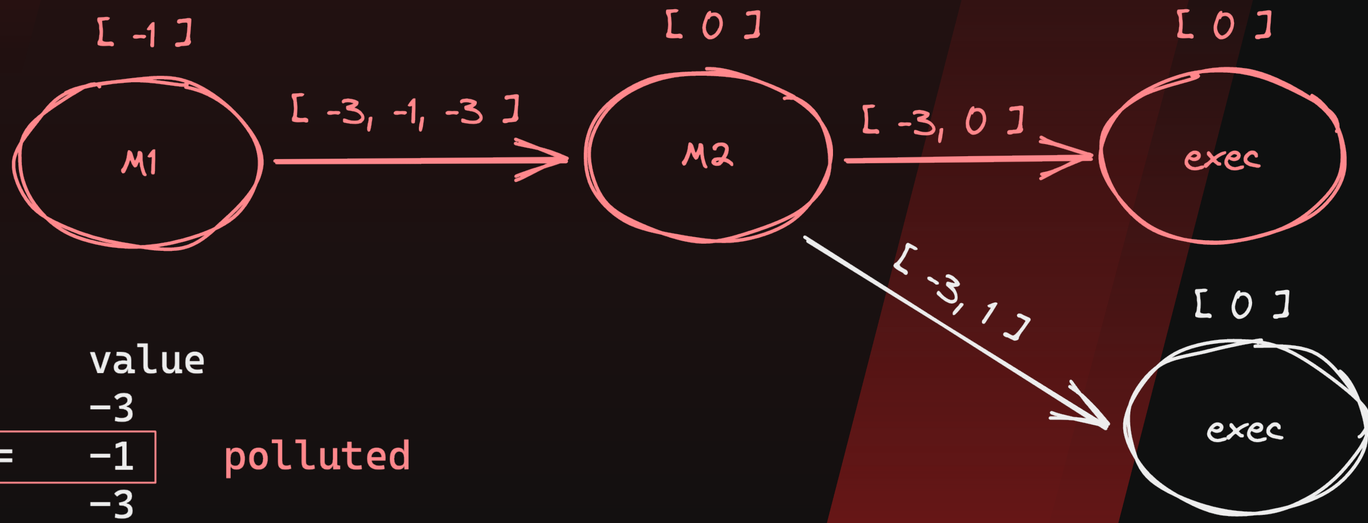


污点分析引擎 | 化零为整

```
void M1( )
{
    A a = new A();

    // [-3, -1, -3]
    a.M2( this.data, "ifconfig" );
}
```

```
void M2( String arg0, String arg1 )
{
    if( arg0 ≠ null ){
        Runtime.exec( arg0 ); // [-3, 0]
    } else {
        Runtime.exec( arg1 ); // [-3, 1]
    }
}
```



current polluted position [-1]

[-3, -1, -3] ⇒ position value

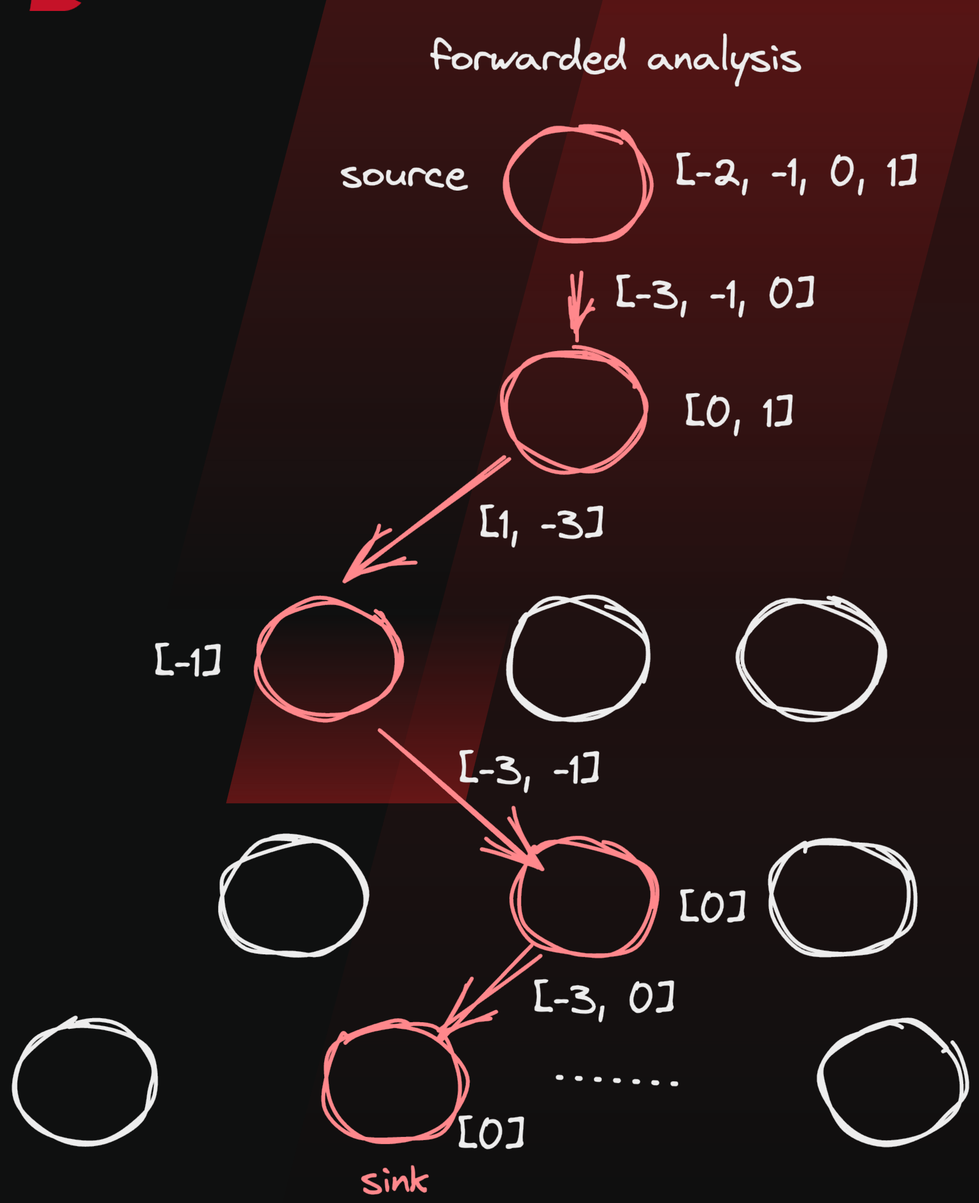
0 = this		-3
1 = param-0	⇐	-1
2 = param-1		-3

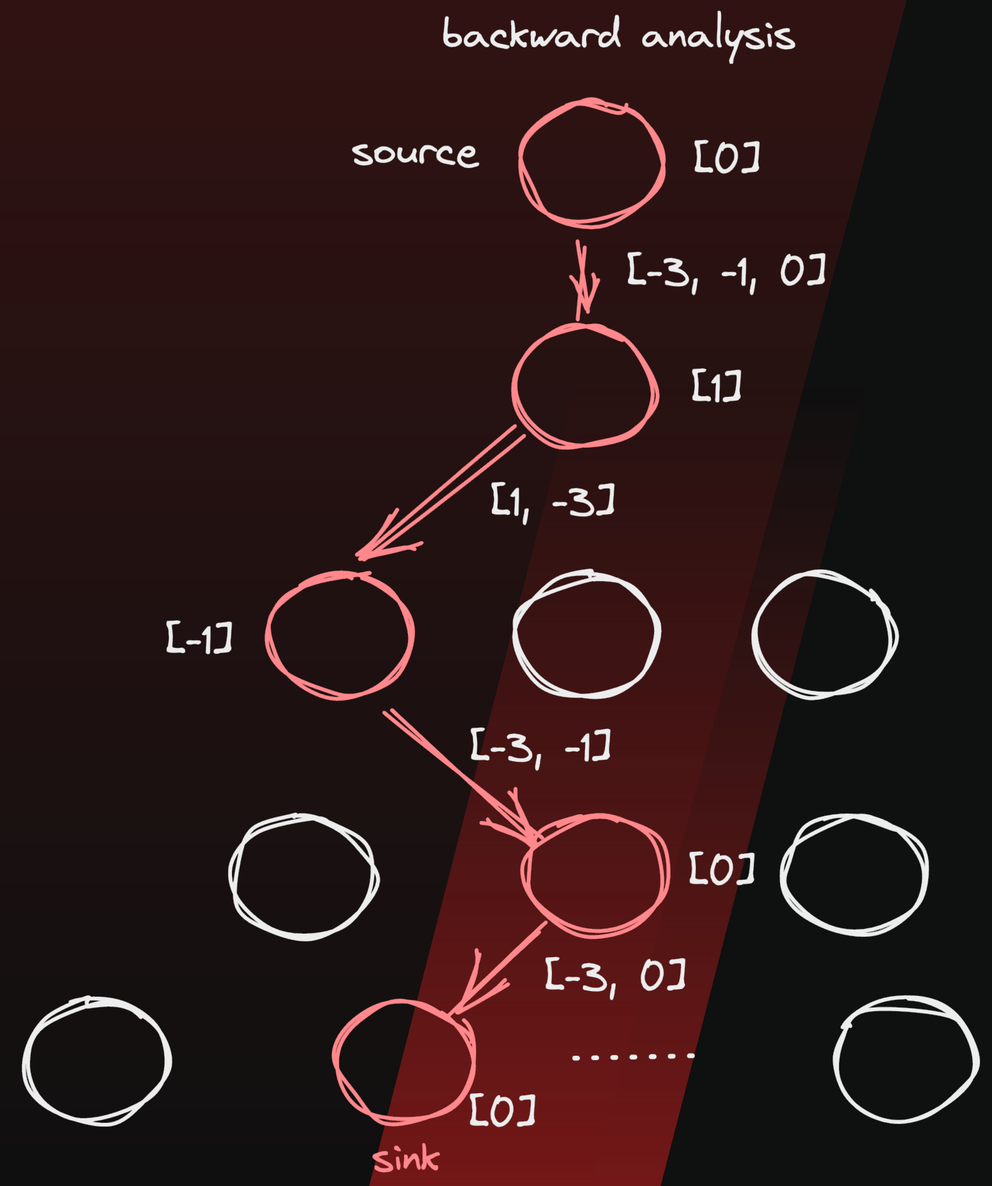
polluted

next polluted position ⇒ [0]

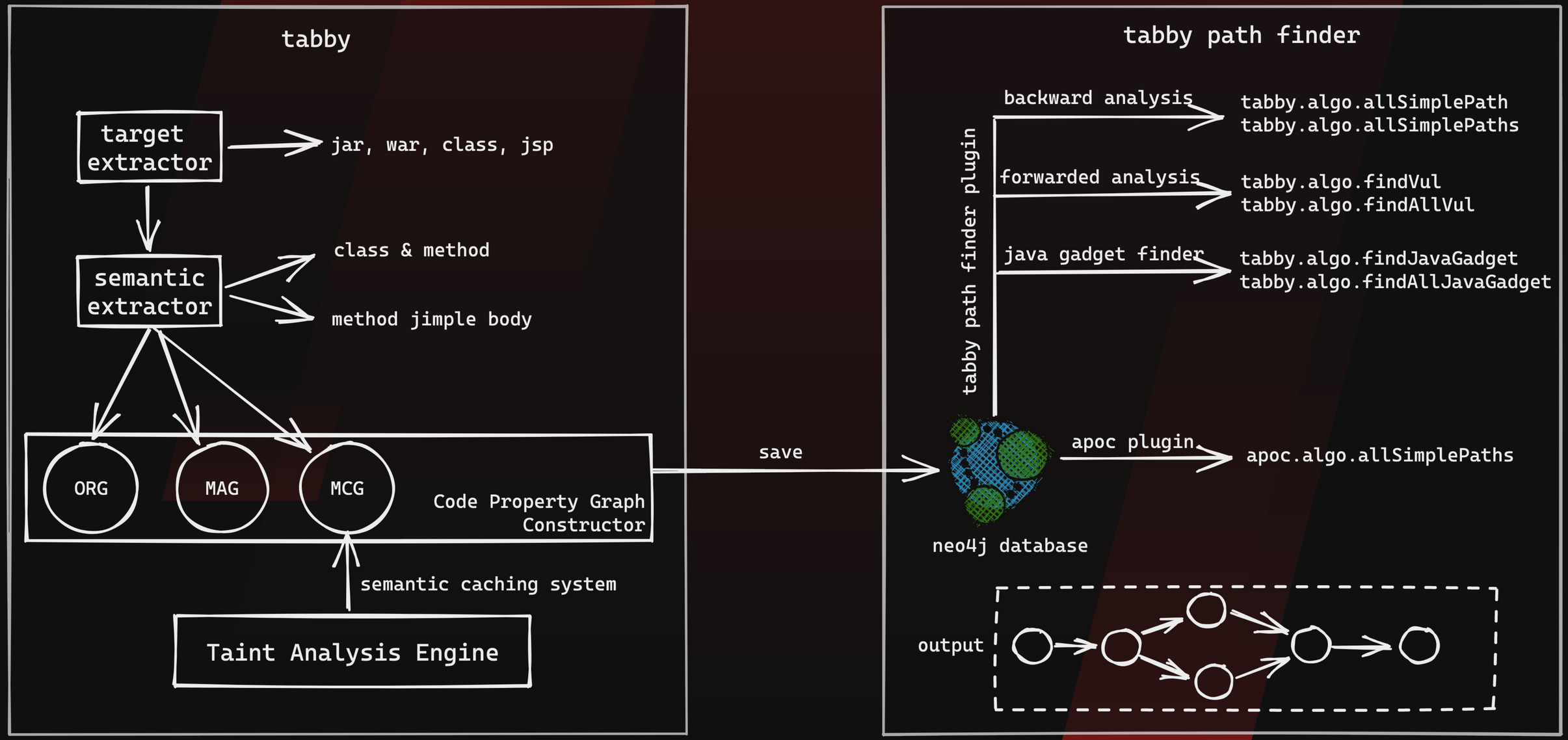
KCon

污点分析引擎 | 化零为整 tabby path finder





小结 look bigger



KCon

+1

Find Java Gadget like a pro

KCon

Find Java Gadget like a pro

+1

Java反序列化漏洞

2015年Frohoff以及FoxGlove Security团队发表了关于Java反序列化漏洞原理以及利用方式。Java反序列化漏洞其本质是**“不安全的反序列化”**，攻击者构造恶意的序列化数据用于正常的反序列化功能，从而导致本不该被访问的对象被调用执行。

可控的Java反序列化触发点

如ObjectInputStream的输入数据是可控的，且后续调用了readObject函数

有效的Java反序列化利用链

利用当前项目的开源基库构造有效的序列化链，该链可最终达成危险函数或对象的调用执行

最终的利用达成

通过有效的利用链，使得应用可任意调用危险函数或对象的执行，如Runtime的exec函数，用于执行系统命令。

XStream反序列化利用链source特征：

1. 存在5种Magic method
2. 无Serializable限制，甚至可还原Method对象

```
match (source:Method)
    where source.NAME in
        [
            "hashCode", "equals",
            "compareTo", "compare",
            "toString", "readObject"
        ]
with collect(source) as sources
```

Converters	可利用的还原对象	Magic Method
SerializableConverter	Serializable对象	readObject
CollectionConverter	HashSet、LinkedHashSet	hashCode、equals
MapConverter	HashMap	hashCode、toString[1]
TreeSet/TreeMapConverter	TreeSet、TreeMap	compareTo, compare

构造sink限制语句

```
match (sink:Method {IS_SINK:true,NAME:"invoke"})
with sources, collect(sink) as sources
```

反序列化利用链最终达成的效果常为如下几个sink函数：

构造tabby路径检索函数调用

1. 反射调用任意函数，Method.invoke
2. JNDI连接，lookup或更底层的函数
3. 文件操作，如任意文件写的相关函数

```
call tabby.algo.allSimplePaths(sinks, sources, 12, false) yield path
```

```
return path limit 1
```

[1] https://github.com/mbechler/marshalsec/blob/master/src/main/java/marshalsec/gadgets/ToStringUtil.java#L

KCon

Find Java Gadget like a pro | XStream

+1

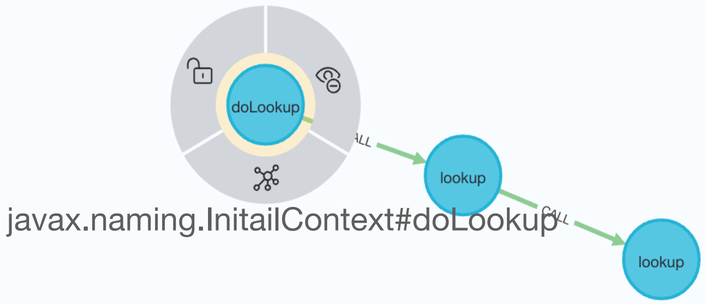
neo4j@neo4j://192.168.100.184:7687/neo4j - Neo4j Browser

```
1 match (source:Method {NAME:"toString"})
2 with collect(source) as sources
3 match (sink:Method {IS_SINK: true, NAME:"invoke"})
4 with sources, collect(sink) as sinks
5 call tabby.algo.allSimplePaths(sinks, sources, 7, false) yield path
6 where none(n in nodes(path) where n.NAME0 in
```

追加限制条件

```
1 match (source:Method {IS_PUBLIC:true, IS_STATIC:true})
2 with collect(source) as sources
3 match (sink:Method {IS_SINK: true, VUL:"JNDI"})
4 with sources, collect(sink) as sinks
5 call tabby.algo.allSimplePaths(sinks, sources, 3, false) yield path
6 return path limit 50
```

Graph
Table
Text
Code



javax.naming.InitailContext#doLookup

Node Properties

Method

IS_STATIC	true
MODIFIERS	9
NAME	doLookup
NAME0	javax.naming.InitialContext.doLookup
PARAMETER_SIZE	1
POLLUTED_POSITION	[]
RETURN_TYPE	java.lang.Object
SIGNATURE	<javax.naming.InitialContext:

Integer

javax.swing.UIDefaults$LazyValue#createValue
sun.swing.SwingLazyValue#createValue
java.lang.reflect.Method#invoke

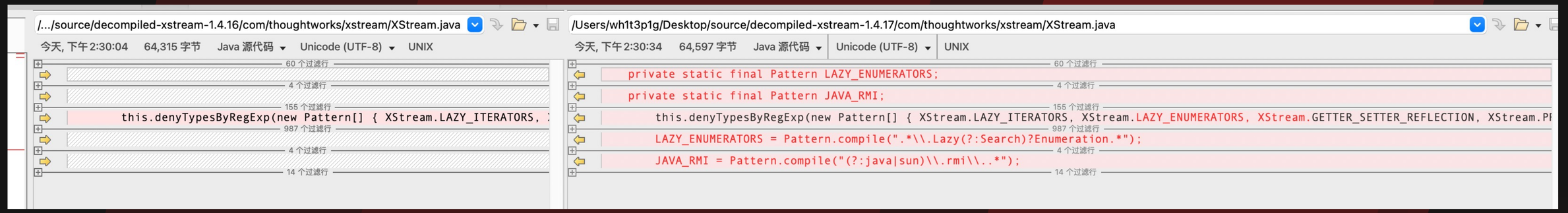
Find Java Gadget like a pro | XStream

依据先验知识，挖掘新利用链

CVE	Description
Version 1.4.16	
CVE-2021-29505	XStream is vulnerable to a Remote Command Execution attack.
Version 1.4.15	
CVE-2021-21341	XStream can cause a Denial of Service.
CVE-2021-21342	A Server-Side Forgery Request can be activated unmarshalling with XStream to access data streams from an arbitrary URL referencing a resource in an intranet or the local host.
CVE-2021-21343	XStream is vulnerable to an Arbitrary File Deletion on the local host when unmarshalling as long as the executing process has sufficient rights.
CVE-2021-21344	XStream is vulnerable to an Arbitrary Code Execution attack.
CVE-2021-21345	XStream is vulnerable to a Remote Command Execution attack.
CVE-2021-21346	XStream is vulnerable to an Arbitrary Code Execution attack.
CVE-2021-21347	XStream is vulnerable to an Arbitrary Code Execution attack.
CVE-2021-21348	XStream is vulnerable to an attack using Regular Expression for a Denial of Service (ReDos).
CVE-2021-21349	A Server-Side Forgery Request can be activated unmarshalling with XStream to access data streams from an arbitrary URL referencing a resource in an intranet or the local host.
CVE-2021-21350	XStream is vulnerable to an Arbitrary Code Execution attack.
CVE-2021-21351	XStream is vulnerable to an Arbitrary Code Execution attack.
Version 1.4.14	
CVE-2020-26258	A Server-Side Forgery Request can be activated unmarshalling with XStream to access data streams from an arbitrary URL referencing a resource in an intranet or the local host.
CVE-2020-26259	XStream is vulnerable to an Arbitrary File Deletion on the local host when unmarshalling as long as the executing process has sufficient rights.
Version 1.4.13	
CVE-2020-26217	XStream can be used for Remote Code Execution.
Version 1.4.9	
CVE-2017-7957	XStream can cause a Denial of Service when unmarshalling void.
Version 1.4.8	
CVE-2016-3674	XML External Entity (XXE) Vulnerability in XStream.
Version 1.4.6 (and 1.4.10)	
CVE-2013-7285	XStream can be used for Remote Code Execution.

Find Java Gadget like a pro | XStream

XStream 1.4.17 黑名单



对象字符黑名单

```
this.denyTypes(new String[] {
        "java.beans.EventHandler",
        "java.lang.ProcessBuilder",
        "javax.imageio.ImageIO$ContainsFilter",
        "jdk.nashorn.internal.objects.NativeString",
        "com.sun.corba.se.impl.activation.ServerTableEntry",
        "com.sun.tools.javac.processing.JavacProcessingEnvironment$NameProcessIterator",
        "sun.awt.datatransfer.DataTransferer$IndexOrderComparator",
        "sun.swing.SwingLazyValue"
});
```

对象正则黑名单

```
GETTER_SETTER_REFLECTION = Pattern.compile(".*\\$GetterSetterReflection");
PRIVILEGED_GETTER = Pattern.compile(".*\\$PrivilegedGetter");
LAZY_ENUMERATORS = Pattern.compile(".*\\.Lazy(?:Search)?Enumeration.*");
LAZY_ITERATORS = Pattern.compile(".*\\$LazyIterator");
JAXWS_ITERATORS = Pattern.compile(".*\\$ServiceNameIterator");
JAVAFX_OBSERVABLE_LIST__ = Pattern.compile("javafx\\.collections\\.ObservableList\\$.*");
JAVAX_CRYPTO = Pattern.compile("javax\\.crypto\\..*");
JAVA_RMI = Pattern.compile("(?:java|sun)\\.rmi\\..*");
BCEL_CL = Pattern.compile(".*\\.bcel\\..*\\.util\\.ClassLoader");
```

非法继承黑名单

```
this.denyTypeHierarchy(InputStream.class);
this.denyTypeHierarchyDynamically("java.nio.channels.Channel");
this.denyTypeHierarchyDynamically("javax.activation.DataSource");
this.denyTypeHierarchyDynamically("javax.sql.rowset.BaseRowSet");
```

Find Java Gadget like a pro | XStream

XStream 1.4.16 绕过 CVE-2021-29505

```
javax.naming.ldap.Rdn$RdnEntry#compareTo
com.sun.org.apache.xpath.internal.objects.XString#equal
com.sun.xml.internal.ws.api.message.Packet#toString
com.sun.xml.internal.ws.message.saaj.SAAJMessage#copy
com.sun.xml.internal.ws.message.saaj.SAAJMessage#getAttachments
com.sun.xml.internal.ws.message.saaj.SAAJMessage$SAAJAttachmentSet#<init>
com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl#getAttachments
com.sun.xml.internal.messaging.saaj.soap.ver1_1.Message1_1Impl#initializeAllAttachments
com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart#getCount
com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart#parse
com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimePullMultipart#parseAll
com.sun.xml.internal.org.jvnet.mimepull.MIMEMessage#getAttachments
com.sun.xml.internal.org.jvnet.mimepull.MIMEMessage#parseAll
com.sun.xml.internal.org.jvnet.mimepull.MIMEMessage#makeProgress
com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator#hasNext
com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeyStoreIterator#findNextCert
com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl#nextElement
com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl#findNextMatch
com.sun.jndi.rmi.registry.BindingEnumeration#next
sun.rmi.registry.RegistryImpl_Stub#lookup
```

前半部分依然可用

命中

命中

黑名单修复

```
Pattern.compile(
      ".*\\.Lazy(?:Search)?Enumeration.*");


Pattern.compile("(?:java|sun)\\.rmi\\..*");
```

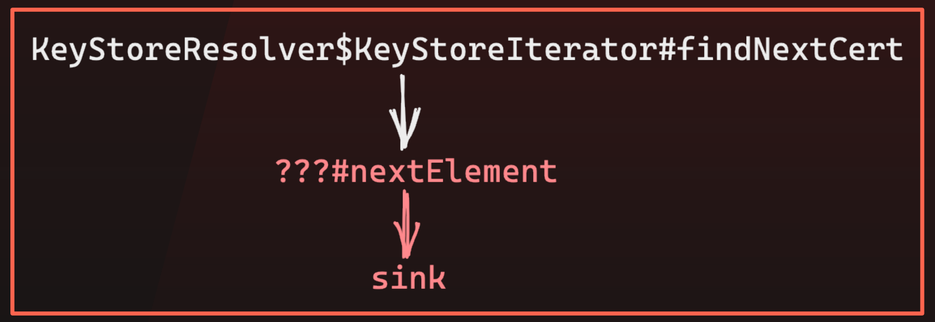
那么，该怎么利用这半条利用链呢？

Find Java Gadget like a pro | XStream

CVE-2021-29505 利用链分析

```
private Certificate findNextCert() {
    while (this.aliases.hasMoreElements()) {
        String alias = this.aliases.nextElement();
        try {
            Certificate cert = this.keyStore.getCertificate(alias);
            if (cert != null) {
                return cert;
            }
        } catch (KeyStoreException ex) {
            return null;
        }
    }
    return null;
}
```

aliases ⇒ com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl



XStream 1.4.17 bypass 转化为寻找合适链路：

1. 函数名nextElement
2.实现了java.util.Enumeration接口
3.存在一条链路能从source到特定sink函数

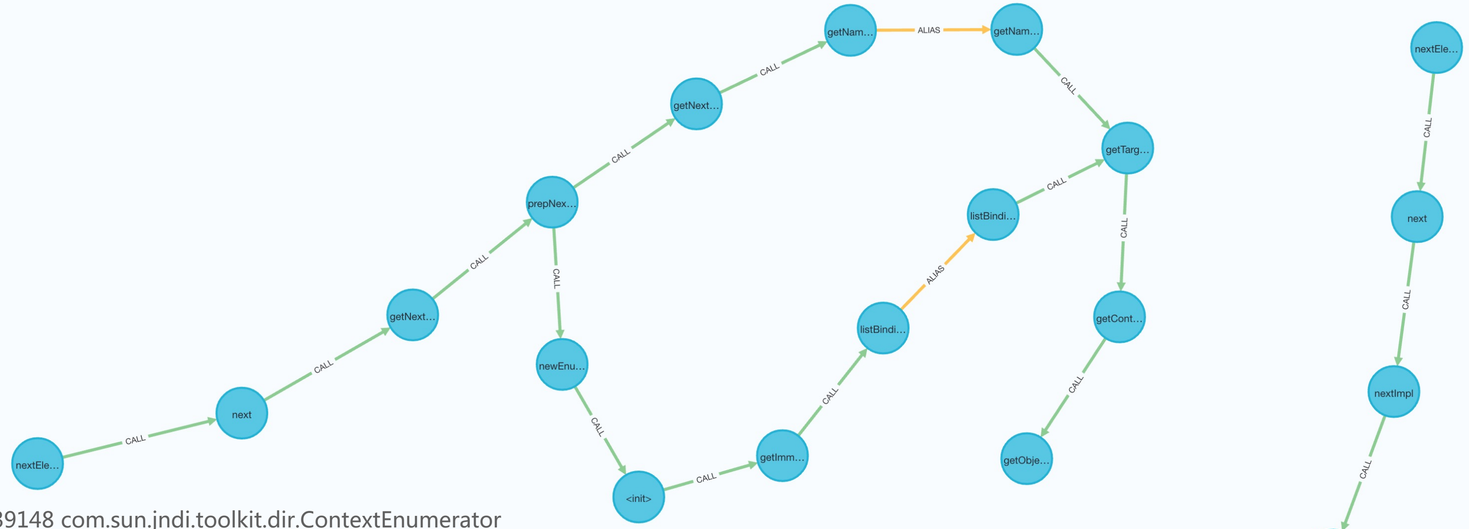
Find Java Gadget like a pro | XStream

neo4j@neo4j://192.168.100.184:7687/neo4j - Neo4j Browser

```
match (source:Method {NAME:"nextElement"})
                    <-[:HAS]-(cls:Class)-[:INTERFACE|EXTENDS*]
                    ->(cls1:Class {NAME:"java.util.Enumeration"})
with collect(source) as sources
match (sink:Method {IS_SINK:true, VUL:"JNDI"})
with sources, collect(sink) as sinks
call tabby.algo.allSimplePaths(sinks, sources, 15, false) yield path
where none(n in nodes(path) where n.CLASSNAME in
["com.sun.jndi.toolkit.dir.LazySearchEnumerationImpl","com.sun.jndi.toolkit.dir.HierMemDirCtx$BaseFlatNames","com.sun.jndi.dns.BaseNameClassPairEnumeration","java.util.jar.JarVerifier$2","com.sun.jndi.cosnaming.CNBindingEnumeration","com.sun.jndi.dns.BindingEnumeration","com.sun.jndi.cosnaming.CNCtx"] or n.NAME0 in ["com.sun.jndi.ldap.LdapReferralContext.getNameParser"])
return path limit 6
```

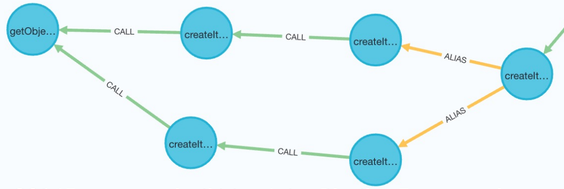
构造source条件

构造sink条件，关注JNDI



CVE-2021-39148 com.sun.jndi.toolkit.dir.ContextEnumerator

CVE-2021-39147 com.sun.jndi.ldap.LdapSearchEnumeration



CVE-2021-39145 com.sun.jndi.ldap.LdapBindingEnumeration

Overview

Node labels

* (25) Method (25)

Relationship Types

* (25) CALL (21) ALIAS (4)

Displaying 25 nodes, 52 relationships.

Find Java Gadget like a pro

小结

本节分享了两种利用链挖掘的方法，但其实质都在于如何构造好查询语句

```
// tabby path finder plugin
match (source:Method)
with collect(source) as sources

match (sink:Method {IS_SINK:true})
with sources, collect(sink) as sinks

call tabby.algo.xxxx() yield path

return path limit 1
```

```
// apoc plugin
match (source:Method)


match (sink:Method {IS_SINK:true})


call apoc.algo.allSimplePaths(sink, source, "<CALL|<ALIAS", 12) yield path

return path limit 1
```

利用链的挖掘过程转化成了

1. 初始模式识别并转化为cypher语句（序列化机制特征）
2. 不断优化查询语句（添加where限制），不断验证所输出利用链的有效性

https://tabby-db-files.oss-cn-hangzhou.aliyuncs.com/jdk/1.8.0_292/graphdb.mv.db

KCon

+1

Find Java Web Vulnerabilities like a pro

Find Java Web Vulnerabilities like a pro

相比利用链的挖掘，Java Web应用的特征识别则相对简单，tabby默认内置了如下的端点识别

Struts类型

```
classRef.setStrutsAction(relatedClassnames.contains("com.opensymphony.xwork2.ActionSupport")
                        || relatedClassnames.contains("com.opensymphony.xwork2.Action")
                        || relatedClassnames.contains("org.apache.struts.actions.DispatchAction") // struts 1.x
);
```

JSP类型

```
// check jsp _jspService
if("_jspService".equals(methodName)){
    return true;
}
```

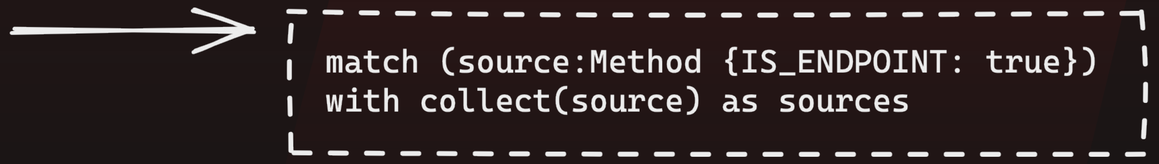
Servlet类型

```
// check from servlet
// https://blog.csdn.net/melissa_heixiu/article/details/52472450
List<String> servletMethods =
        Arrays.asList("doGet","doPost","doPut","doDelete","doHead","doOptions","doTrace","service");
if((relatedClassnames.contains("javax.servlet.Servlet")
        || relatedClassnames.contains("javax.servlet.http.HttpServlet")
        || relatedClassnames.contains("javax.servlet.GenericServlet"))
        && servletMethods.contains(methodName)){
    return true;
}
```

注释类型

```
// check from annotation
List<Tag> tags = method.getTags();
for (Tag tag : tags) {
    if (tag instanceof VisibilityAnnotationTag) {
        VisibilityAnnotationTag visibilityAnnotationTag = (VisibilityAnnotationTag) tag;
        for (AnnotationTag annotationTag : visibilityAnnotationTag.getAnnotations()) {
            String type = annotationTag.getType();

            if(type.endsWith("Mapping;")
                    || type.endsWith("javax/ws/rs/Path;")
                    || type.endsWith("javax/ws/rs/GET;")
                    || type.endsWith("javax/ws/rs/PUT;")
                    || type.endsWith("javax/ws/rs/DELETE;")
                    || type.endsWith("javax/ws/rs/POST;")){
                return true;
            }
        }
    }
}
```

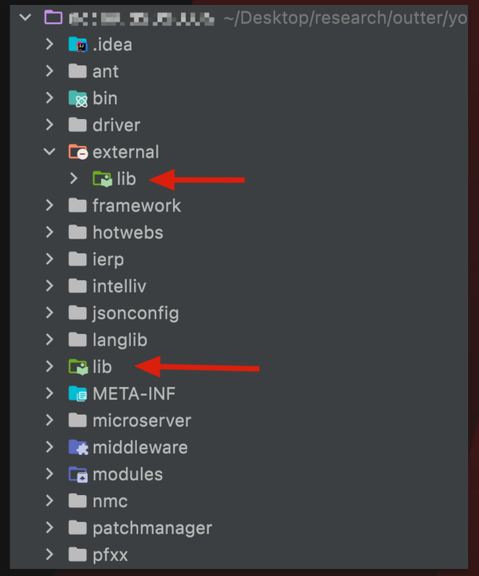


KCon

Find Java Web Vulnerabilities like a pro

+1

取某著名oa系统依赖库生成代码属性图



nc.bs.framework.server.InvokerServlet#doAction

```
match (source:Method {IS_ENDPOINT:true})
with collect(source) as sources
```

```
match (source:Method {NAME:"doAction"})
    <-[:HAS]-(c:Class)-[:INTERFACE|EXTENDS*]
    ->(c1:Class {NAME:"nc.bs.framework.adaptor.IHttpServletAdaptor"})
with collect(source) as sources
```

```
match (source:Method {
    SUB_SIGNATURE:
        "void doAction(javax.servlet.http.HttpServletRequest,
                       javax.servlet.http.HttpServletResponse)"
})
with collect(source) as sources
```

1. Servlet类型，主动调用serivce函数
2. IHttpServletAdaptor类型，主动调用doAction函数
3. doAction函数，且参数类型为HttpServletRequest、HttpServletResponse

KCon

Find Java Web Vulnerabilities like a pro

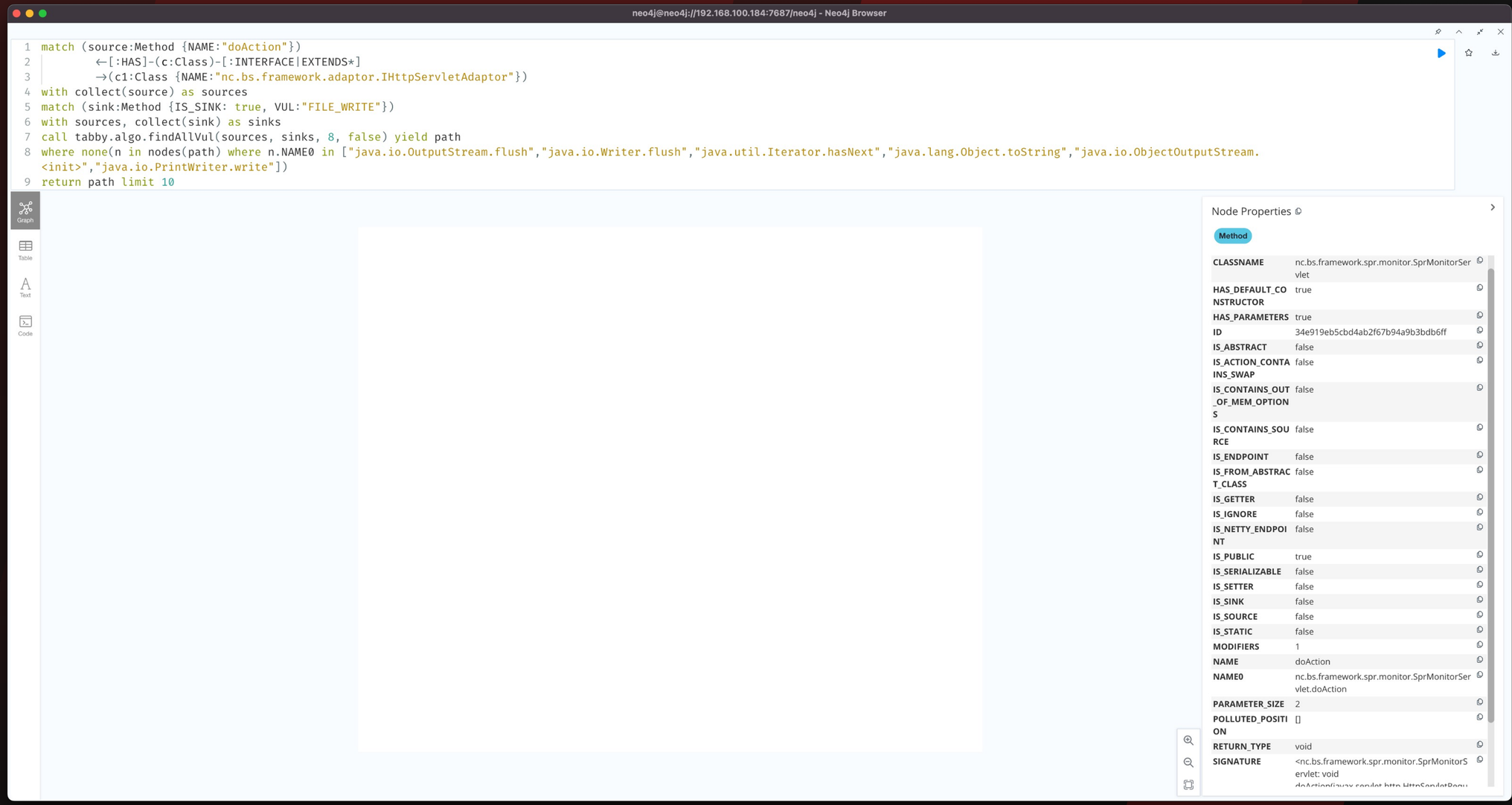
+1

彩蛋在哪里XD

针对常见的Web漏洞，tabby内置了常见的sink函数，使用VUL标签来区分：

- SQLI
- SSRF
- FILE
- FILE_WRITE
- CODE
- EXEC
- XXE
- SERIALIZE

除此之外，对于Web漏洞，推荐使用前向分析，由source函数开始查找至sink函数



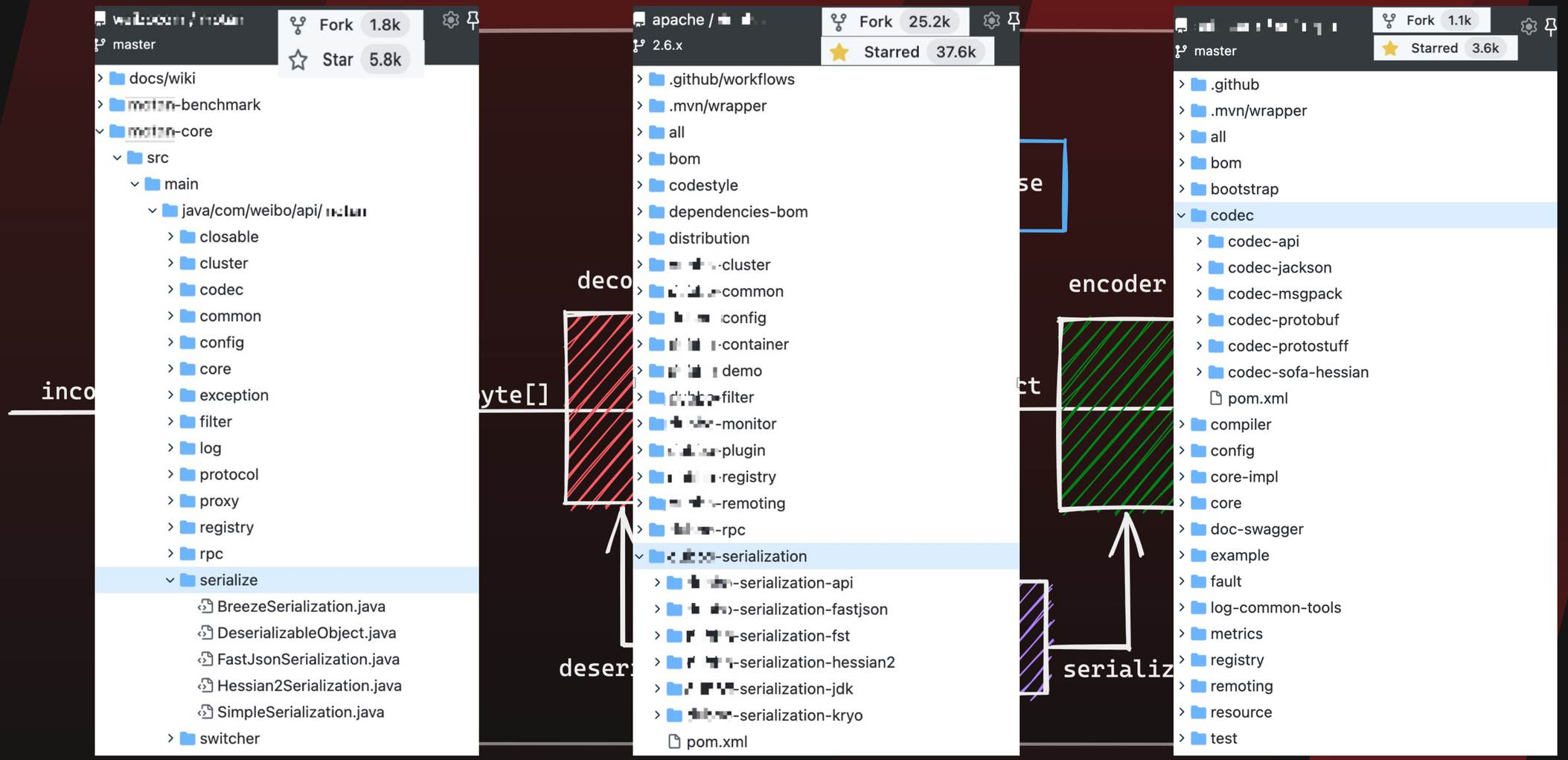
KCon

+1

Find Java RPC Framework Vulnerabilities like a pro

KCon

Find Java RPC Framework Vulnerabilities like a pro



Find Java RPC Framework Vulnerabilities like a pro

com.weibo.api.motan.transport.netty.NettyServer#initServerBootstrap

```
bootstrap.setPipelineFactory(new ChannelPipelineFactory() {
    // FrameDecoder非线程安全，每个连接一个 Pipeline
    @Override
    public ChannelPipeline getPipeline() {
        ChannelPipeline pipeline = Channels.pipeline();
        pipeline.addLast( name: "channel_manage", channelManage);
        pipeline.addLast( name: "decoder", new NettyDecoder(codec, client: NettyServer.this, maxContentLength))
        pipeline.addLast( name: "encoder", new NettyEncoder(codec, client: NettyServer.this));
        pipeline.addLast( name: "nettyChannelHandler", nettyChannelHandler);
        return pipeline;
    }
});
```

com.weibo.api.motan.transport.netty.NettyDecoder#decode

```
@Override
protected Object decode(ChannelHandlerContext ctx, Channel channel, ChannelBuffer buffer) throws Exception {
    //根据版本号决定走哪个分支
    ...
    switch (rpcVersion) {
        case 0:
            result = decodeV1(ctx, channel, buffer);
            break;
        case 1:
            result = decodeV2(ctx, channel, buffer);
            break;
        default:
            result = decodeV2(ctx, channel, buffer);
    }

    ...
    return result;
}
```

```
private Object decodeV1(ChannelHandlerContext ctx, Channel channel,
    ...
    return decode(data, channel, isRequest: messageType == MotanConst
}
```

```
private Object decode(byte[] data, Channel channel, boolean isRe
    ...
    try {
        return codec.decode(client, remoteIp, data);
    } ...
}
```

Find Java RPC Framework Vulnerabilities like a pro

com.weibo.api.motan.protocol.rpc.DefaultRpcCodec#decode

```
public Object decode(Channel channel, String remoteIp, byte[] data) throws IOException {
    ...
    Serialization serialization = ExtensionLoader.getExtensionLoader(Serialization.class)
            .getExtension(channel.getUrl().getParameter(URLParamType.serialize.getName(), URLParamType.serialize.getValue()));

    try {
        if (isResponse) { // response
            return decodeResponse(body, dataType, requestId, serialization);
        } else {
            return decodeRequest(body, requestId, serialization);
        }
```

com.weibo.api.motan.protocol.rpc.DefaultRpcCodec#decodeRequest

```
private Object decodeRequest(byte[] body, long requestId, Serialization serialization) throws IOException, ClassNotFoundException {
    ByteArrayInputStream inputStream = new ByteArrayInputStream(body);
    ObjectInput input = createInput(inputStream);

    String interfaceName = input.readUTF();
    String methodName = input.readUTF();
    String paramtersDesc = input.readUTF();

    DefaultRequest rpcRequest = new DefaultRequest();
    rpcRequest.setRequestId(requestId);
    rpcRequest.setInterfaceName(interfaceName);
    rpcRequest.setMethodName(methodName);
    rpcRequest.setParamtersDesc(paramtersDesc);
    rpcRequest.setArguments(decodeRequestParameter(input, paramtersDesc, serialization));
    rpcRequest.setAttachments(decodeRequestAttachments(input));

    input.close();

    return rpcRequest;
}
```

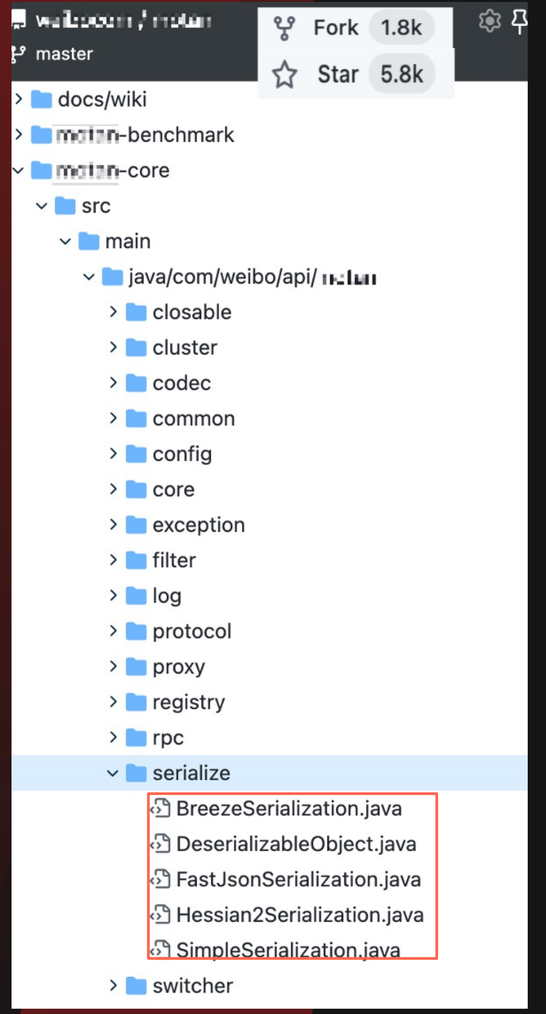
```
private Object[] decodeRequestParameter(ObjectInput input, String parameterDesc, Serializat
        ClassNotFoundException {
    if (parameterDesc == null || parameterDesc.equals("")) {
        return null;
    }

    Class<?>[] classTypes = ReflectUtil.forNames(parameterDesc);

    Object[] paramObjs = new Object[classTypes.length];

    for (int i = 0; i < classTypes.length; i++) {
        paramObjs[i] = deserialize((byte[]) input.readObject(), classTypes[i], serialization);
    }

    return paramObjs;
}
```



Find Java RPC Framework Vulnerabilities like a pro


```java
public static boolean isNettyEndpoint(SootMethod method, Set<String> relatedClassnames){
    String classname = method.getDeclaringClass().getName();
    if("io.netty.channel.ChannelInboundHandler".equals(classname)
            || "io.netty.handler.codec.ByteToMessageDecoder".equals(classname)
            || "io.netty.handler.codec.MessageToMessageDecoder".equals(classname)
            || "org.jboss.netty.handler.codec.frame.FrameDecoder".equals(classname)
            || "org.jboss.netty.channel.SimpleChannelUpstreamHandler".equals(classname)
    ){
        return false;
    }

    String methodName = method.getName();

    // check from Decoder
    if( (relatedClassnames.contains("io.netty.handler.codec.ByteToMessageDecoder")
            || relatedClassnames.contains("io.netty.handler.codec.MessageToMessageDecoder")
            || relatedClassnames.contains("org.jboss.netty.handler.codec.frame.FrameDecoder")
    ) && "decode".equals(methodName)){
        return true;
    }

    // check from ChannelInboundHandler
    List<String> nettyReadMethods = Arrays.asList("channelRead", "channelRead0", "messageReceived");
    if((relatedClassnames.contains("io.netty.channel.ChannelInboundHandler")
            || relatedClassnames.contains("org.jboss.netty.channel.SimpleChannelUpstreamHandler")
        ) && nettyReadMethods.contains(methodName)){
        return true;
    }

    // not an endpoint
    return false;
}
```

已知的Netty通用调用

未知的用户逻辑

已知的反序列化逻辑

KCon

Find Java RPC Framework Vulnerabilities like a pro

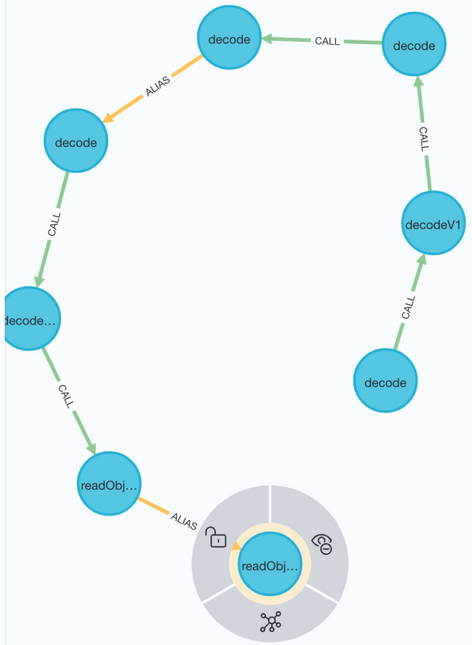
+1

neo4j@neo4j://192.168.100.184:7687/neo4j - Neo4j Browser

```
1 match (source:Method {IS_NETTY_ENDPOINT:true})
2     where not(source.CLASSNAME =~ "org.jboss.*")
3 with collect(source) as sources
4 match (sink:Method {IS_SINK:true, CLASSNAME:"java.io.ObjectInputStream"})
5 with sources, collect(sink) as sinks
6 call tabby.algo.findAllVul(sources, sinks, 12, false) yield path
7 return path limit 1
```

Graph | Table | Text | Code

Java原生反序列化调用链



Node Properties

Method

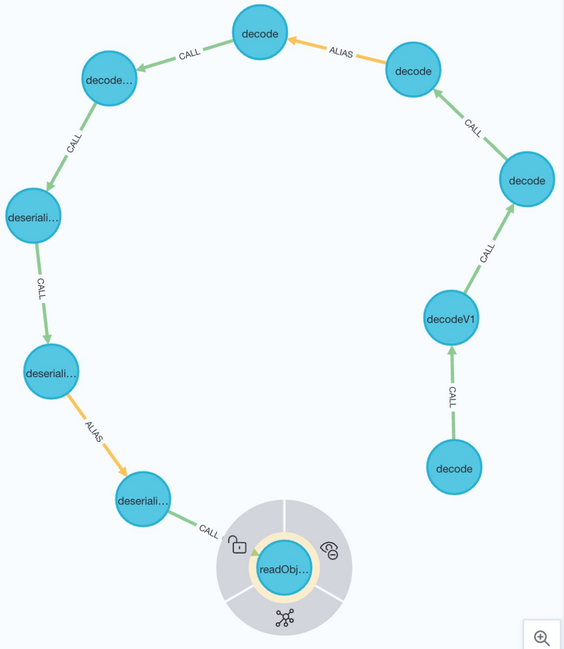
ID	e9c146afdc3e5d83bb2e56563210e166
IS_ABSTRACT	false
IS_ACTION_CONTAINS_SWAP	false
IS_CONTAINS_OUT_OF_MEM_OPTIONS	false
IS_CONTAINS_SOURCE	false
IS_ENDPOINT	false
IS_FROM_ABSTRACT_CLASS	false
IS_GETTER	false
IS_IGNORE	false
IS_NETTY_ENDPOINT	false
IS_PUBLIC	true
IS_SERIALIZABLE	false
IS_SETTER	false
IS_SINK	true
IS_SOURCE	false
IS_STATIC	false
MODIFIERS	17
NAME	readObject
NAME0	java.io.ObjectInputStream.readObject
PARAMETER_SIZE	0
POLLUTED_POSITION	[-1]
RETURN_TYPE	java.lang.Object
SIGNATURE	<java.io.ObjectInputStream: java.lang.Object readObject()>
SUB_SIGNATURE	java.lang.Object readObject()
VUL	SERIALIZE

neo4j@neo4j://192.168.100.184:7687/neo4j - Neo4j Browser

```
1 match (source:Method {IS_NETTY_ENDPOINT:true})
2     where not(source.CLASSNAME =~ "org.jboss.*")
3 with collect(source) as sources
4 match (sink:Method {IS_SINK:true,
  CLASSNAME:"com.caucho.hessian.io.Hessian2Input"})
5 with sources, collect(sink) as sinks
6 call tabby.algo.findAllVul(sources, sinks, 12, false) yield path
7 return path limit 1
```

Graph | Table | Text | Code

Hessian反序列化调用链



Node Properties

Method

TAINS_SWAP	
IS_CONTAINS_OUT_OF_MEM_OPTIONS	false
IS_CONTAINS_SOURCE	false
IS_ENDPOINT	false
IS_FROM_ABSTRACT_CLASS	false
IS_GETTER	false
IS_IGNORE	false
IS_NETTY_ENDPOINT	false
IS_PUBLIC	true
IS_SERIALIZABLE	false
IS_SETTER	false
IS_SINK	true
IS_SOURCE	false
IS_STATIC	false
MODIFIERS	1
NAME	readObject
NAME0	com.caucho.hessian.io.Hessian2Input.readObject
PARAMETER_SIZE	1
POLLUTED_POSITION	[-1]
RETURN_TYPE	java.lang.Object
SIGNATURE	<com.caucho.hessian.io.Hessian2Input: java.lang.Object readObject(java.lang.Class)>
SUB_SIGNATURE	java.lang.Object readObject(java.lang.Class)
VUL	SERIALIZE

Find Java RPC Framework Vulnerabilities like a pro

Hessian利用链 SpringAbstractBeanFactoryPointcutAdvisor

```
org.springframework.aop.support.AbstractPointcutAdvisor#equals
org.springframework.aop.support.AbstractBeanFactoryPointcutAdvisor#getAdvice
org.springframework.beans.factory.BeanFactory#getBean
```

```
match (source:Method {NAME:"getBean"})
      <-[:HAS]-(c:Class)-[:EXTENDS|INTERFACE*]
      ->(c1:Class {NAME:"org.springframework.beans.factory.BeanFactory"})
with collect(source) as sources


match (sink:Method {IS_SINK:true, NAME:"invoke"})
with sources, collect(sink) as sinks
call tabby.algo.findAllVul(sources, sinks, 6, false) yield path
return path limit 1
```

如果目标不出网？
如果目标JDK版本很高？
如果目标不存在原生反序列化利用链？
如何做到hessian to rce？

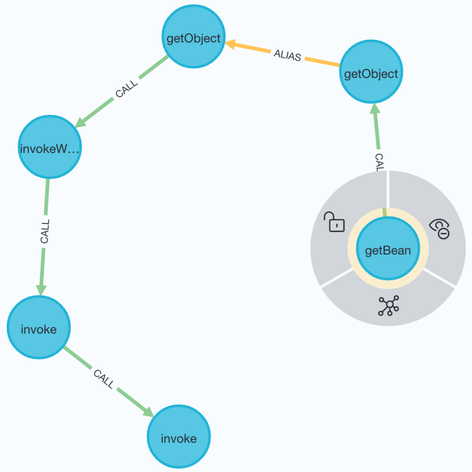
Hessian利用链 SpringPartiallyComparableAdvisorHolder

```
org.springframework.aop.aspectj.autoproxy.AspectJAwareAdvisorAutoProxyCreator$PartiallyComparableAdvisorHolder#toString
org.springframework.aop.aspectj.AspectJPointcutAdvisor#getOrder
org.springframework.aop.aspectj.AbstractAspectJAdvice#getOrder
org.springframework.aop.aspectj.annotation.BeanFactoryAspectInstanceFactory#getOrder
org.springframework.beans.factory.BeanFactory#getType
```

Find Java RPC Framework Vulnerabilities like a pro

neo4j@neo4j://192.168.100.184:7687/neo4j - Neo4j Browser

```
match (source:Method {NAME:"getBean"})<-[:HAS]-(c:Class)-[:EXTENDS|INTERFACE*]->(c1:Class
{NAME:"org.springframework.beans.factory.BeanFactory"})
with collect(source) as sources
match (sink:Method {IS_SINK:true, NAME:"invoke"})
with sources, collect(sink) as sinks
call tabby.algo.findAllVul(sources, sinks, 6, false) yield path
return path limit 1
```



Node Properties

Method

<id>	10622
ACTIONS	{"return":["thisu003cfu003ebeans","this"]}
CLASSNAME	org.springframework.beans.factory.support.StaticListableBeanFactory
HAS_DEFAULT_CONSTRUCTOR	true
HAS_PARAMETERS	true
ID	06f0fd00cf223c04eebe96e0380fb55e
IS_ABSTRACT	false
IS_ACTION_CONTAINS_SWAP	false
IS_CONTAINS_OUT_OF_MEM_OPTIONS	false
IS_CONTAINS_SOURCE	false
IS_ENDPOINT	false
IS_FROM_ABSTRACT_CLASS	false
IS_GETTER	false
IS_IGNORE	false
IS_NETTY_ENDPOINT	false
IS_PUBLIC	true
IS_SERIALIZABLE	false
IS_SETTER	false
IS_SINK	false
IS_SOURCE	false
IS_STATIC	false
MODIFIERS	1
NAME	getBean
NAME0	org.springframework.beans.factory.support.StaticListableBeanFactory.getBean
PARAMETER_SIZE	1
POLLUTED_POSITION	[]
RETURN_TYPE	java.lang.Object
SIGNATURE	<org.springframework.beans.factory.support.StaticListableBeanFactory: java.lang.Object getBean(java.lang.String)>

org.springframework.beans.factory.support.StaticListableBeanFactory#getBean
org.springframework.beans.factory.FactoryBean#getObject
org.springframework.beans.factory.config.MethodInvokingFactoryBean#getObject
org.springframework.beans.factory.config.MethodInvokingBean#invokeWithTargetException
org.springframework.util.MethodInvoker#invoke
java.lang.reflect.Method#invoke

org.springframework.util.MethodInvoker#invoke

```
public Object invoke() ... {
    // In the static case, target will simply be {@code null}.
    Object targetObject = getTargetObject();
    Method preparedMethod = getPreparedMethod();
    ...
    ReflectionUtils.makeAccessible(preparedMethod);
    return preparedMethod.invoke(targetObject, getArguments());
}
```

Reflection to RCE

```
public Object makeBean() throws Exception {
    MethodInvokingFactoryBean bean = new MethodInvokingFactoryBean();
    bean.setSingleton(false);
    bean.setTargetObject(Runtime.getRuntime());
    Class cls = Runtime.class;
    Method method = cls.getMethod( name: "exec", String[].class);
    ReflectionHelper.setFieldValue(bean,  fieldName: "methodObject", method);
    ReflectionHelper.setFieldValue(bean,  fieldName: "beanClassLoader",  value: null);
    bean.setArguments(new Object[]{new String[]{"bash", "-c", command}});
    return bean;
}
```

Find Java RPC Framework Vulnerabilities like a pro

One More Step !

如果是无spring依赖的情况，如何hessian to rce ?

JDK是一座深山，永远可以发现有意思的东西！

Hessian反序列化流程的特征：

1. **Magic method 类似Xstream，有equals、toString等函数**
2. **无法还原构造好的恶意Iterator、Enumeration、Map、List对象内容**
3. **默认使用unsafe初始化对象，无getter、setter调用**
4. **但也意味着可还原除特殊几个对象的任意对象，如Class、Method对象**
5. **类属性还原忽略Transient、Static**

```
match(source:Method {NAME:"toString"})
with collect(source) as sources

match(sink:Method {IS_SINK:true, NAME:"invoke"})
with sources, collect(sink) as sinks

call tabby.algo.allSimplePaths(sinks, sources, 12, false) yield path
where none(n in nodes(path) where n.CLASSNAME in [
                "java.util.Iterator",
                "java.util.Enumeration",
                "java.util.Map","java.util.List"])

return path limit 1
```

排除中间节点出现黑名单对象

Find Java RPC Framework Vulnerabilities like a pro

静态函数调用

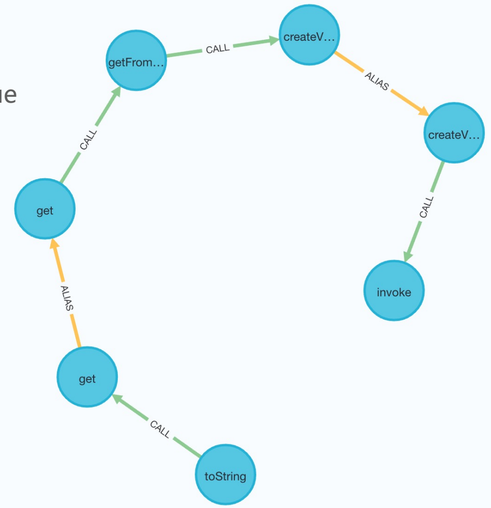
neo4j@neo4j://192.168.100.184:7687/neo4j - Neo4j Browser

```
match(source:Method {NAME:"toString", PARAMETER_SIZE:0})
with collect(source) as sources
match(sink:Method {IS_SINK:true, NAME:"invoke"})
with sources, collect(sink) as sinks
call tabby.algo.allSimplePaths(sinks, sources, 12, false) yield path
where none(n in nodes(path) where n.CLASSNAME
in["java.util.Iterator","java.util.Enumeration","java.util.Map","java.util.List","com.sun.jndi.ldap.ClientId","com.oracle.webservices.internal.api.message.BasePropertySet$MethodAccessor","jdk.nashorn.api.scripting.ScriptObjectMirror","java.awt.datatransfer.MimeTypeParameterList","com.sun.java.swing.plaf.gtk.GTKStyle$GTKLazyValue"])
return path limit 1
```

Graph / Table / Text / Code

XStream CVE-2021-21346 变种

javax.activation.MimeTypeParameterList#toString
java.util.Hashtable#get
javax.swing.UIDefaults#get
javax.swing.UIDefaults#getFromHashtable
javax.swing.UIDefaults$LazyValue#createValue
sun.swing.SwingLazyValue#createValue
java.lang.reflect.Method#invoke



Overview

Node labels

* (7) Method (7)

Relationship Types

* (6) CALL (4) ALIAS (2)

Displaying 7 nodes, 6 relationships.

sun.swing.SwingLazyValue#createValue

```
public Object createValue(final UIDefaults table) {
    try {
        ReflectUtil.checkPackageAccess(className);
        Class<?> c = Class.forName(className, initialize: true, loader: null);
        if (methodName != null) {
            Class[] types = getClassArray(args);
            Method m = c.getMethod(methodName, types);
            makeAccessible(m);
            return m.invoke(c, args);
```

Not JNDI Again !

FileWrite + URLClassLoader = RCE

KCon

Find Java RPC Framework Vulnerabilities like a pro

+1

1. write to /tmp/a.jar
2. load /tmp/a.jar
3. newInstance trigger static blocks
4. Execute any Java code

com.sun.org.apache.xml.internal.security.utils.JavaUtils#writeBytesToFilename

```java
public static void writeBytesToFilename(String filename, byte[] bytes) {
    FileOutputStream fos = null;
    try {
        if (filename != null && bytes != null) {
            File f = new File(filename);

            fos = new FileOutputStream(f);

            fos.write(bytes);
            fos.close();
```

sun.security.tools.keytool.Main#main

```java
public static void main(String[] args) throws Exception {
    Main kt = new Main();
    kt.run(args, System.out);
}

private void run(String[] args, PrintStream out) throws Exception {
    try {
        parseArgs(args);
        if (command != null) {
            doCommands(out);
        }
    } catch (Exception e) {
        System.out.println(rb.getString( key: "keytool.error.") + e);
        if (verbose) {
            e.printStackTrace(System.out);
        }
        if (!debug) {
            System.exit( status: 1);
        } else {
            throw e;
        }
    } finally {
        printWeakWarnings( newLine: false);
        for (char[] pass : passwords) {...}

        if (ksStream != null) {...}
    }
}
```

```java
void doCommands(PrintStream out) throws Exception {
    ...// Try to load and install specified provider
    if (providers != null) {
        ClassLoader cl = null;
        if (pathlist != null) {
            String path = null;
            ...
            path = PathList.appendPath(path, pathlist);

            URL[] urls = PathList.pathToURLs(path);
            cl = new URLClassLoader(urls);
        } else {...}
```

load class from local file

```java
        for (Pair <String, String> provider: providers) {
            String provName = provider.fst;
            Class<?> provClass;
            if (cl != null) {
                provClass = cl.loadClass(provName);
            } else {
                provClass = Class.forName(provName);
            }

            String provArg = provider.snd;
            Object obj;
            if (provArg == null) {
                obj = provClass.newInstance();
            } else {
                Constructor<?> c = provClass.getConstructor(PARAM_STRING);
                obj = c.newInstance(provArg);
            }
            if (!(obj instanceof Provider)) {...}
            Security.addProvider((Provider)obj);
        }
    }
```

class newInstance

KCon

总 结

+1

1. tabby善于链路挖掘，但模式识别仍需人工参与
2. 规则库的完善度决定了分析效果
3. 加内存！升级CPU！买高配云主机！！！

Happy Hunting Bugs！

Ps: 分享基于最新的 tabby 2.0，开源时间待定

Ps：分享的利用链均开源于ysomap

分享涉及的开源库：

1. https://github.com/wh1t3p1g/tabby
2. https://github.com/wh1t3p1g/tabby-path-finder
3. https://github.com/wh1t3p1g/ysomap

KCon

+1

感谢您的观看

THANK YOU FOR YOUR WATCHING

KCon 2022 黑客大会