

44th IEEE Symposium on Security and Privacy

ODDFUZZ: Discovering Java Deserialization Vulnerability via Structure-Aware Directed Greybox Fuzzing

Sicong Cao¹, Biao He², Xiaobing Sun¹, Yu Ouyang², Chao Zhang³, Xiaoxue Wu¹,
Ting Su⁴, Lili Bo¹, Bin Li¹, Chuanlei Ma², Jiajia Li², and Tao Wei²

¹Yangzhou University

²Ant Group

³Tsinghua University

⁴East China Normal University



揚州大學
YANGZHOU UNIVERSITY



蚂蚁集团
ANT GROUP

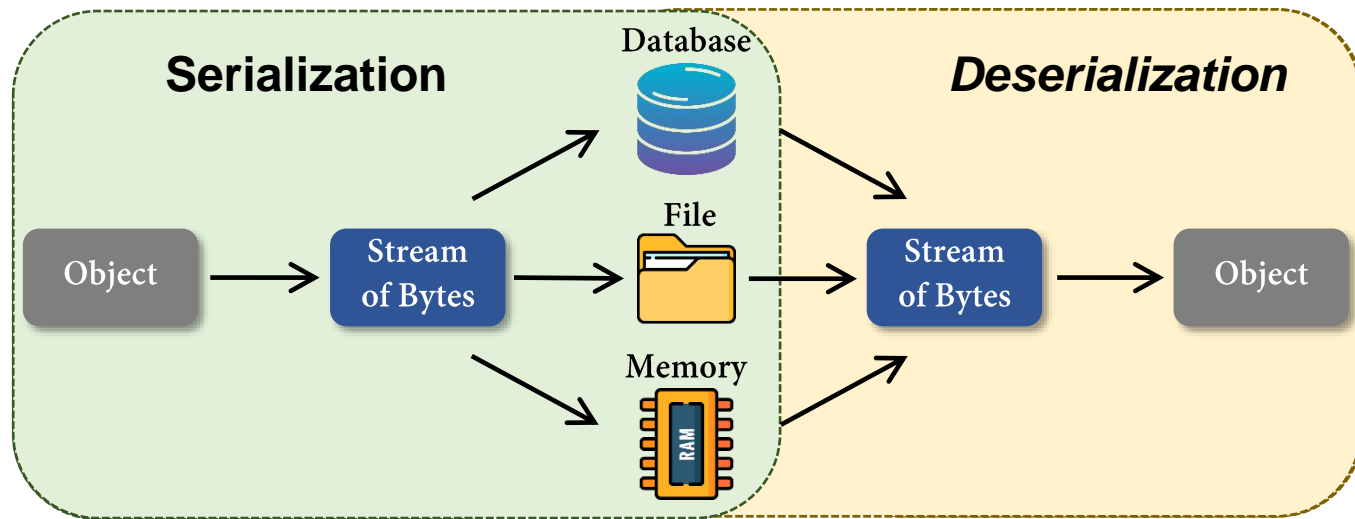


清華大學
Tsinghua University



華東師範大學
EAST CHINA NORMAL
UNIVERSITY

Java Deserialization



- Communication
- Caching
- Deep Copy
- Cross JVM Synchronization
- Persistence

ODD Vulnerability Attack

🔑 O --- Open

Arbitrary objects may be injected by adversaries, which breaks the traditional trust boundary of inter-process data transmission and introduces attack surfaces.

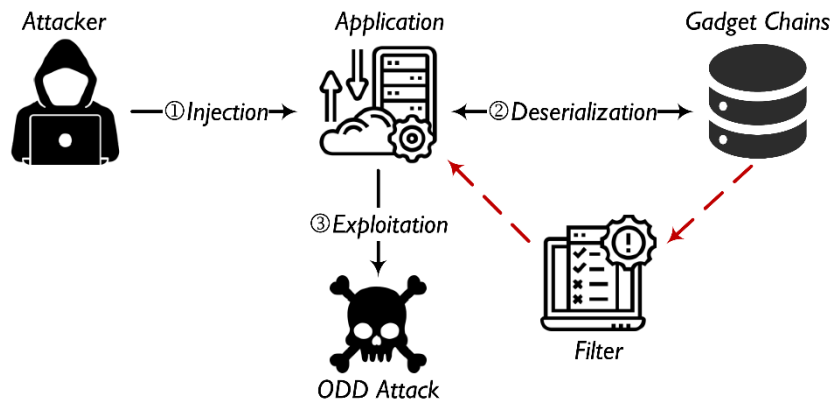
🔑 D --- Dynamic

The deserialization path of an injected serialized objects can be manipulated by abusing runtime polymorphic or other dynamic features.

🔑 D --- Deserialization

Magic methods get executed *automatically* by the deserializer, even before deserialization finishes!

**Controlling Data Types
=> Controlling Code!**



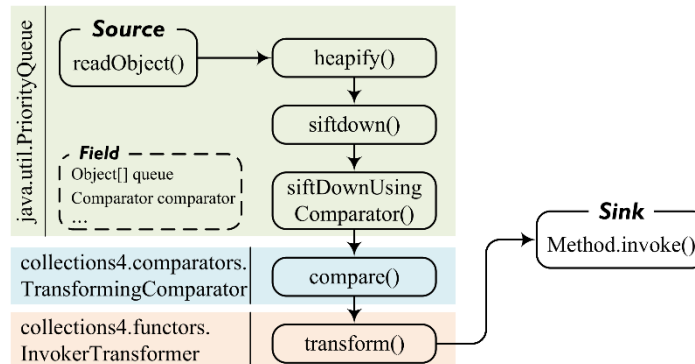
Threat Model

ODD Vulnerability Attack

```
1 public class PriorityQueue<E> implements Serializable {
2     transient Object[] queue;
3     private final Comparator<? super E> comparator;
4     private void readObject(java.io.ObjectInputStream s)
5         {...} /*Source*/
6     private void siftDown(int k, E x) { /*3rd Gadget*/
7         if (comparator != null)
8             siftDownUsingComparator(k, x);
9         ...
10    private void siftDownUsingComparator(int k, E x) { /*4th Gadget*/
11        if (right < size &&
12            comparator.compare((E) c, (E) queue[right]) > 0)
13            ...}}
14    /*org.apache.commons.collections4.comparators*/
15    public class TransformingComparator implements Comparator<E> {
16        private final Comparator<E> comparator;
17        public int compare(E o1, E o2) {
18            Object value1 = o1;
19            Object value2 = o2;
20            ...}}
21    /*org.apache.commons.collections4.functions*/
22    public class InvokerTransformer<I, O> implements Transformer<I, O>, Serializable {
23        public Object transform(Object input) { /*6th Gadget*/
24            Class<?> cls = input.getClass();
25            Method method = cls.getMethod(this.iMethodName,
26                                           this.iParamTypes);
27            return method.invoke(input, this.iArgs); /*Sink*/
28        }
29    }
```



Confirming whether there are exploitable gadget chains in the applications is more urgent!



queue = [obj1, obj2]
comparator = T1

T1 : TransformingComparator
transformer = I1

I1 : InvokerTransformer

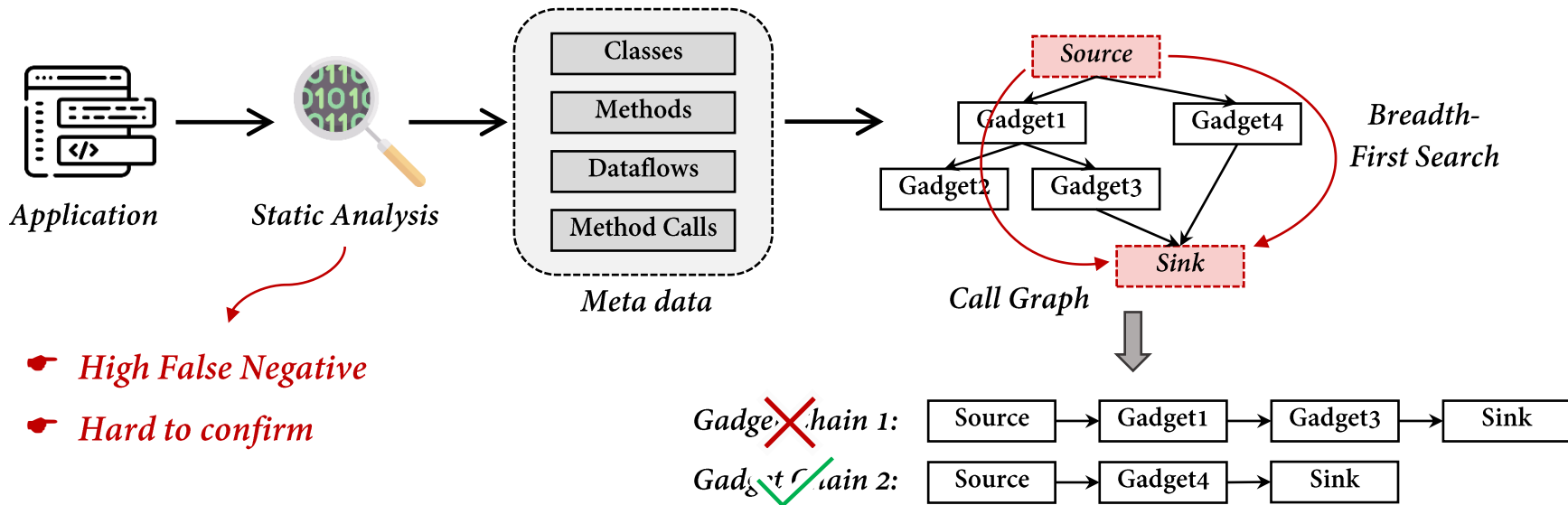
iMethodName = "MethodName"
iParamTypes = null
iArgs = null

Apache Commons Collections

Injection object

Existing Solutions

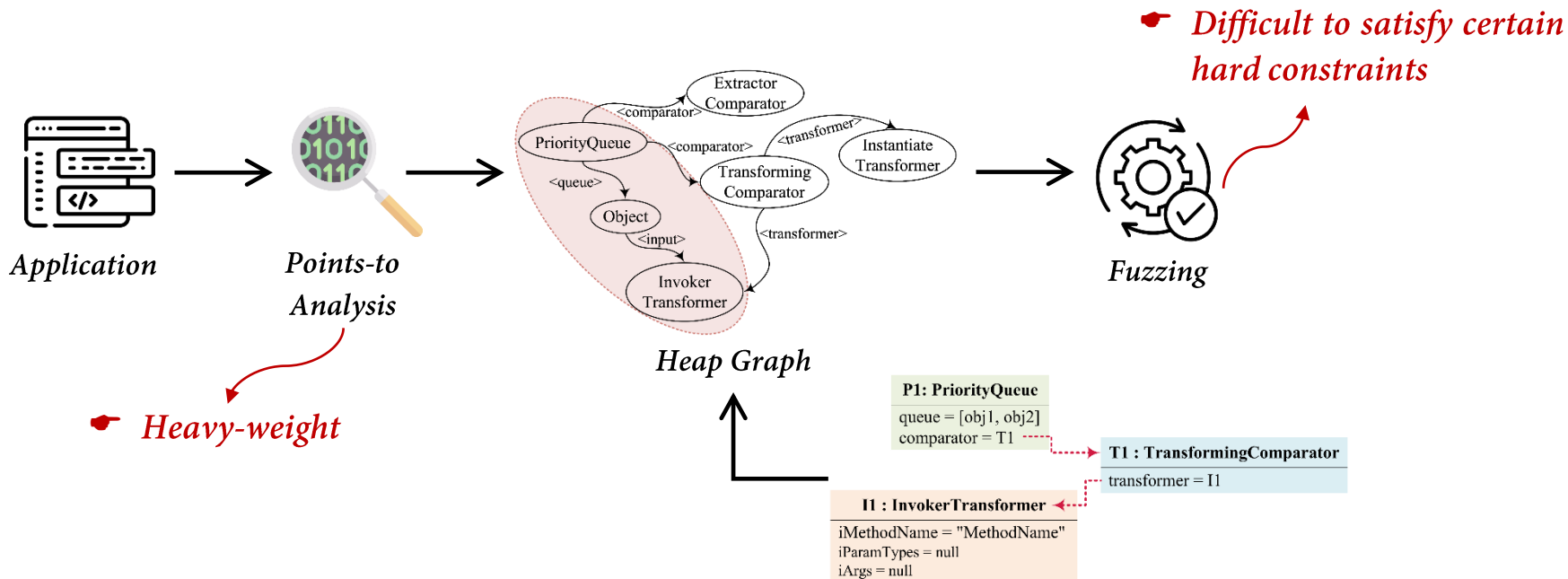
Gadget Inspector [BlackHat 2018]



¹I. Haken, “Automated discovery of deserialization gadget chains,” BlackHat USA, 2018.

Existing Solutions

SerHybrid [ASE 2022]

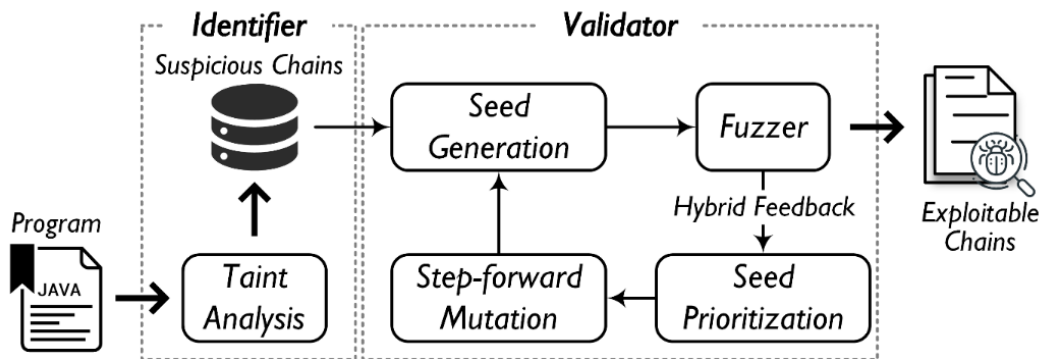


²S. Rasheed and J. Dietrich, "A hybrid analysis to detect java serialization vulnerabilities," IEEE/ACM International Conference on Software Engineering, 2020.

Our approach: ODDFUZZ

Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- ❑ *Summary*-based taint analysis + CHA
- ❑ *Structure-aware* seed generation
- ❑ *Hybrid* feedback-based prioritization
- ❑ *Step-forward* mutation rules



Our approach: ODDFUZZ

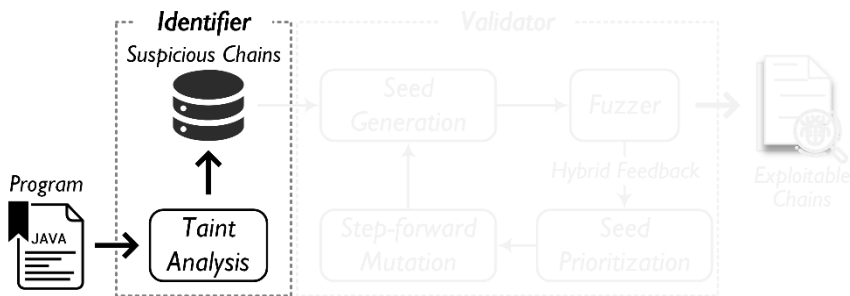
Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- ❑ *Summary*-based taint analysis + CHA
- ❑ *Structure-aware* seed generation
- ❑ *Hybrid* feedback-based prioritization
- ❑ *Step-forward* mutation rules



How to *effectively* and *efficiently* identify potential gadget chains?

- Conduct BFS based on the pre-computed method summaries to chain available gadgets.
- Perform Class Hierarchy Analysis (CHA) on the call statement *only* when the caller is tainted to *avoid the path explosion issue* while covering all available gadgets which can be exploited by abusing *Java runtime polymorphism*.



Our approach: ODDFUZZ

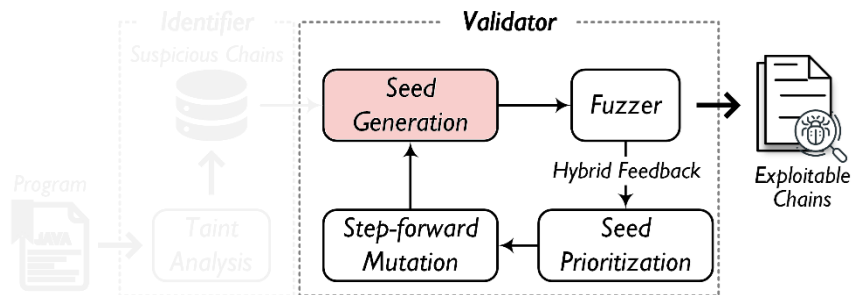
Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- Summary-based taint analysis + CHA
- ▣ *Structure-aware* seed generation
- Hybrid feedback-based prioritization
- Step-forward mutation rules



How to handle the *multilevel nested structure* of injection objects?

- Instantiate each class involved in the gadget chain and leverage *reflection* to dynamically collect available properties of each class to construct a *property tree*.



Our approach: ODDFUZZ

Lightweight Taint Analysis with Directed Greybox Fuzzing

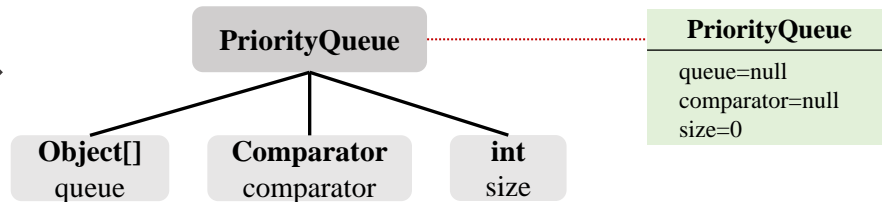
- Summary-based taint analysis + CHA
- **Structure-aware** seed generation
- Hybrid feedback-based prioritization
- Step-forward mutation rules



How to handle the **multilevel nested structure** of injection objects?

- Instantiate each class involved in the gadget chain and leverage **reflection** to dynamically collect available properties of each class to construct a **property tree**.

```
1 public class PriorityQueue<E> implements Serializable {
2     transient Object[] queue;
3     private final Comparator<? super E> comparator;
4     private void readObject(java.io.ObjectInputStream s)
5         {...} /*Source*/
6     private void heapify() {...} /*2nd Gadget*/
7     private void siftDown(int k, E x) { /*3rd Gadget*/
8         if (comparator != null)
9             siftDownUsingComparator(k, x);
10    ...}
11    private void siftDownUsingComparator(int k, E x) { /*4th Gadget*/
12        if (right < size &&
13            comparator.compare((E) c, (E) queue[right]) > 0)
14    ...}
15    /*org.apache.commons.collections4.comparators*/
16    public class TransformingComparator<I, O>
17        implements Comparator<I>, Serializable {
18        private final Transformer<? super I, ? extends O> transformer;
19        public int compare(I obj1, I obj2) { /*5th Gadget*/
20            Object value2 = this.transformer.transform(obj2);
21            ...}
22    }
```



Our approach: ODDFUZZ

Lightweight Taint Analysis with Directed Greybox Fuzzing

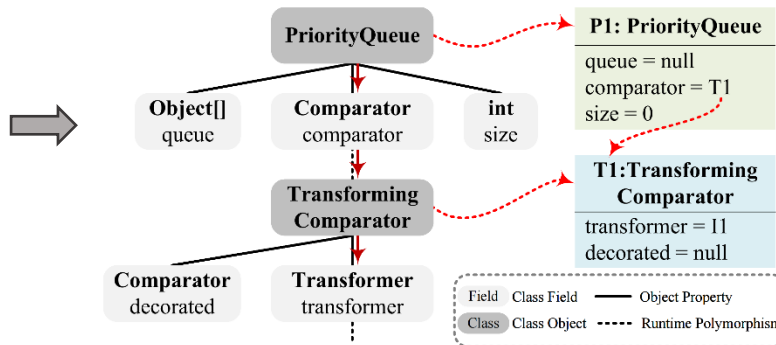
- Summary-based taint analysis + CHA
- Structure-aware seed generation
- Hybrid feedback-based prioritization
- Step-forward mutation rules

```
1 public class PriorityQueue<E> implements Serializable {
2     transient Object[] queue;
3     private final Comparator<? super E> comparator;
4     private void readObject(java.io.ObjectInputStream s)
5         {...} /*Source*/
6     private void heapify() {...} /*2nd Gadget*/
7     private void siftDown(int k, E x) { /*3rd Gadget*/
8         if (comparator != null)
9             siftDownUsingComparator(k, x);
10    ...}
11    private void siftDownUsingComparator(int k, E x) { /*4th Gadget*/
12        if (right < size &&
13            comparator.compare((E) c, (E) queue[right]) > 0)
14    ...}
15    /*org.apache.commons.collections4.comparators*/
16    public class TransformingComparator<I, O>
17        implements Comparator<I>, Serializable {
18        private final Transformer<? super I, ? extends O> transformer;
19        public int compare(I obj1, I obj2) { /*5th Gadget*/
20            Object value2 = this.transformer.transform(obj2);
21            ...}
22    }
```



How to handle the *multilevel nested structure* of injection objects?

- Merge two property trees if the property type of a field node in tree A is an object *inherited* by tree B of which the class holds the next gadget in the target chain.



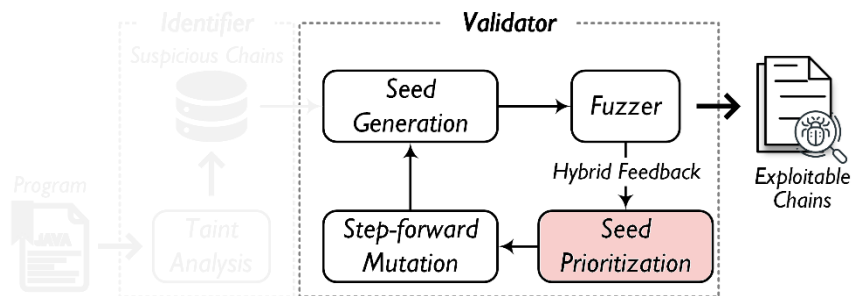
Our approach: ODDFUZZ

Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- Summary-based taint analysis + CHA
- Structure-aware seed generation
- ▣ *Hybrid feedback*-based prioritization
- Step-forward mutation rules



How to efficiently *select* and *schedule* the seeds to trigger sensitive sinks?



$$p(s, T_b) = \varphi(s) \cdot (1 - \tilde{d}(s, T_b))$$

Closer to sink

Trigger more branches in the gadget (exploring diverse execution paths)

Our approach: ODDFUZZ

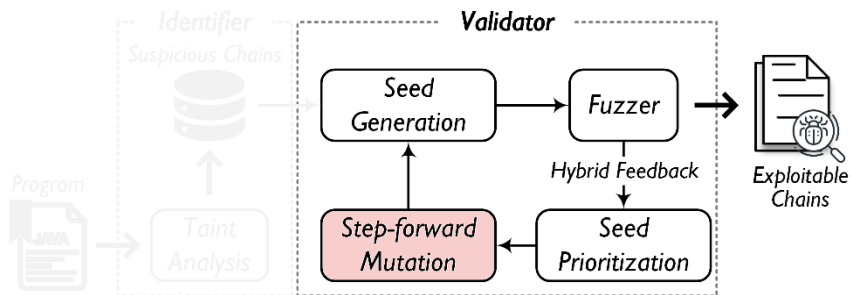
Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- Summary-based taint analysis + CHA
- Structure-aware seed generation
- Hybrid feedback-based prioritization
- ▣ *Step-forward* mutation rules



How to guarantee the *structural and semantical validity* of mutation?

- Employ JQF³, a *parametric* fuzzing framework which maps the structured inputs to a sequence of untyped bits (i.e., parameters), to mutate the generated seeds at the bit-level.

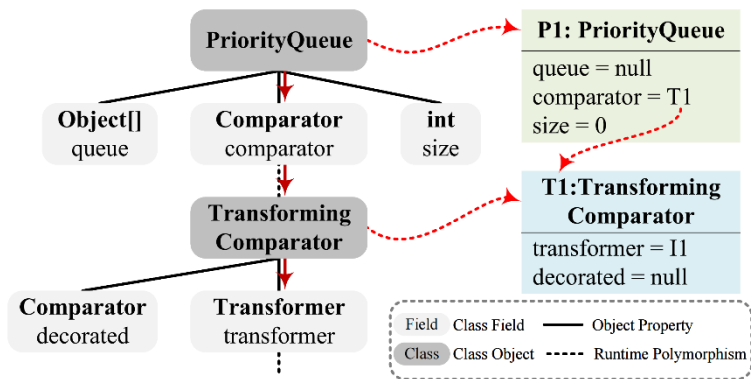


³<https://github.com/rohanpadhye/JQF>

Our approach: ODDFUZZ

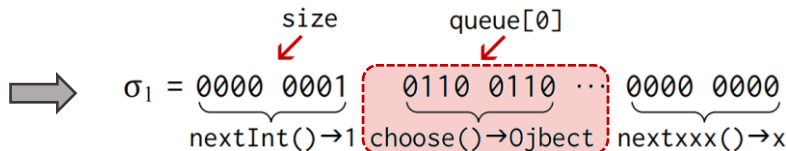
Lightweight Taint Analysis with Directed Greybox Fuzzing

- Summary-based taint analysis + CHA
- Structure-aware seed generation
- Hybrid feedback-based prioritization
- ▣ **Step-forward** mutation rules



How to guarantee the **structural** and **semantical validity** of mutation?

- Employ JQF³, a **parametric** fuzzing framework which maps the structured inputs to a sequence of untyped bits (i.e., parameters), to mutate the generated seeds at the bit-level.

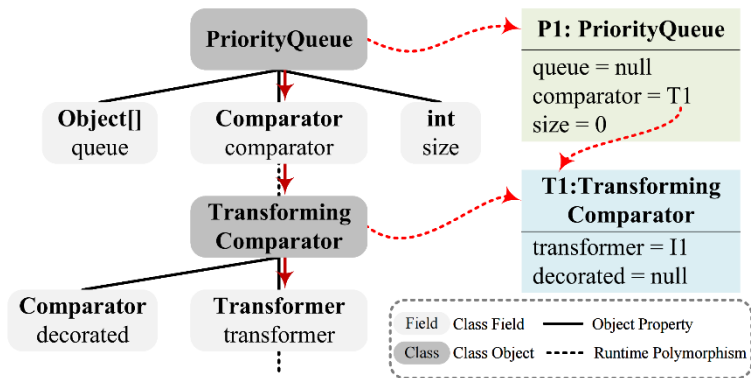


³<https://github.com/rohanpadhye/JQF>

Our approach: ODDFUZZ

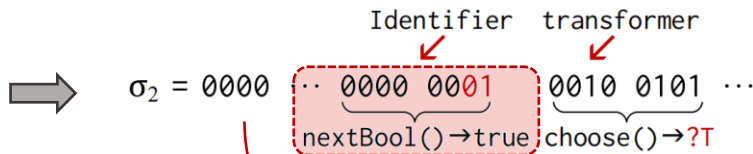
Lightweight Taint Analysis with Directed Greybox Fuzzing

- Summary-based taint analysis + CHA
- Structure-aware seed generation
- Hybrid feedback-based prioritization
- **Step-forward** mutation rules



How to guarantee the **structural and semantical validity** of mutation?

- Insert additional **identifier bytes** into the parametric sequence to **mutate the nested sub-objects** of the interesting injection object at the bit-level one by one.



Preserve mutated properties

Performance of ODDFUZZ

Experimental setup

- Target Applications
 - 22 Java libraries (covering 34 chains) from ysoserial
 - 4 well-known applications (Oracle WebLogic Server, Sonatype Nexus, Apache Dubbo, protostuff)
- Implementations
 - Repeat each experiment **10** times and report the average statistical performance.
 - Set the threshold for each gadget chain to **15** gadgets.
 - Limit the fuzzing campaign to **120** seconds

Performance of ODDFUZZ

Application	Version	LoC	Classes	Methods	Covered Sources	Covered Sinks	Known Chains	Identified Chains	Confirmed Chains	Analysis Time	Fuzzing Time
JDK	1.7	4.4M	38.5K	324.6K	7	4	4	9 (1)	1	1m51s	16m32s
AspectJWeaver	1.9.2	692.4K	7.1K	19.8K	4	2	1	9 (1)	0	1m56s	18m
BeanShell	2.0b5	44.8K	1.1K	17K	3	1	1	8 (0)	0	1m53s	16m
C3PO	0.9.5.2	30.3K	644	10.1K	6	3	1	13 (1)	1	1m50s	25m53s
Click	2.3.0	10.8K	73	8.5K	4	1	1	8 (1)	1	1m48s	15m26s
Clojure	1.8.0	58.4K	3.8K	25.7K	5	4	1	184 (1)	1	3m30s	6h7m34s
CommonsBeanutils	1.9.2	71.4K	504	7.8K	3	1	1	8 (1)	1	1m52s	14m25s
CommonsCollections	3.1	101K	798	9.7K	7	4	5	97 (5)	3	1m58s	3h10m53s
CommonsCollections4	4.0	101K	630	7.4K	5	2	2	112 (2)	2	1m55s	3h41m9s
FileUpload	1.3.1	10.5K	56	3.1K	3	1	1	8 (0)	0	1m55s	16m
Groovy	2.3.9	252.4K	4.2K	45.6K	4	1	1	13 (0)	0	2m8s	26m
Hibernate	4.3.11	855.7K	7.4K	42.7K	3	1	2	8 (2)	2	2m8s	14m7s
JBossInterceptors	2.0.0	24.2K	166	2.3K	2	1	1	8 (0)	0	1m51s	16m
JSON	2.4	28K	172	5.9K	3	2	1	9 (0)	0	1m52s	18m
JavassistWeld	3.12.1	60.4K	813	11.3K	2	1	1	8 (0)	0	1m58s	16m
Jython	2.5.2	271.9K	6.7K	66.4K	4	1	1	32 (1)	0	2m54s	1h4m
MozillaRhino	1.7R2	118.7K	329	8.2K	4	2	2	7 (2)	2	1m56s	12m10s
Myfaces	2.2.9	330.1K	1.8K	22.8K	2	1	2	7 (0)	0	2m1s	14m
ROME	1.0	94.5K	423	6.9K	2	1	1	5 (1)	1	1m48s	8m53s
Spring	4.1.4	904.3K	1.3K	14.5K	3	2	2	10 (0)	0	1m59s	20m
Vaadin	7.7.14	572.1K	4.5K	17.5K	4	1	1	13 (1)	1	1m54s	24m37s
Wicket	6.23.0	420.7K	3.2K	11.1K	2	1	1	7 (0)	0	1m50s	14m
Total		-	-	-	-	-	34	583 (20)	16	-	-

Performance of ODDFUZZ

Application	Known Chains	GadgetInspector			SerHybrid			ODDFUZZ			
		Identified Chains	Confirmed Chains	Analysis Time	Identified Chains	Confirmed Chains	Analysis Time	Identified Chains	Confirmed Chains	Analysis Time	Fuzzing Time
JDK	4	5	0	53s	N/A	N/A	N/A	9 (1)	1	1m51s	16m32s
AspectJWeaver	1	6	0	41s	N/A	N/A	N/A	9 (1)	0	1m56s	18m
BeanShell	1	2	0	49s	1	0	10m55s	8 (0)	0	1m53s	16m
C3P0	1	2	0	48s	N/A	N/A	N/A	13 (1)	1	1m50s	25m53s
Click	1	4	0	39s	N/A	N/A	N/A	8 (1)	1	1m48s	15m26s
Clojure	1	12	1	40s	N/A	N/A	Timeout	184 (1)	1	3m30s	6h7m34s
CommonsBeanutils	1	2	0	37s	0	0	13m6s	8 (1)	1	1m52s	14m25s
CommonsCollections	5	4	1	39s	1	1	26m51s	97 (5)	3	1m58s	3h10m53s
CommonsCollections4	2	4	0	38s	1	1	11m21s	112 (2)	2	1m55s	3h41m9s
FileUpload	1	3	0	38s	N/A	N/A	N/A	8 (0)	0	1m55s	16m
Groovy	1	4	0	47s	3	0	1h26m	13 (0)	0	2m8s	26m
Hibernate	2	3	0	41s	3	0	56m37s	8 (2)	2	2m8s	14m7s
JBossInterceptors	1	2	0	38s	N/A	N/A	N/A	8 (0)	0	1m51s	16m
JSON	1	2	0	39s	N/A	N/A	N/A	9 (0)	0	1m52s	18m
JavassistWeld	1	2	0	39s	N/A	N/A	N/A	8 (0)	0	1m58s	16m
Jython	1	42	1	50s	N/A	N/A	Timeout	32 (1)	0	2m54s	1h4m
MozillaRhino	2	3	0	40s	N/A	N/A	N/A	7 (2)	2	1m56s	12m10s
Myfaces	2	2	0	37s	N/A	N/A	N/A	7 (0)	0	2m1s	14m
ROME	1	2	0	36s	0	0	6m30s	5 (1)	1	1m48s	8m53s
Spring	2	2	0	38s	N/A	N/A	N/A	10 (0)	0	1m59s	20m
Vaadin	1	5	0	37s	N/A	N/A	N/A	13 (1)	1	1m54s	24m37s
Wicket	1	3	0	36s	N/A	N/A	N/A	7 (0)	0	1m50s	14m
Total	34	116	3	-	9	2	-	583 (20)	16	-	-

Performance of ODDFUZZ

ID	Gadget Chain	Affected Version	# Gadgets	ODDFUZZ		GadgetInspector	SerHybrid
				Identified	Validated		
1	AspectJWeaver	aspectjweaver-1.9.2	9	✓	✗	-	-
2	BeanShell1	bsh-2.0b5	6	-	-	-	-
3	C3P0	c3p0-0.9.5.2	6	✓	✓	-	-
4	Click1	click-nodeps-2.3.0	10	✓	✓	-	-
5	Clojure	clojure-1.8.0	10	✓	✓	✓	-
6	CommonsBeanutils1	commons-beanutils-1.9.2	5	✓	✓	-	-
7	CommonsCollections1	commons-collections-3.1	7	✓	✗	✓	-
8	CommonsCollections2	commons-collections4-4.0	13	✓	✓	-	✓
9	CommonsCollections3	commons-collections-3.1	13	✓	✗	-	-
10	CommonsCollections4	commons-collections4-4.0	15	✓	✓	-	-
11	CommonsCollections5	commons-collections-3.1	8	✓	✓	-	-
12	CommonsCollections6	commons-collections-3.1	10	✓	✓	-	✓
13	CommonsCollections7	commons-collections-3.1	9	✓	✓	-	-
14	FileUpload1	commons-fileupload-1.3.1	3	-	-	-	-
15	Groovy1	groovy-2.3.9	10	-	-	-	-
16	Hibernate1	hibernate-core-4.3.11.Final	7	✓	✓	-	-
17	Hibernate2	hibernate-core-4.3.11.Final	9	✓	✓	-	-
18	JBossInterceptors1	jboss-interceptor-core:2.0.0.Final	5	-	-	-	-
19	JRMPCClient	JDK-1.7	13	-	-	-	-
20	JRMPListener	JDK-1.7	9	-	-	-	-
21	JSON1	json-lib-jar-jdk15:2.4	22	-	-	-	-
22	JavassistWeld1	javassist-3.12.1.GA	5	-	-	-	-
23	Jdk7u21	JDK-1.7	11	-	-	-	-
24	Jython1	jython-standalone-2.5.2	5	✓	✗	✓	-
25	MozillaRhino1	js-1.7R2	8	✓	✓	-	-
26	MozillaRhino2	js-1.7R2	12	✓	✓	-	-
27	Myfaces1	myfaces-impl-2.2.9	4	-	-	-	-
28	Myfaces2	myfaces-impl-2.2.9	6	-	-	-	-
29	ROME	rome-1.0	15	✓	✓	-	-
30	Spring1	spring-core:4.1.4.RELEASE	11	-	-	-	-
31	Spring2	spring-core:4.1.4.RELEASE	12	-	-	-	-
32	URLDNS	JDK	7	✓	✓	-	-
33	Vaadin1	vaadin-server-7.7.14	10	✓	✓	-	-
34	Wicket1	wicket-util-6.23.0	3	-	-	-	-

No.	Application	Version	Impact	Status	CVE-ID
1	WebLogic	12.2.1.4.0	RCE	Patched	CVE-2020-14756
2	WebLogic	12.2.1.4.0	RCE	Patched	CVE-2020-14825
3	WebLogic	12.2.1.4.0	RCE	Patched	CVE-2021-2135
4	Sonatype Nexus	3.25.0	RCE	Patched	CVE-2020-15871
5	Apache Dubbo	2.7.7	RCE	Patched	CVE-2020-11995
6	ProtoStuff	1.8.0	RCE	Reported	

Effectiveness

- ✓ Statically identify 20 chains and dynamically validated 16 of them, covering *all* cases found by baselines.
- ✓ Report 6 exploitable Java ODD vulnerabilities, 5 of them have been assigned CVE-IDs.

Limitations

- ✓ Imprecise and unsound static analysis.
- ✓ Sub-optimal generation strategy.
- ✓ Require human efforts to construct practical exploits.

Conclusion

ODD Vulnerability Attack

O -- Open

Arbitrary objects may be injected by adversaries, which breaks the traditional trust boundary of inter-process data transmission to induce attacks on users.

D -- Dynamic

The deserialization path of an injected serialized object can be manipulated by abusing runtime polymorphic or other dynamic features.

D -- Deserialization

Magic methods get executed automatically by the deserializer even after deserialization finishes!

Controlling Data Types
=> Controlling Code!

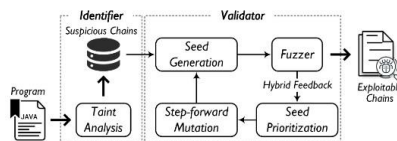


Threat Model

Our approach: ODDFUZZ

Lightweight Taint Analysis with Directed Greybox Fuzzing

- Summary-based taint analysis + CHA
- Structural seed generation
- Hybrid feedback-based prioritization
- Step-forward mutation rules



Performance of ODDFUZZ

Application	Known Chains	GadgetInspector			SerHybrid			ODDFUZZ		
		Identified Chains	Confirmed Chains	Analysis Time	Identified Chains	Confirmed Chains	Analysis Time	Identified Chains	Confirmed Chains	Analysis Time
JDK	4	0	53%	N/A	N/A	N/A	9 (1)	1	1m51s	18m32s
AspectWeaver	1	6	0	41s	N/A	N/A	9 (1)	0	1m56s	16m
BeanShell	1	2	0	49s	1	0	10m55s	0	1m53s	16m
CPUJ	1	2	0	46s	N/A	N/A	13 (1)	1	1m56s	25m33s
Click	1	4	0	39s	N/A	N/A	8 (1)	1	1m48s	15m26s
Clojure	1	12	1	46s	N/A	N/A	Timeout	184 (1)	1	3m36s
CommonsBeans	1	2	0	37s	0	0	13m45s	8 (1)	0	1m53s
CommonsCollections	5	4	1	39s	1	1	26m51s	97 (5)	3	1m56s
CommonsCollections4	2	4	0	38s	1	1	11m21s	112 (2)	2	1m53s
FileUpload	1	1	0	38s	N/A	N/A	8 (0)	0	1m53s	14m25s
Groovy	1	4	0	47s	3	0	1h26m	13 (0)	0	2m8s
Hibernate	2	3	0	41s	3	0	56m37s	8 (2)	2	2m8s
JBIInterceptors	1	2	0	38s	N/A	N/A	8 (0)	0	1m51s	16m
JSON	1	2	0	39s	N/A	N/A	9 (0)	0	1m52s	18m
JavaScripWorld	1	2	0	39s	N/A	N/A	8 (0)	0	1m56s	16m
Jython	1	42	1	50s	N/A	N/A	Timeout	32 (1)	0	2m54s
MailHutlino	2	3	0	46s	N/A	N/A	7 (2)	2	1m56s	12m10s
MyFaces	2	2	0	37s	N/A	N/A	7 (0)	0	2m1s	14m
ROME	1	2	0	36s	0	0	6m30s	5 (1)	1	1m48s
Spring	2	2	0	38s	N/A	N/A	10 (0)	0	1m59s	20m
Validator	1	5	0	37s	N/A	N/A	13 (1)	1	1m54s	24m37s
Wicket	1	3	0	36s	N/A	N/A	7 (0)	0	1m50s	14m
Total	34	116	1	-	9	2	-	583 (20)	16	-

Performance of ODDFUZZ

ID	Gadget Chains	Affected Version	# Gadgets	ODDFUZZ Identified	ODDFUZZ Validated	GadgetInspector	SerHybrid
1	AspectWeaver	aspectjweaver-1.9.2	9	✓	✓	-	-
2	BeanShell	bsf-2.0.0	6	✓	✓	-	-
3	CPUJ	cpuaj-0.9.2	6	✓	✓	-	-
4	Click	click-0.9.2	10	✓	✓	-	-
5	Clojure	clojure-1.10.0	10	✓	✓	-	-
6	CommonsBeans	commons-beans-1.2	5	✓	✓	-	-
7	CommonsCollections	commons-collections-3.1	7	✓	✓	-	-
8	CommonsCollections4	commons-collections4-0.4	10	✓	✓	-	-
9	CommonsCollections	commons-collections-3.1	10	✓	✓	-	-
10	CommonsCollections	commons-collections-4.0	10	✓	✓	-	-
11	CommonsCollections	commons-collections-3.1	10	✓	✓	-	-
12	CommonsCollections	commons-collections-3.1	10	✓	✓	-	-
13	CommonsCollections	commons-collections-3.1	10	✓	✓	-	-
14	FileUpload	commons-fileupload-1.3.1	5	✓	✓	-	-
15	Groovy	groovy-2.5.0	10	✓	✓	-	-
16	Hibernate	hibernate-core-5.3.11.Final	7	✓	✓	-	-
17	JSON	json-20180127	7	✓	✓	-	-
18	MailHutlino	mail-hutlino-core-2.0.0.Final	3	✓	✓	-	-
19	ROME	rome-1.7	10	✓	✓	-	-
20	Spring	spring-core-5.3.10	10	✓	✓	-	-
21	Validator	validation-api-2.0.1	10	✓	✓	-	-
22	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
23	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
24	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
25	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
26	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
27	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
28	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
29	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
30	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
31	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
32	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
33	Wicket	wicket-core-7.1.0	10	✓	✓	-	-
34	Wicket	wicket-core-7.1.0	10	✓	✓	-	-

No.	Application	Version	Impact	Status	CVE-ID
1	WebLogic	12.2.1.4.0	RCE	Patched	CVE-2020-14756
2	WebLogic	12.2.1.4.0	RCE	Patched	CVE-2020-14825
3	WebLogic	12.2.1.4.0	RCE	Patched	CVE-2021-2135
4	Sonytype Nexus	3.25.0	RCE	Patched	CVE-2020-15871
5	Apache Dubbo	2.7.7	RCE	Patched	CVE-2020-11995
6	ProxStuf	1.8.0	RCE	Reported	-

Effectiveness

- ✓ Statically identify 20 chains and dynamically validated 16 of them, covering all cases found by baselines
- ✓ Report 6 exploitable Java ODD vulnerabilities, 5 of them have been assigned CVE IDs

Limitations

- ✓ Imperfect and unsound static analysis
- ✓ Sub-optimal generation strategy
- ✓ Require human effort to construct practical exploits

Thanks for listening!

✉ DX120210088@yzu.edu.cn

🔗 <https://github.com/ODDFuzz/ODDFuzz>



RiSS Lab



Personal Page



Paper Link



揚州大學
YANGZHOU UNIVERSITY



蚂蚁集团
ANT GROUP



清華大學
Tsinghua University



華東師範大學
EAST CHINA NORMAL
UNIVERSITY