Automatic Policy Synthesis and Enforcement for Protecting Untrusted Deserialization

Quan Zhang, Yiwen Xu, Zijing Yin, Chijin Zhou, Yu Jiang

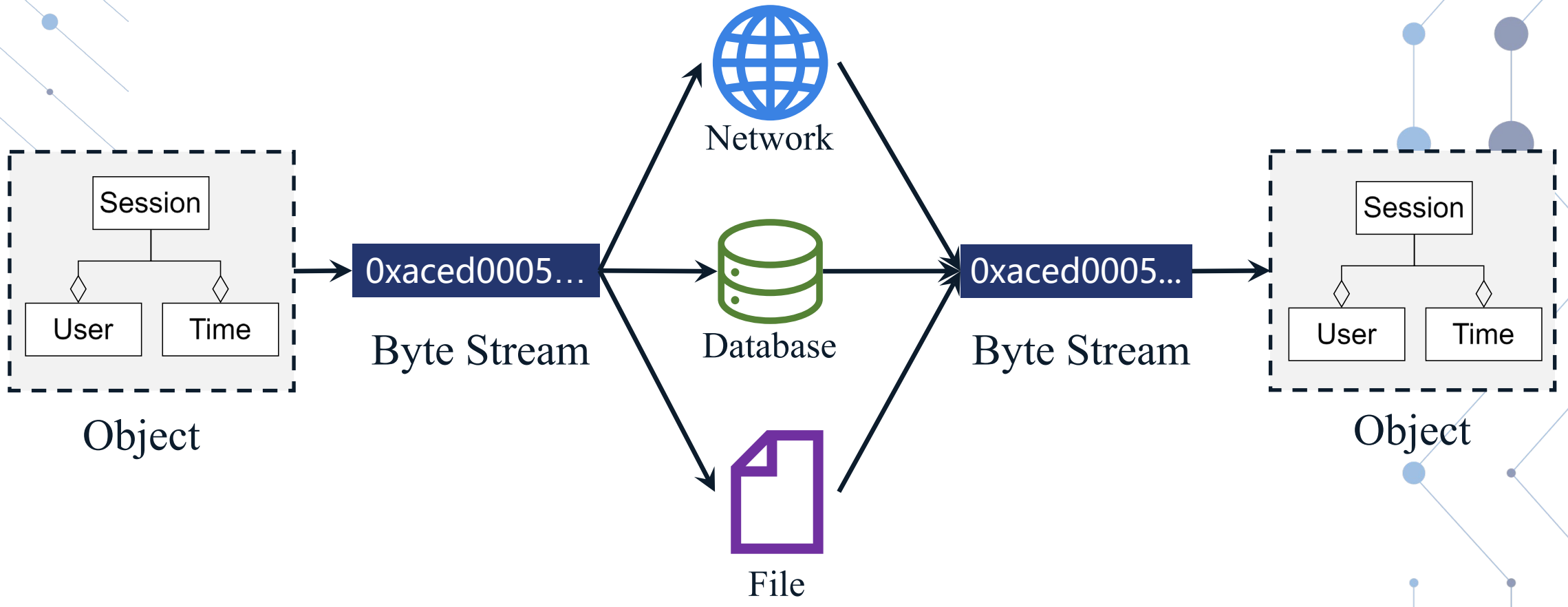
School of Software, Tsinghua University

NDSS SYMPOSIUM/2024

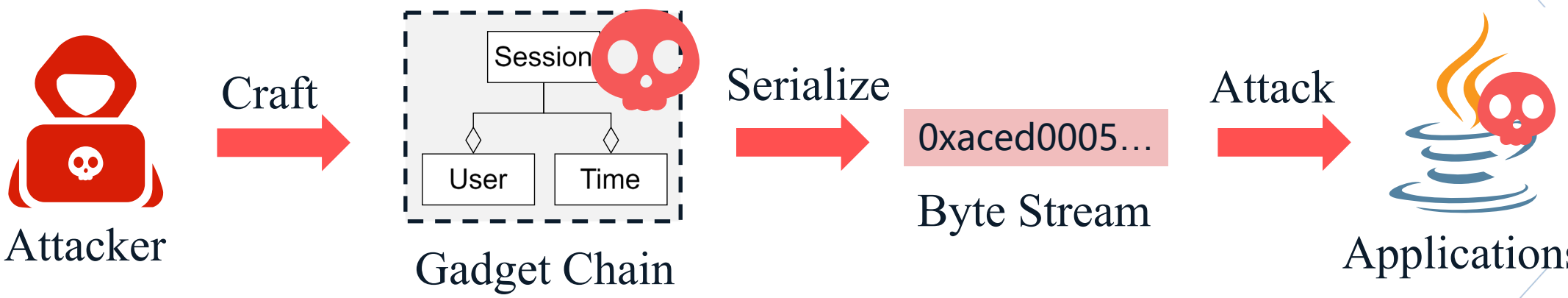
Presented by Internet Society

#NDSSSymposium2024

Deserialization Attack

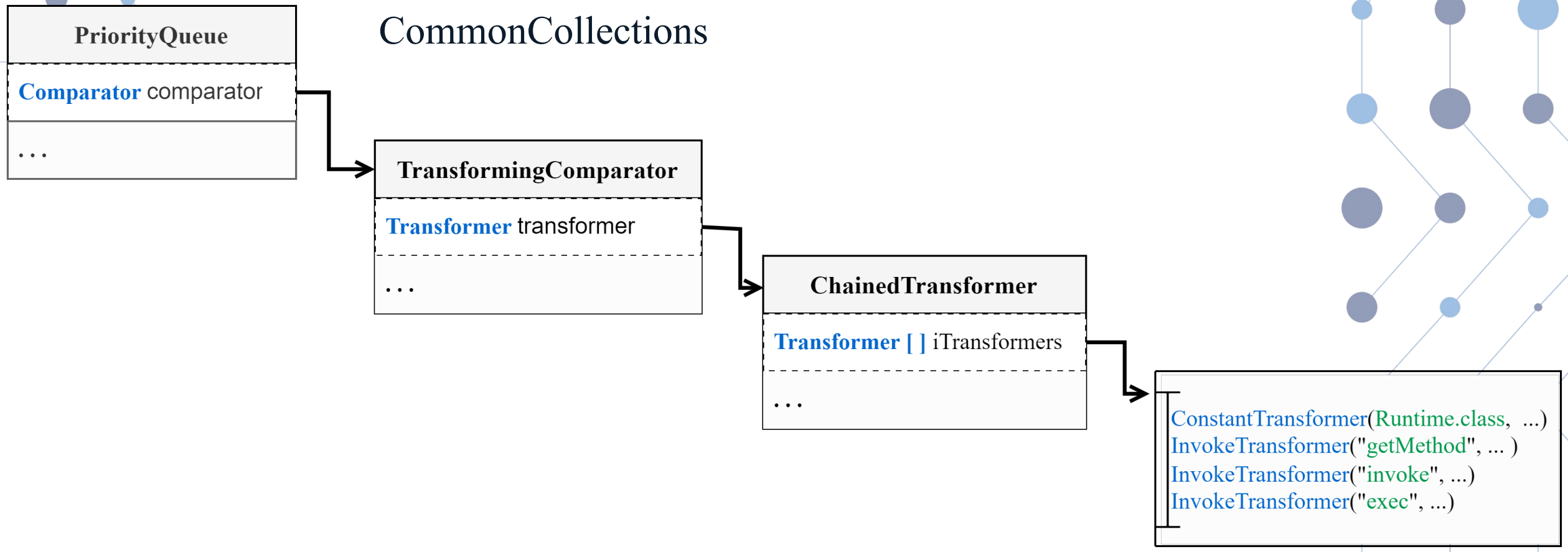


Deserialization Attack-Scenario

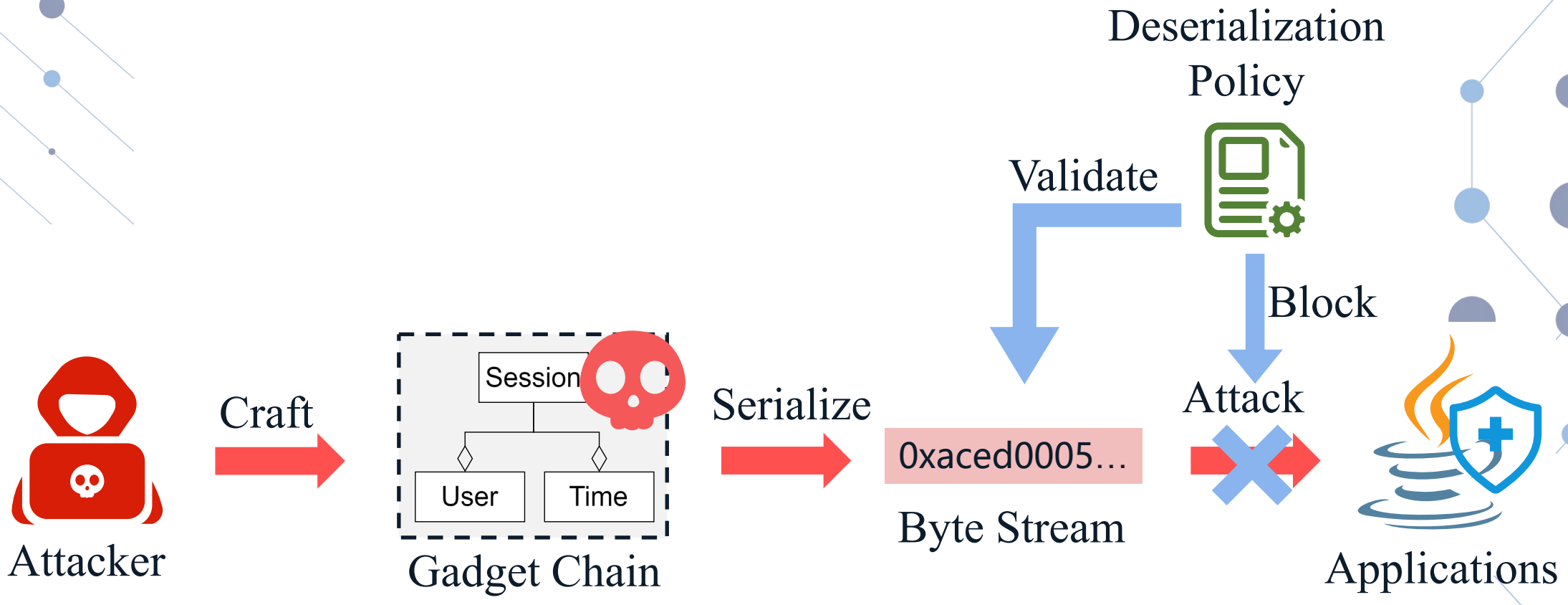


Deserialization Attack-Gadget Chain

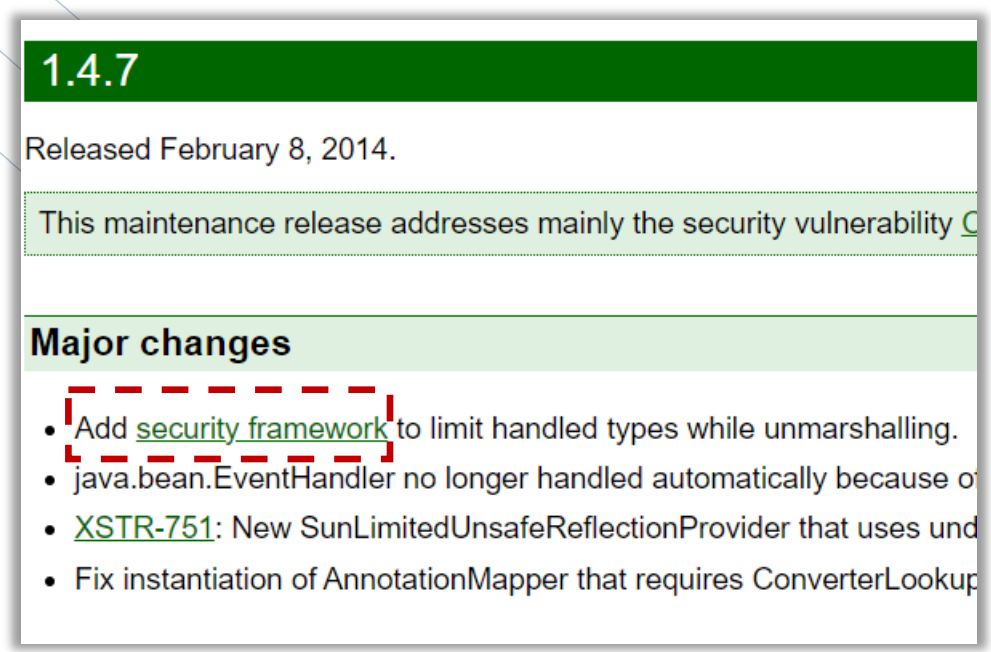
CommonCollections



Deserialization Defense



Deserialization Defense-Mechanism



1.4.7

Released February 8, 2014.

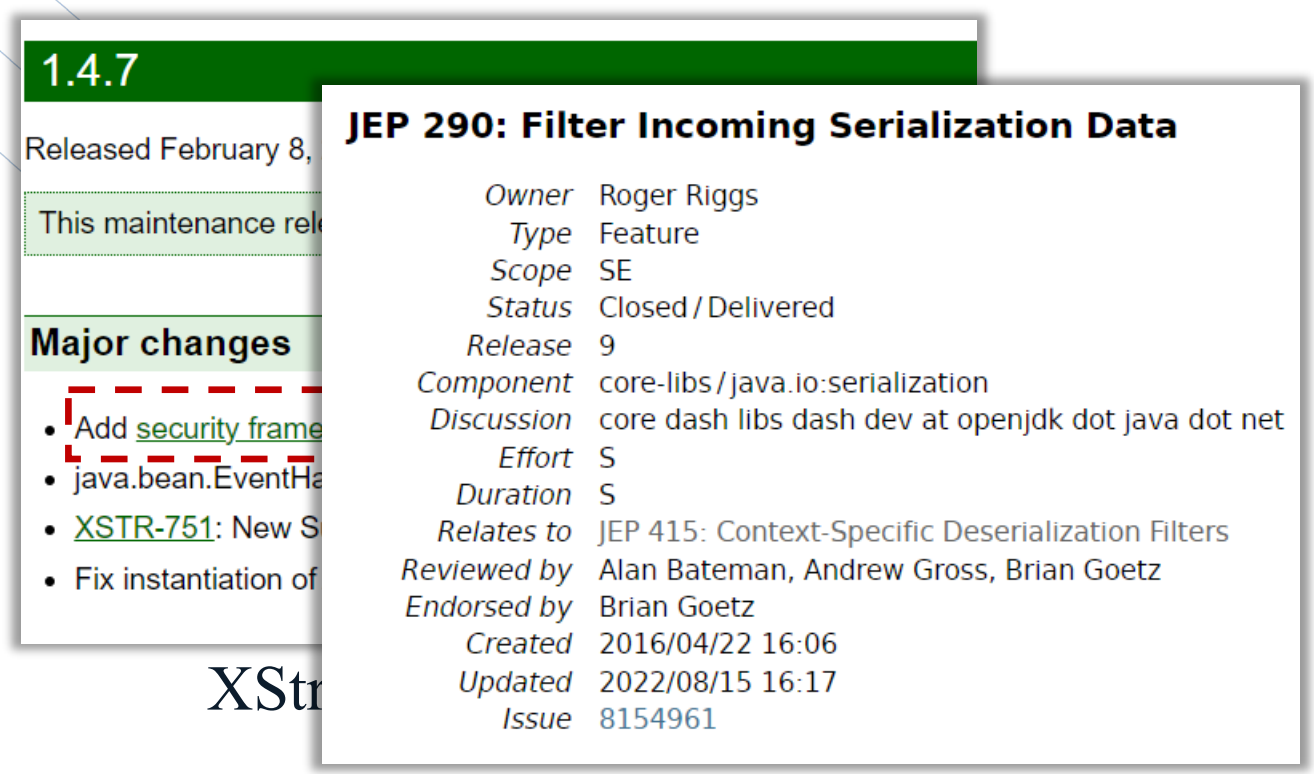
This maintenance release addresses mainly the security vulnerability C

Major changes

- Add security framework to limit handled types while unmarshalling.
- java.bean.EventHandler no longer handled automatically because o
- XSTR-751: New SunLimitedUnsafeReflectionProvider that uses und
- Fix instantiation of AnnotationMapper that requires ConverterLookup

XStream, 2014

Deserialization Defense-Mechanism



XStr

JEP 290, 2016

Deserialization Defense-Mechanism

XStr

FastJson, 2016

Motivation Example

Ofbiz

```
1  public final class SafeObjectInputStream extends ObjectInputStream {
2      protected Class<?> resolveClass(ObjectStreamClass classDesc)
                              throws IOException, ClassNotFoundException {
3          String className = classDesc.getName();
4          // BlockList exploits; eg: don't allow RMI here
5          if (className.contains("java.rmi.server")) {
6              Debug.logWarning("***Incompatible class***");
7              return null;
8          }
9          if (!whitelistPattern.matcher(className).find()) {
10             Debug.logWarning("***Incompatible class***");
11             throw new ClassCastException("Incompatible class");
12         }
13         return ObjectType.loadClass(classDesc.getName());
14     }
15 }
```

SafeObjectInputStream.java

Motivation Example

Patch for
(1) CVE-2019-0189

```
"byte\\[\\]", "foo", "\\[Z", "\\[B",
"\\[S", "\\[I", "\\[J", "\\[F", "\\[D", "\\[C",
"SerializationInjector",
"java\..*",
"sun\.util\.calendar\..*",
"org\.apache\.ofbiz\..*",
"org\.codehaus\.groovy\.runtime\.GStringImpl",
"groovy\.lang\.GString",
```

Allowlist

Ofbiz

```
1  public final class SafeObjectInputStream extends ObjectInputStream {
2      protected Class<?> resolveClass(ObjectStreamClass classDesc)
                          throws IOException, ClassNotFoundException {
3          String className = classDesc.getName();
4          // BlockList exploits; eg: don't allow RMI here
5          if (className.contains("java.rmi.server")) {
6              Debug.logWarning("***Incompatible class***");
7              return null;
8          }
9          if (!whitelistPattern.matcher(className).find()) {
10             Debug.logWarning("***Incompatible class***");
11             throw new ClassCastException("Incompatible class");
12         }
13         return ObjectType.loadClass(classDesc.getName());
14     }
15 }
```

SafeObjectInputStream.java

Motivation Example

(2) CVE-2021-26295

```
"byte\\[\\]", "foo", "\\[Z", "\\[B",
"\\[S", "\\[I", "\\[J", "\\[F", "\\[D", "\\[C",
"SerializationInjector",
"java\..*",
"sun\.util\.calendar\..*",
"org\.apache\.ofbiz\..*",
"org\.codehaus\.groovy\.runtime\.GStringImpl",
"groovy\.lang\.GString",
```

blocklist patch

Ofbiz

```
1  public final class SafeObjectInputStream extends ObjectInputStream {
2      protected Class<?> resolveClass(ObjectStreamClass classDesc)
                          throws IOException, ClassNotFoundException {
3          String className = classDesc.getName();
4          // BlockList exploits: eg: don't allow RMI here
5          if (className.contains("java.rmi.server")) {
6              Debug.logWarning("***Incompatible class***");
7              return null;
8          }
9          if (!whitelistPattern.matcher(className).find()) {
10             Debug.logWarning("***Incompatible class***");
11             throw new ClassCastException("Incompatible class");
12         }
13         return ObjectType.loadClass(classDesc.getName());
14     }
15 }
```

Patch for (1) CVE-2019-0189

SafeObjectInputStream.java

Motivation Example

Ofbiz

(3) CVE-2021-29200

(4) CVE-2021-30128

```
1  public final class SafeObjectInputStream extends ObjectInputStream {
2      protected Class<?> resolveClass(ObjectStreamClass classDesc)
                          throws IOException, ClassNotFoundException {
3          String className = classDesc.getName();
4          // BlockList exploits: eg: don't allow RMI here
5          if (className.contains("java.rmi.server")) {
6              Debug.logWarning("***Incompatible class***");
7              return null;
8          }                                   Patch for (2) CVE-2021-26295
9          if (!whitelistPattern.matcher(className).find()) {
10             Debug.logWarning("***Incompatible class***");
11             throw new ClassCastException("Incompatible class");
12         }                                   Patch for (1) CVE-2019-0189
13         return ObjectType.loadClass(classDesc.getName());
14     }
15 }
```

SafeObjectInputStream.java

Motivation Example

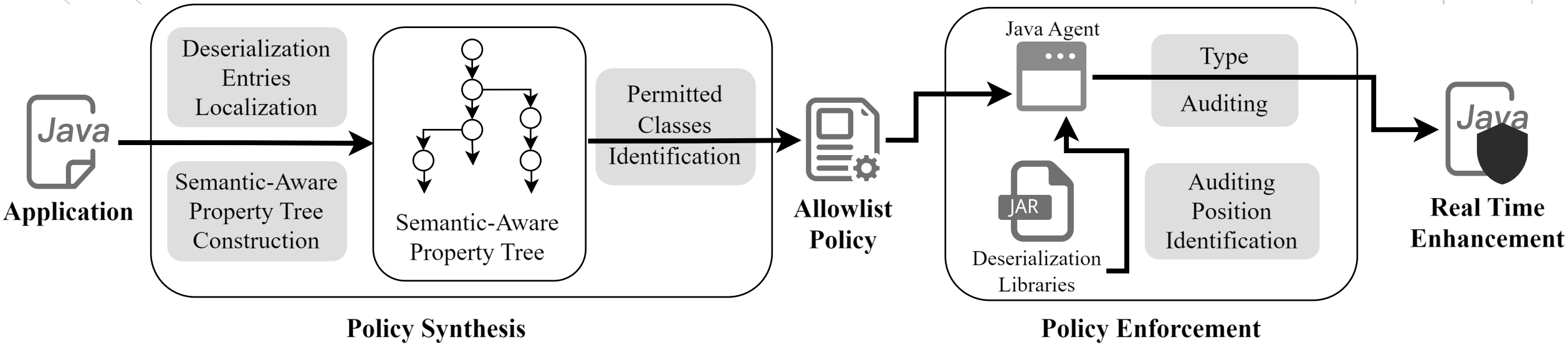
CVEs

- CVE-2019-0189
- CVE-2021-26295
- CVE-2021-29200
- CVE-2021-30128

Challenges

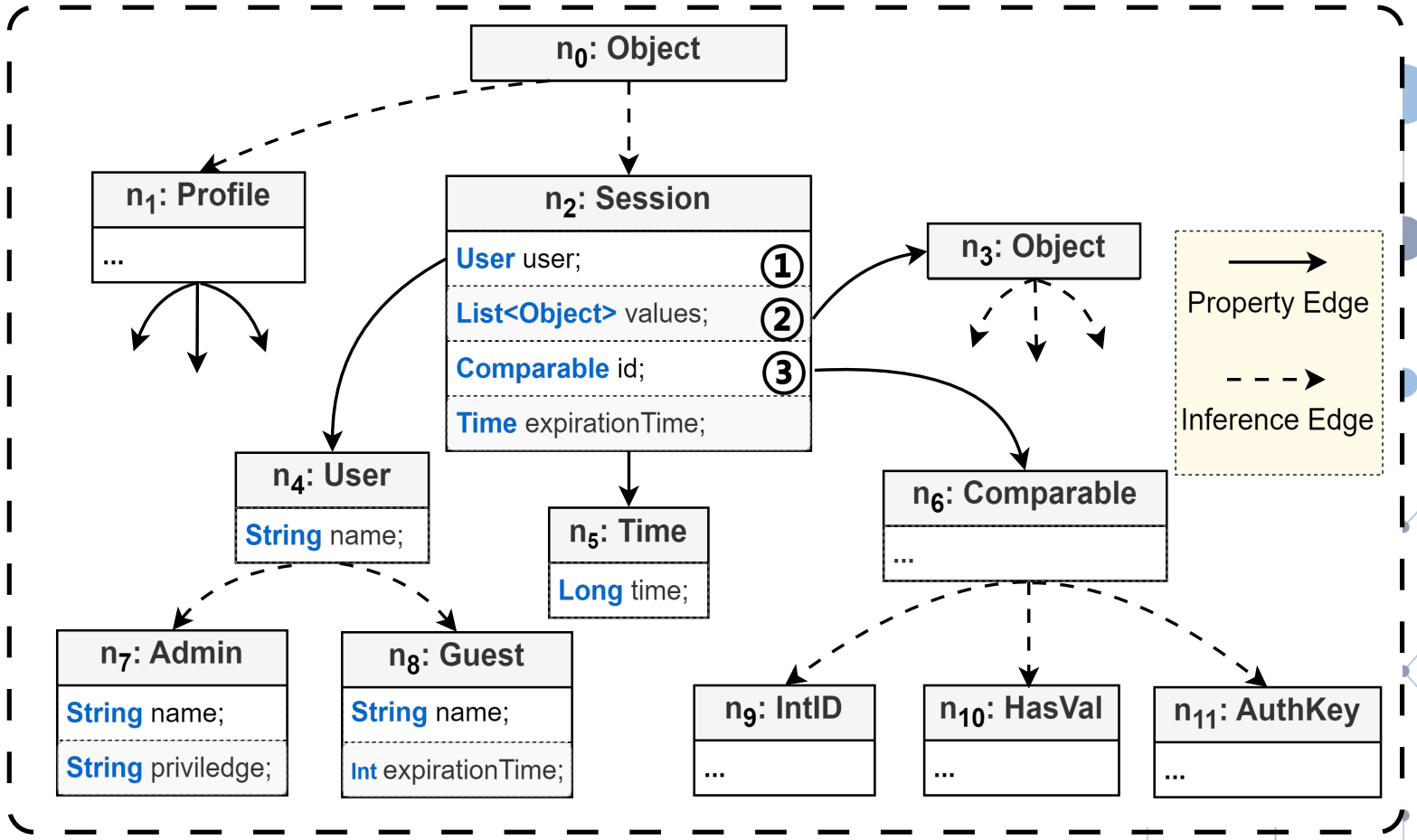
- **Policy Synthesis**
  - Demanding
  - Error-Prone
- **Policy Enforcement**
  - Various Libraries
  - Incorrect Implementation

DeseriGuard-Overview



Policy Synthesis-Semantic-Aware Property Tree (SAPT)

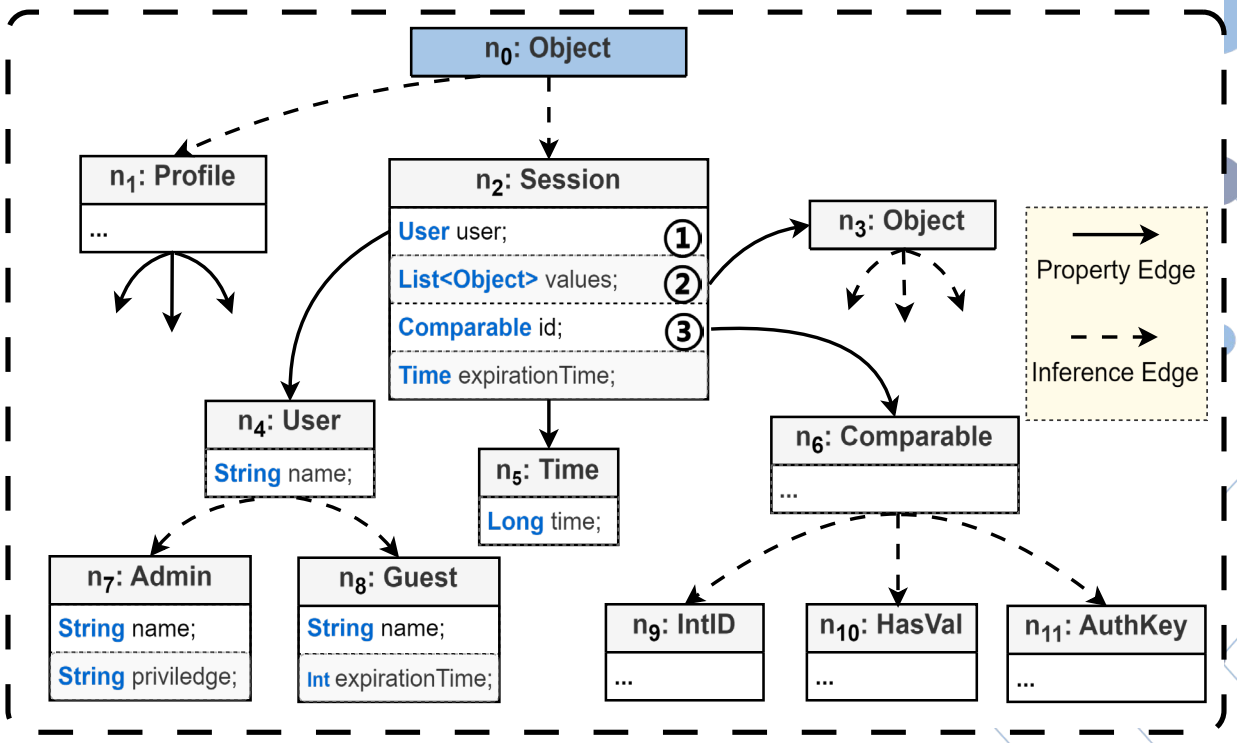
- **Nodes**
  - Java Classes
- **Edges**
  - Property Edges
  - Inference Edges



SAPT

Policy Synthesis-Root Node Identification

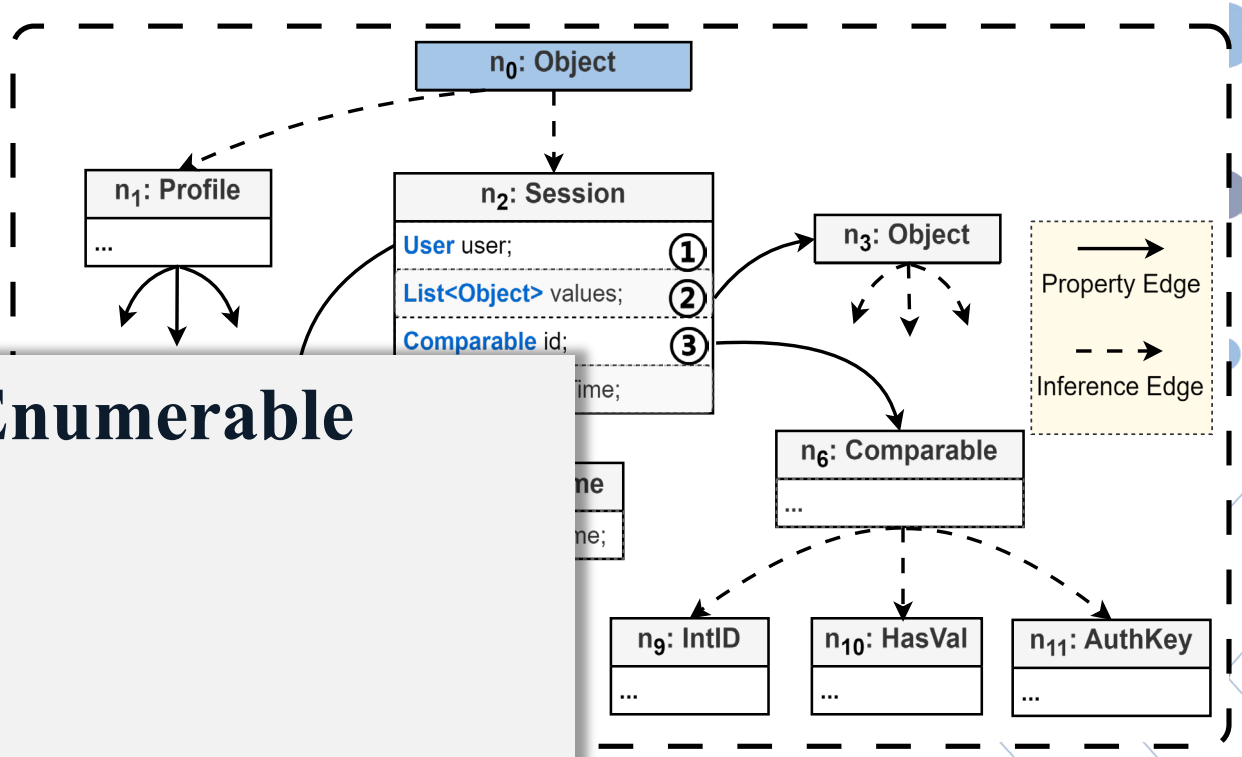
```
try{
    ObjectInputStream byteInputStream =
            new ByteArrayInputStream(serData);
    ObjectInputStream objIn =
            new ObjectInputStream(byteInputStream);
    obj = objIn.readObject();
} catch (Exception e) {
    e.printStackTrace();
    return null;}
if(obj instanceof profile){
    session = createSession((Profile) obj)
}else{
    session = (Session)obj;}
```



Semantic-Aware Property Tree (SAPT)

Policy Synthesis-Root Node Identification

```
try{
    ObjectInputStream byteInputStream =
            new ByteArrayInputStream(serData);
    ObjectInputStream objIn =
            new ObjectInputStream(byteInputStream);
    obj = objIn.readObject();
} catch (Exception e) {
    e.printStackTrace();
    return null;}
if(obj instanceof profile
    session = createSessi
}else{
    session = (Session)ob
```



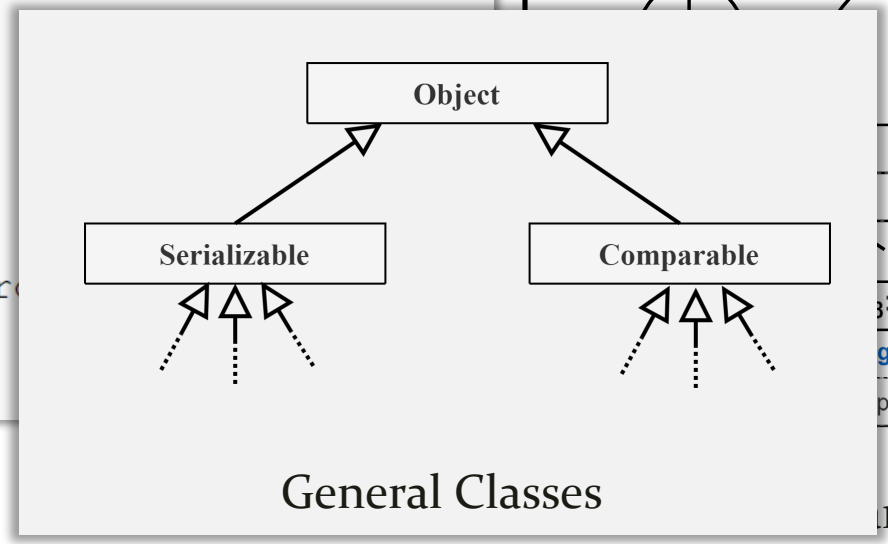
e Property Tree (SAPT)

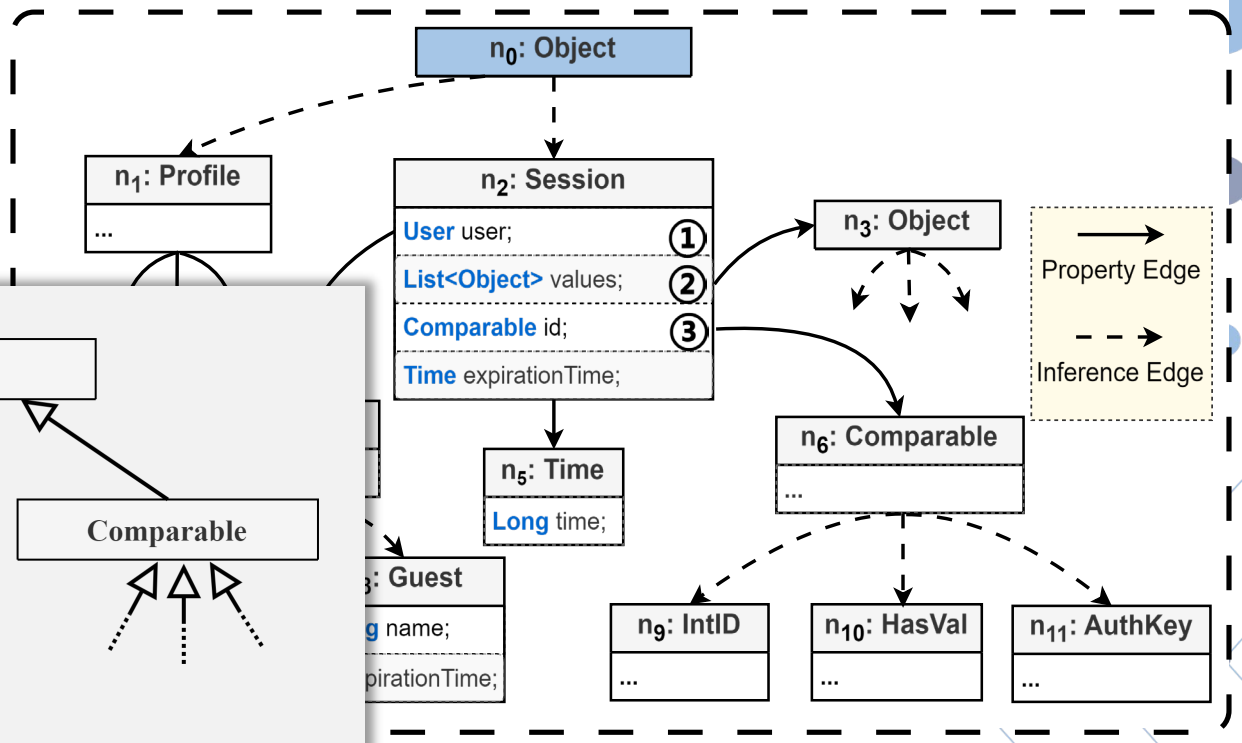
Entries are Enumerable

- ObjectInputStream
  - readObject
- XStream
  - fromXML
- ...

Policy Synthesis-Root Node Identification

```
try{
    ObjectInputStream byteInputStream =
            new ByteArrayInputStream(serData);
    ObjectInputStream objIn =
            new ObjectInputStream(byteInputStream);
    obj = objIn.readObject();
} catch (Exception e) {
    e.printStackTrace();
    return null;}
if(obj instanceof profile){
    session = createSession((Pr
}else{
    session = (Session)obj;}
```

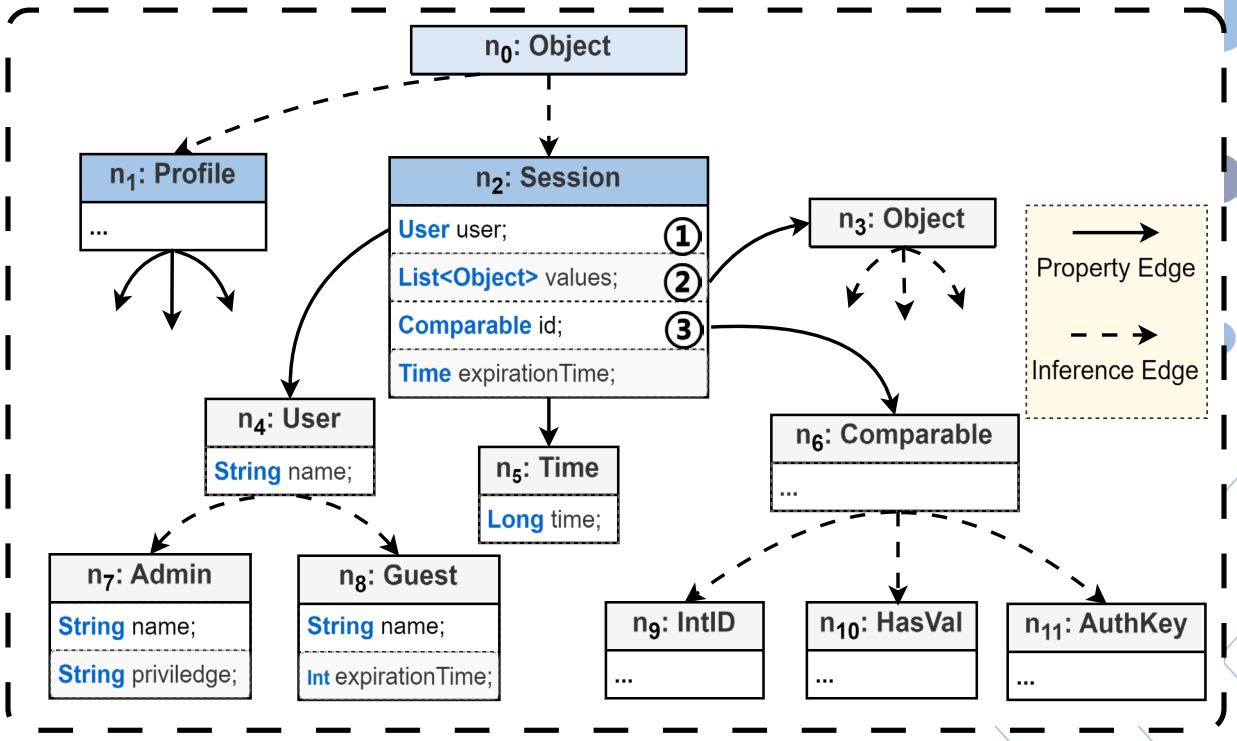




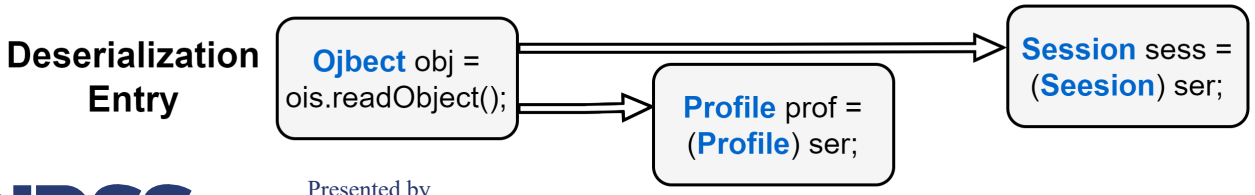
ntic-Aware Property Tree (SAPT)

Policy Synthesis-Root Node Identification

```
try{
    ObjectInputStream byteInputStream =
            new ByteArrayInputStream(serData);
    ObjectInputStream objIn =
            new ObjectInputStream(byteInputStream);
    obj = objIn.readObject();
} catch (Exception e) {
    e.printStackTrace();
    return null;}
if(obj instanceof profile){
    session = createSession((Profile) obj)
}else{
    session = (Session)obj;}
```

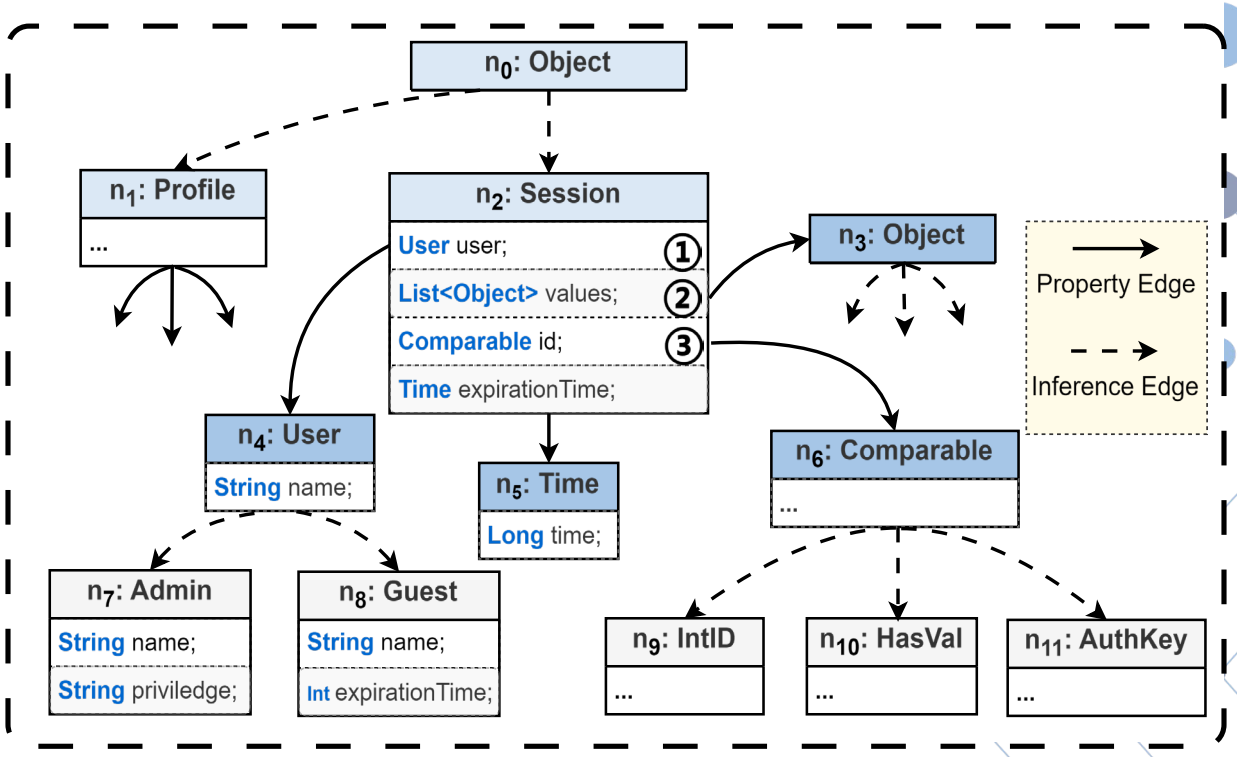


Semantic-Aware Property Tree (SAPT)



Policy Synthesis-Property Edges Connection

```
class Session{  // Session.java
    private User user;
    private List<Object> values;
    private Time expirationTime;
    public Comparable id;
    public Object getValue(int index){
        return values.get(i);
    }
}
```

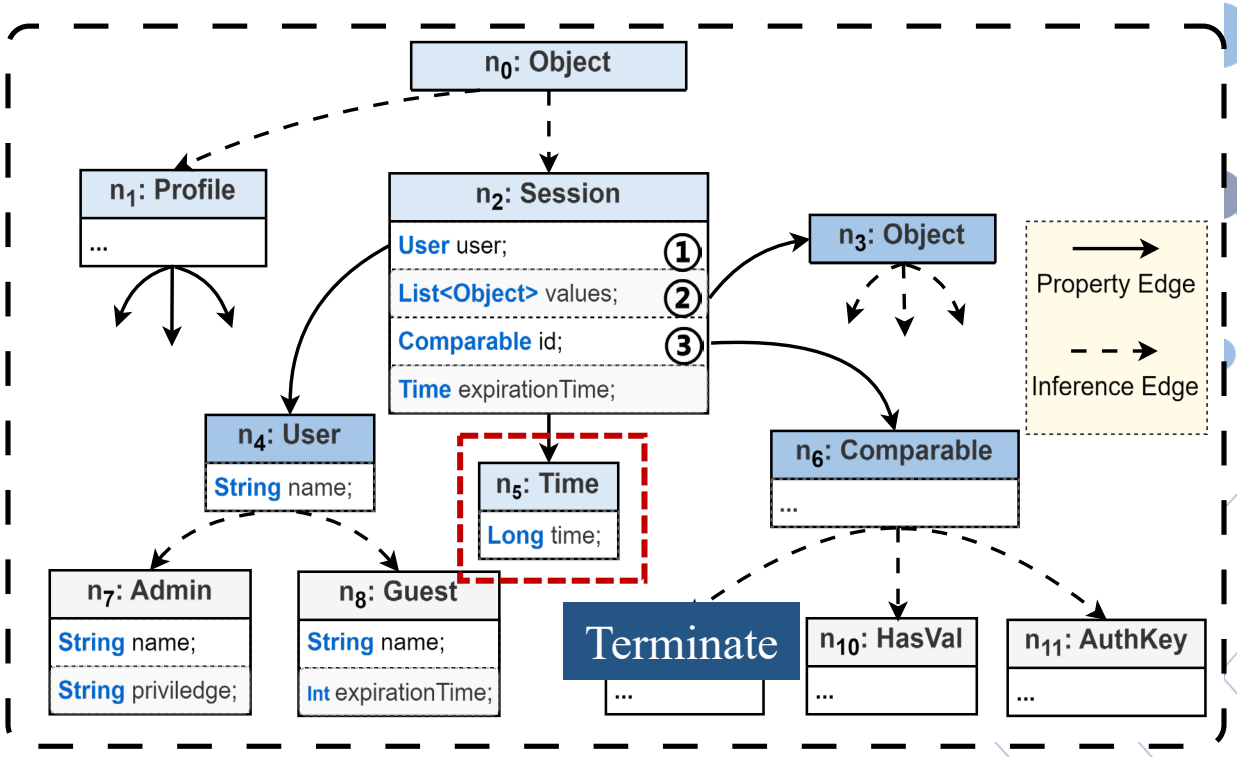


Semantic-Aware Property Tree (SAPT)

Policy Synthesis-Property Edges Connection

Basic Types

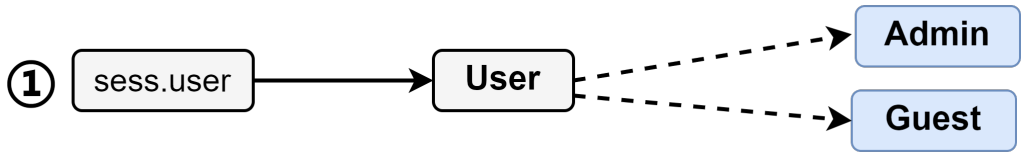
- **Primitive Types**
  - Int
  - Long
  - Boolean
  - ...
- **String**

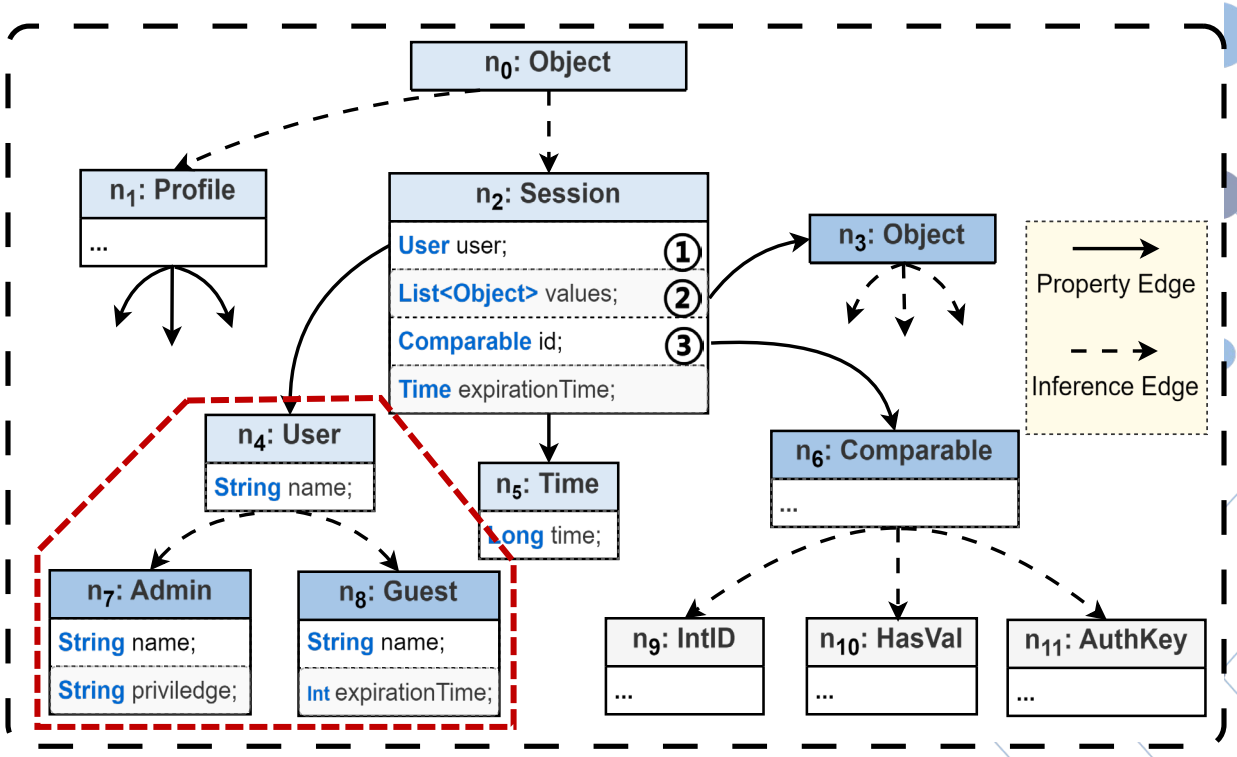


Semantic-Aware Property Tree (SAPT)

Policy Synthesis-Inference Edges Solvement

```
class User{    // User.java
    private String name;
}
class Admin extends User{   // Admin.java
    private String privilege;
}
class Guest extends User{   // Guest.java
    int expirationTime;
}
```

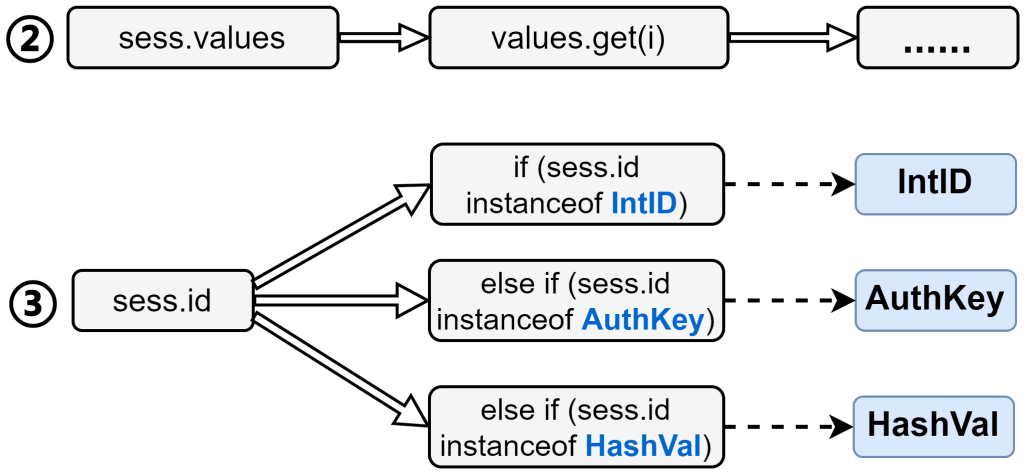


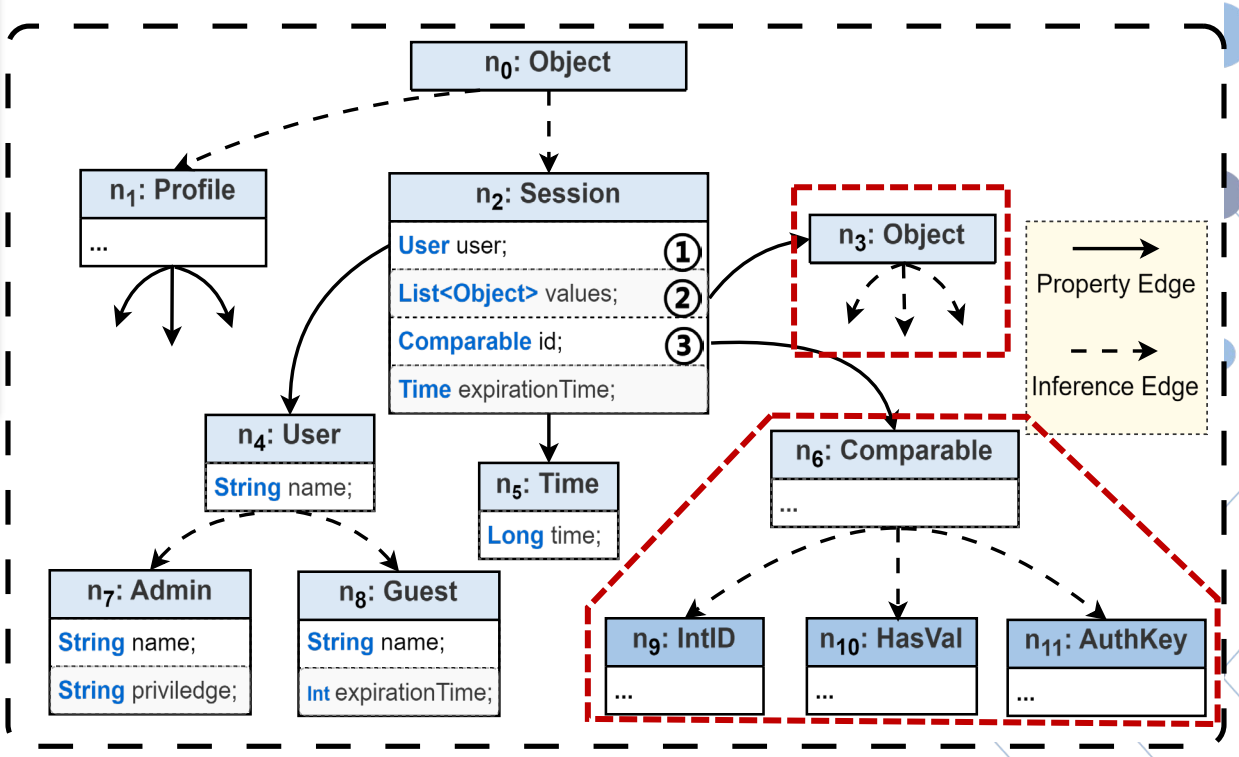


Semantic-Aware Property Tree (SAPT)

Policy Synthesis-Inference Edges Solvement

```
Comparable userIndex = Session.id;
int priority;
if (userIndex instanceof IntID){
    priority = ((IntID)userIndex).toInt();
}else if (userIndex instanceof AuthKey){
    priority = getPriority((AuthKey)userIndex);}
else if (userIndex instanceof HashVal){
    priority = priorityMap.get((HashVal)userIndex);
```

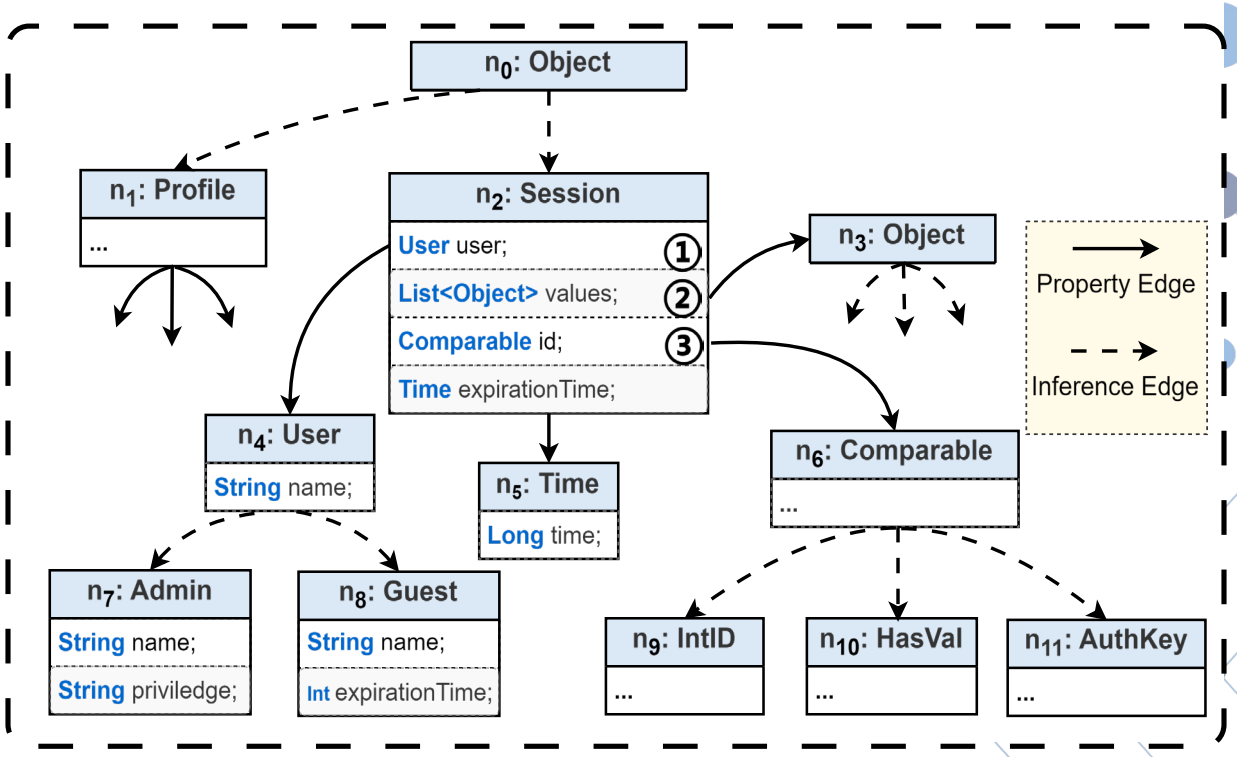




Semantic-Aware Property Tree (SAPT)

Policy Synthesis-Inference Edges Solvement

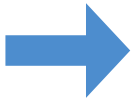
```
"example.app.Session",   "example.app.IntID",
"example.app.User",      "example.app.AuthKey",
"example.app.HashVal",   "example.app.Time",
```



Semantic-Aware Property Tree (SAPT)

Policy Enforcement

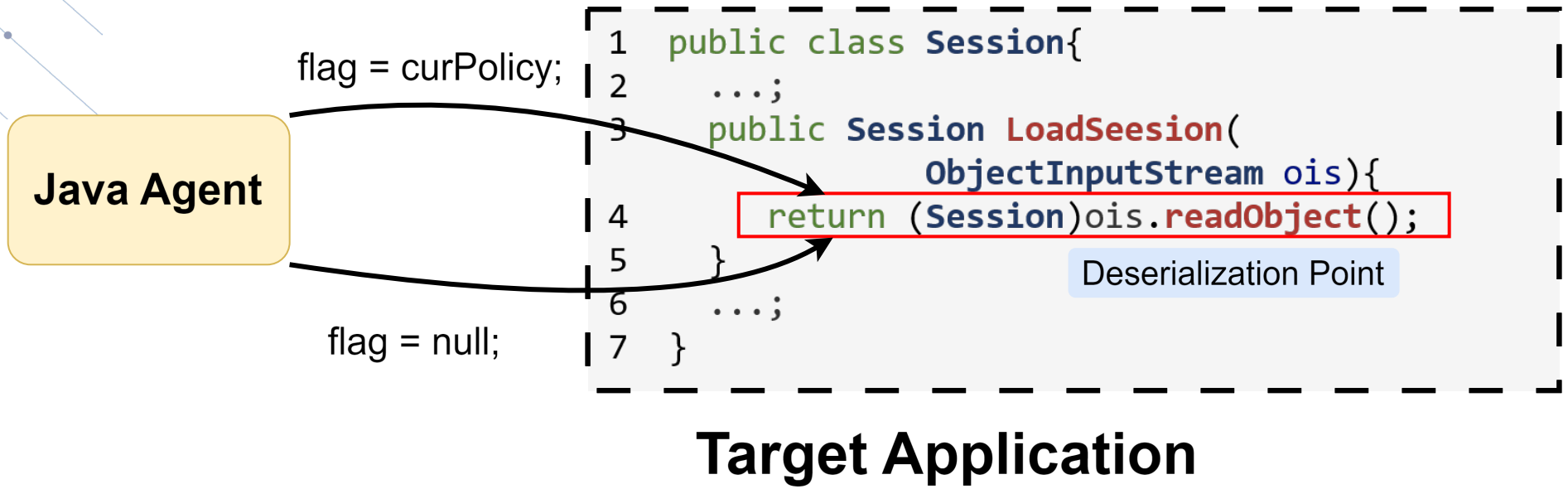
- **Various Deserialization Libraries**
  - ObjectInputStream
  - XStream
  - FastJson
  - ...



Automatic Policy Enforcement with Java Agent Instrumentation

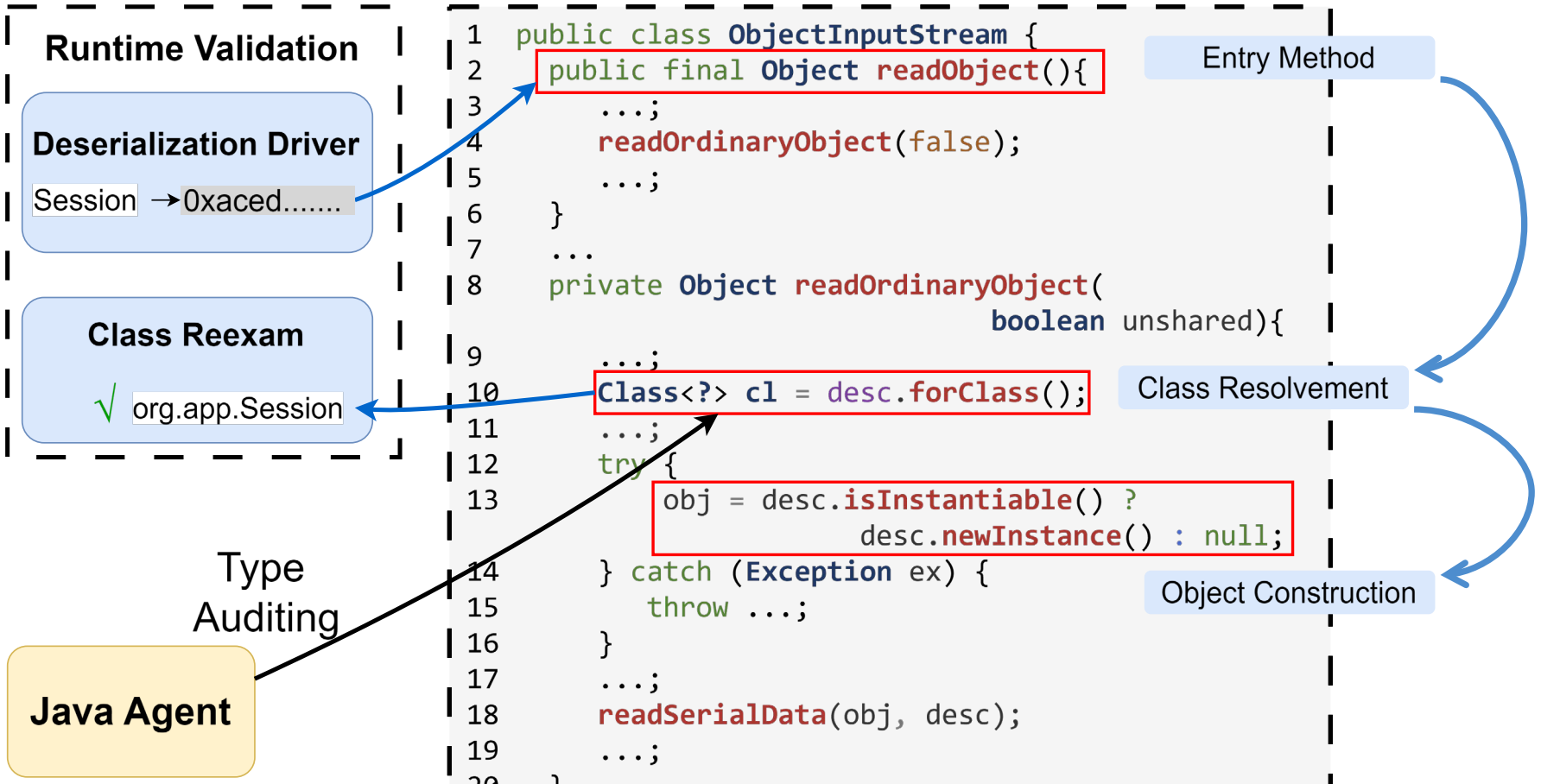
- Activator
- Auditor

Policy Enforcement-Activator



Policy Enforcement-Auditor

Policy Enforcement-Auditor



Evaluation-Setting Up

- **Real World Vulnerabilities**
  - 12 vulnerabilities
- **Developer-Designed Policies**
  - 109 policies from 40 projects
  - collected from GitHub with more than 100 stars

Evaluation-Real World Vulnerabilities

Application	Label	LoC	Classes	Resist	False Alarms
Apereo CAS-4.1.5	CAS 4.1.x	1.86M	49.12K	✓	No
Richfaces-4.3.3	CVE-2013-2165	57.5K	5.74K	✓	No
Jenkins-1.637	CVE-2015-8103	643.07K	23.19K	✓	No
Shiro-1.2.4	CVE-2016-4437	82.85K	5.60K	✓	No
Jenkins-2.46.1	CVE-2017-1000353	646.45K	18.67K	✓	No
Olingo-4.6.0	CVE-2019-17556	150.82K	13.81K	✓	No
Tomcat-10.0.0	CVE-2020-9484	171.89K	17.50K	✓	No
Ofbiz-17.02.03	CVE-2020-9496	2.00M	25.87K	✓	No
Ofbiz-17.12.05	CVE-2021-26295	2.79M	30.71K	✓	No
Ofbiz-17.12.06	CVE-2021-29200	2.09M	27.51K	✓	No
Ofbiz-17.12.06	CVE-2021-30128	2.09M	27.51K	✓	No
Log4j-1.2.17	CVE-2022-23307	695.99K	2.91K	✓	No

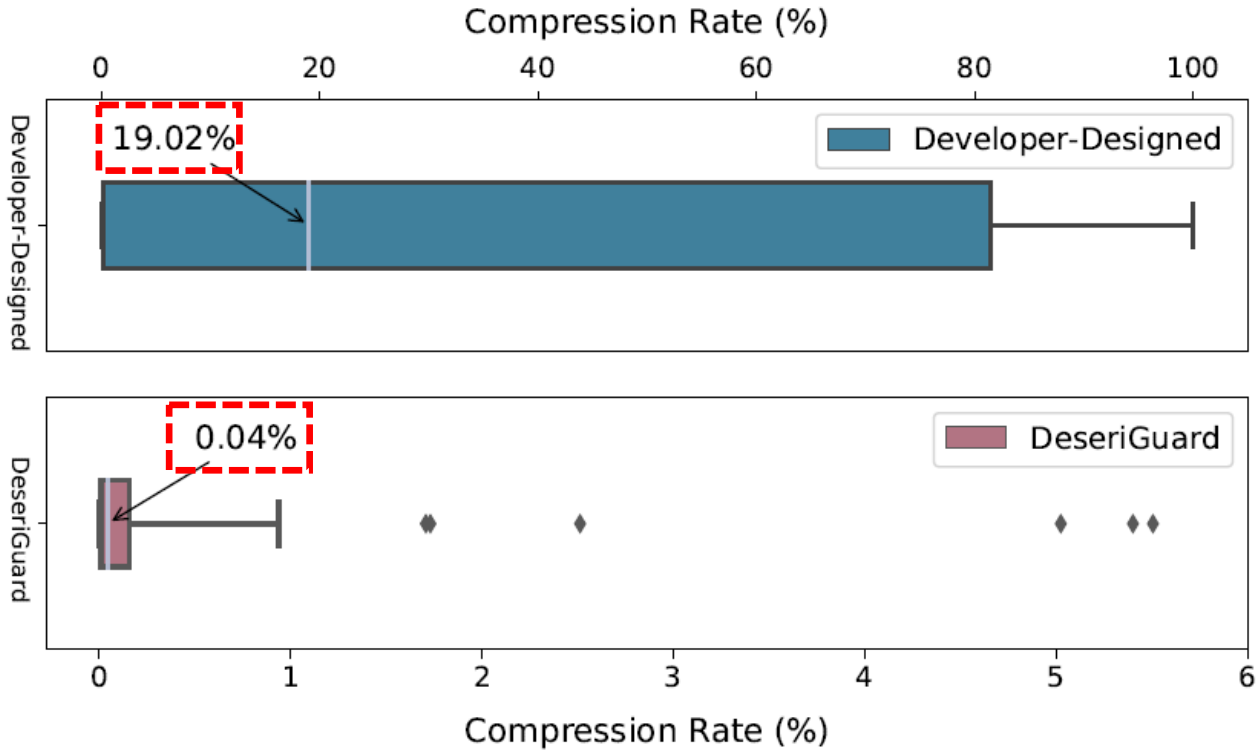
- 1.11M LoC
- 20.68K Classes
- 100% Defense Rate
- No False Alarm

Evaluation-Real World Vulnerabilities

Application	Label	LoC	Classes	Resist	Policy Rules	Permitted Classes
Apereo CAS-4.1.5	CAS 4.1.x	1.86M	49.12K	✓	4	4
Richfaces-4.3.3	CVE-2013-2165	57.5K	5.74K	✓	1	1
Jenkins-1.637	CVE-2015-8103	643.07K	23.19K	✓	1	1
Shiro-1.2.4	CVE-2016-4437	82.85K	5.60K	✓	79	79
Jenkins-2.46.1	CVE-2017-1000353	646.45K	18.67K	✓	28	161
Olingo-4.6.0	CVE-2019-17556	150.82K	13.81K	✓	33	58
Tomcat-10.0.0	CVE-2020-9484	171.89K	17.50K	✓	14	23
Ofbiz-17.02.03	CVE-2020-9496	2.00M	25.87K	✓	413	935
Ofbiz-17.12.05	CVE-2021-26295	2.79M	30.71K	✓	623	1342
Ofbiz-17.12.06	CVE-2021-29200	2.09M	27.51K	✓	392	905
Ofbiz-17.12.06	CVE-2021-30128	2.09M	27.51K	✓	392	905
Log4j-1.2.17	CVE-2022-23307	695.99K	2.91K	✓	20	28

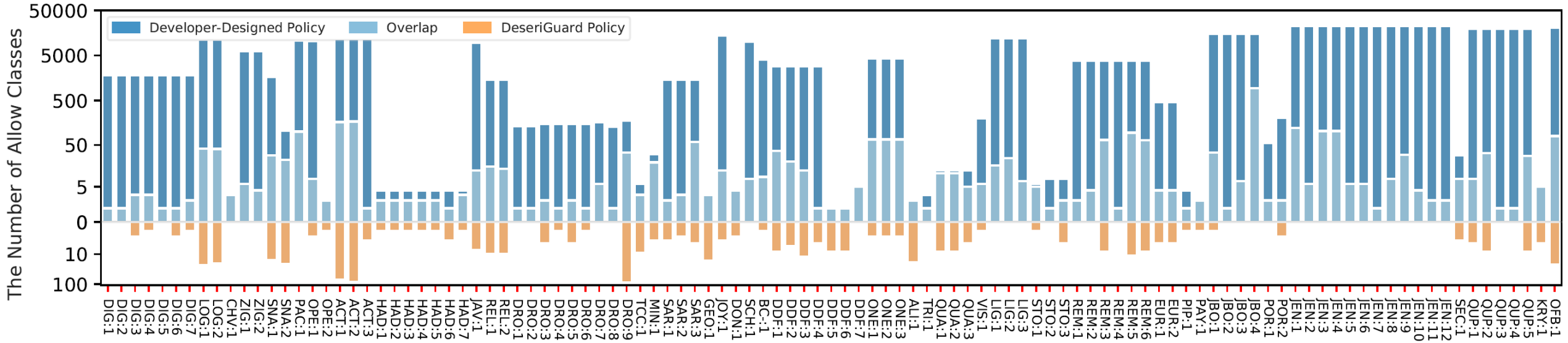
- 1~623 rules
- 1~1342 permitted classes

Evaluation-Developer-Designed Policies



$$compression\ rate = \frac{permitted\ classes}{total\ classes}$$

Evaluation-Developer-Designed Policies



- 99.12% fewer classes
- 1~85 additional classes

Conclusion

- DeseriGuard automatically synthesizes allowlist policies for Java applications
  - Manually formulating is challenging
- DeseriGuard automatically enforces policies for various deserialization libraries
  - Enforcement position location is tough
- DeseriGuard mitigates 12 vulnerabilities and permits 99.12% fewer classes on 109 deserialization entries

Thank you for the listening!

quanzh98@gmail.com

Thank Deng Feng Fund.

Evaluation-SOTA Approaches

Application	GadgetInspector		Ysoserial
	Gadget Chains	Resist	Resist
Apereo CAS-4.1.5	10	✓	✓
RichFaces-4.3.3	3	✓	✓
Jenkins-1.637	20	✓	✓
Shiro-1.2.4	3	✓	✓
Jenkins-2.46.1	20	✓	✓
Olingo-4.6.0	16	✓	✓
Tomcat-10.0.0	15	✓	✓
Ofbiz-17.02.03	18	✓	✓
Ofbiz-17.12.05	19	✓	✓
Ofbiz-17.12.06	19	✓	✓
Ofbiz-17.12.06	19	✓	✓
Log4j-1.2.17	2	✓	✓

Gadget Chain Mining

Evaluation-SOTA Approaches

Applications	Defense Performance		False Alarm	
	DESERIGUARD	Trusted	DESERIGUARD	Trusted
Apereo CAS-4.1.5	✓	✓	No	No
RichFaces-4.3.3	✓	✓	No	No
Jenkins-1.637	✓	✓	No	No
Shiro-1.2.4	✓	✓	No	Yes
Jenkins-2.46.1	✓	✓	No	Yes
Olingo-4.6.0	✓	✓	No	Yes
Tomcat-10.0.0	✓	✓	No	Yes
Ofbiz-17.02.03	✓	✓	No	Yes
Ofbiz-17.12.05	✓	✓	No	Yes
Ofbiz-17.12.06	✓	✓	No	Yes
Ofbiz-17.12.06	✓	✓	No	Yes
Log4j-1.2.17	✓	✓	No	No

Policy Learning

Evaluation-Real World Vulnerabilities

Application	Before Runtime		Runtime	
	Analysis	Initialization	Auditing	Slowdown
Apereo CAS-4.1.5	10s	8.10ms	0.100ms	0.795%
Richfaces-4.3.3	6s	2.37ms	0.030ms	4.296%
Jenkins-1.637	21s	11.99ms	0.042ms	4.320%
Shiro-1.2.4	8s	15.54ms	0.032ms	3.656%
Jenkins-2.46.1	22s	84.71ms	0.034ms	2.931%
Olingo-4.6.0	46s	3.84ms	0.074ms	3.201%
Tomcat-10.0.0	39s	36.88ms	0.031ms	3.759%
Ofbiz-17.02.03	65s	15.54ms	0.017ms	0.388%
Ofbiz-17.12.05	69s	170.84ms	0.024ms	0.632%
Ofbiz-17.12.06	71s	100.75ms	0.021ms	0.966%
Ofbiz-17.12.06	70s	90.40ms	0.020ms	0.625%
Log4j-1.2.17	6s	11.99ms	0.032ms	0.443%
Average	36.1s	46.08ms	0.039ms	2.168%

Overhead