44th IEEE Symposium on Security and Privacy

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

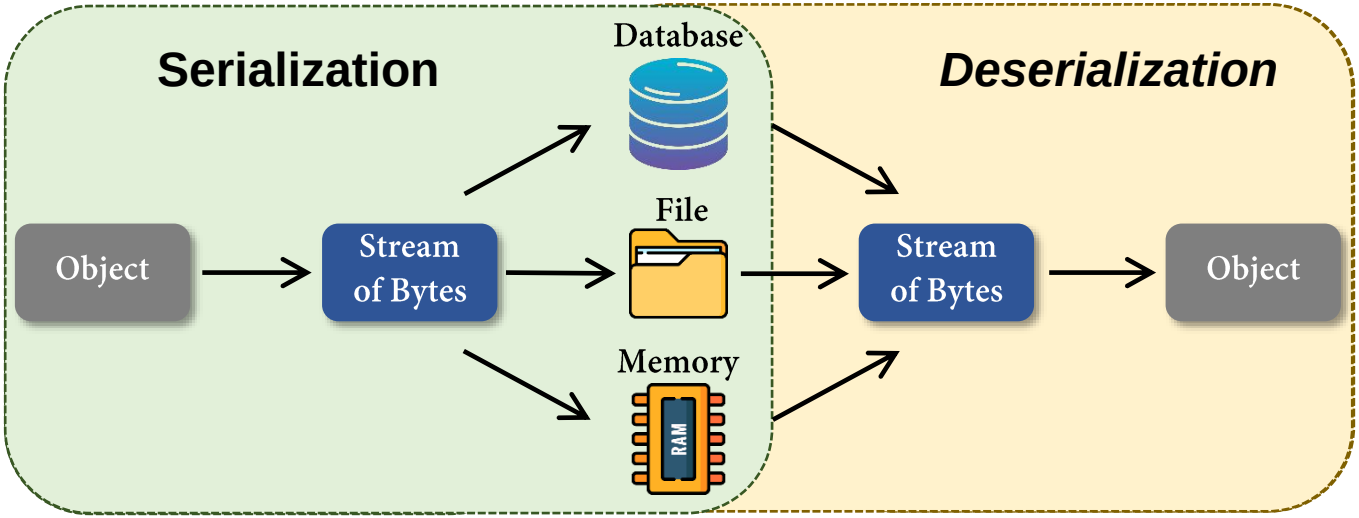
Sicong Cao[1], Biao He[2], Xiaobing Sun[1], Yu Ouyang[2], Chao Zhang[3], Xiaoxue Wu[1], Ting Su[4], Lili Bo[1], Bin Li[1], Chuanlei Ma[2], Jiajia Li[2], and Tao Wei[2]

[1]Yangzhou University
[2]Ant Group
[3]Tsinghua University
[4]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.

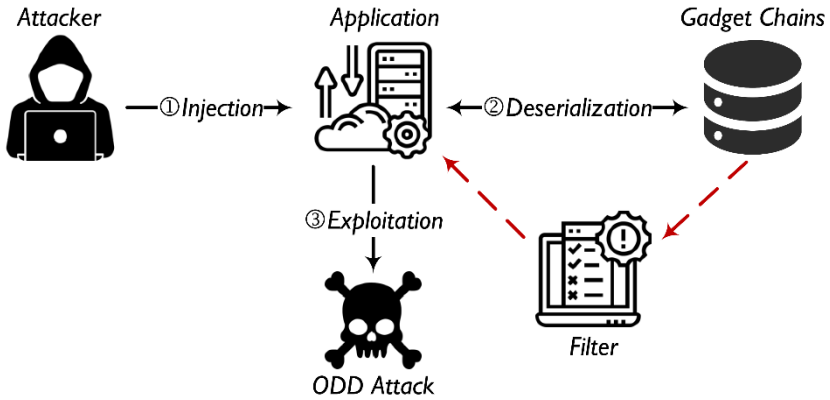
- ***D --- Dynamic***

  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

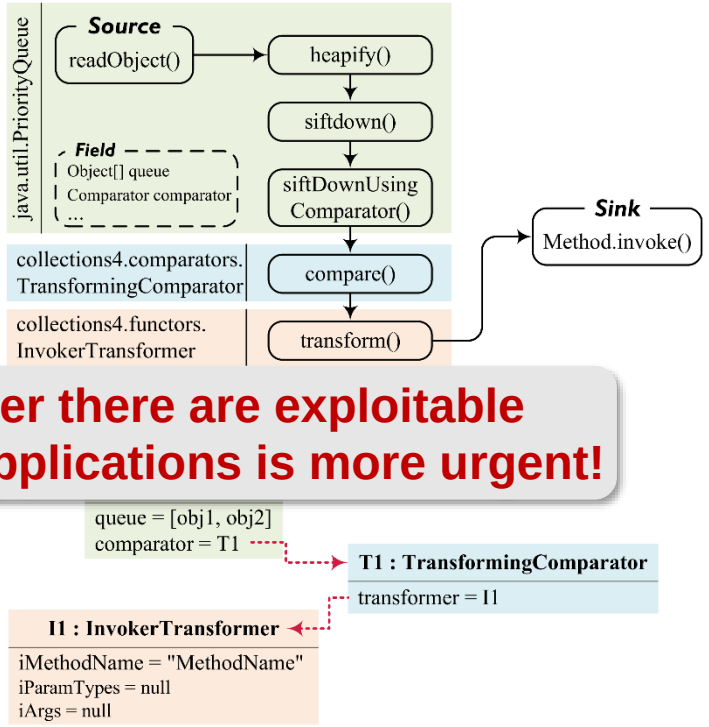
ODD Vulnerability Attack

```
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)
                          {...} /*Source*/
5    private void heapify() {...} /*2nd Gadget*/
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.comm...ctions4.comparators*/
15 public class Tr...
           impleme...
16   private final T...
17   public int comp...
18     Object value2...
19     ...}}

20 /*org.apache.commons.collections4.functors*/
21 public class InvokerTransformer<I, O>
           implements Transformer<I, O>, Serializable {
22   public Object transform(Object input) { /*6th Gadget/
23     Class<?> cls = input.getClass();
24     Method method = cls.getMethod(this.iMethodName,
                              this.iParamTypes);
25     return method.invoke(input, this.iArgs); /*Sink*/
26 }}
```

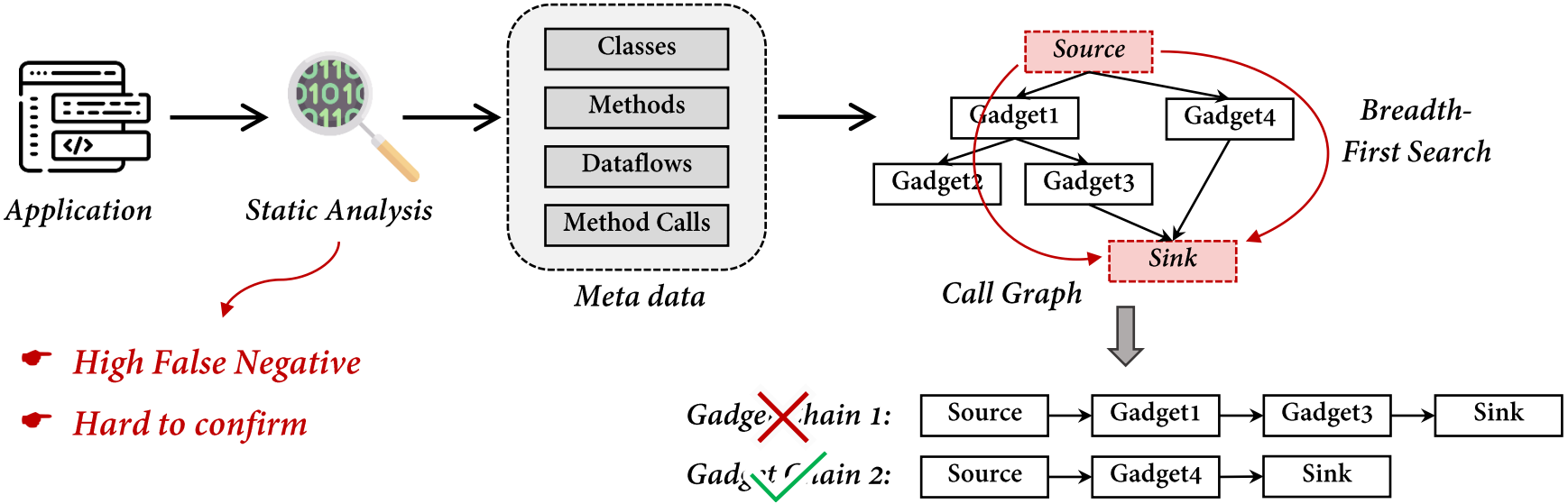
Apache Commons Collections



Injection object

Existing Solutions

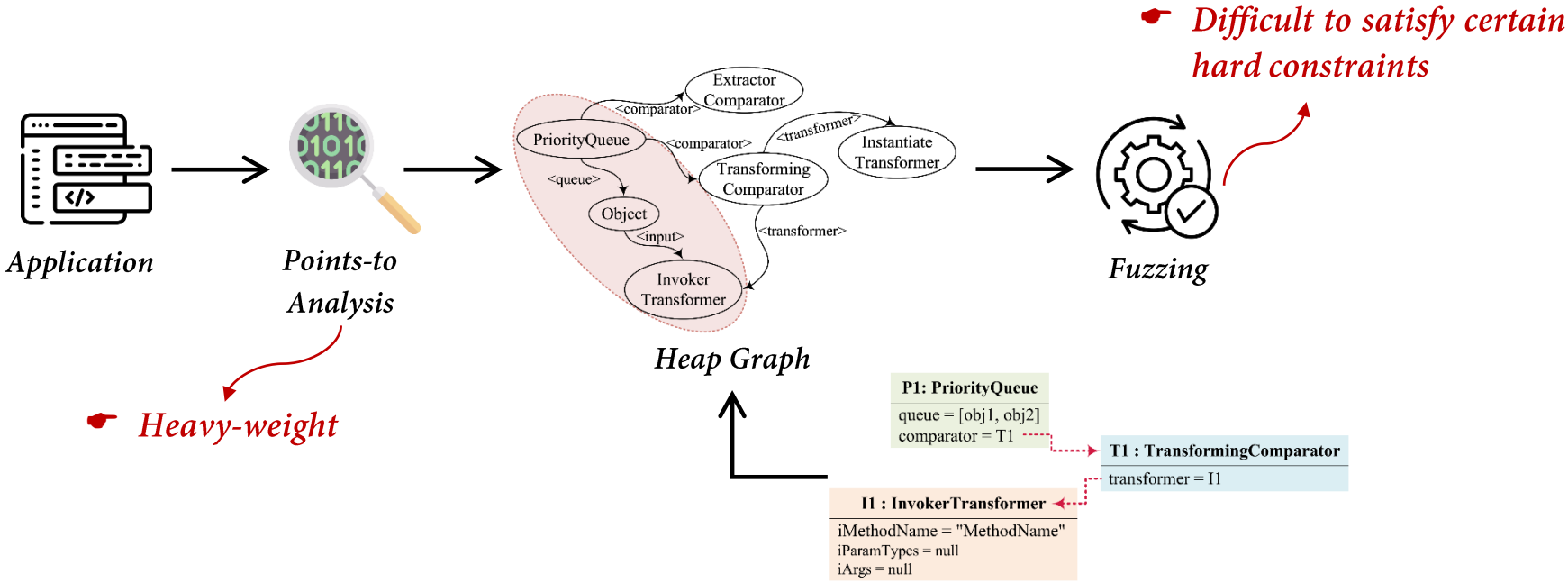
Gadget Inspector [BlackHat 2018]



[1]I. Haken, "Automated discovery of deserialization gadget chains," BlackHat USA, 2018.

Existing Solutions

SerHybrid [ASE 2022]

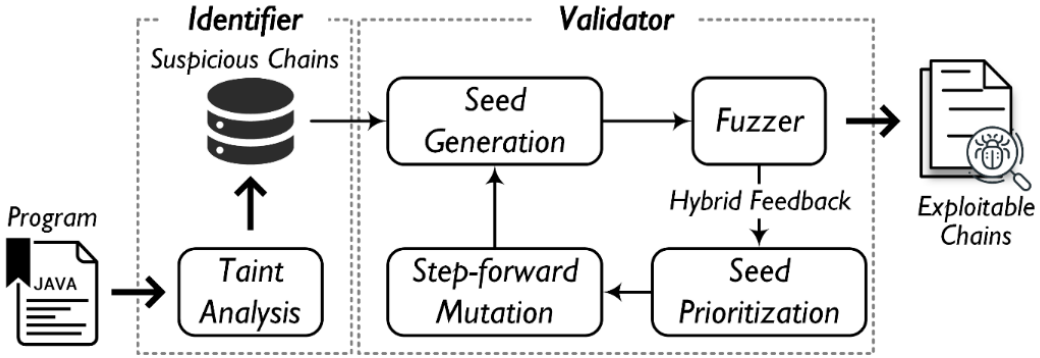


[2]S. Rasheed and J. Dietrich, "A hybrid analysis to detect java serialization vulnerabilities," IEEE/ACM International Conference on Software Engineering, 2020.

Our approach: ODDFuzz

Lightweight Taint Analysis with *Directed* Greybox Fuzzing

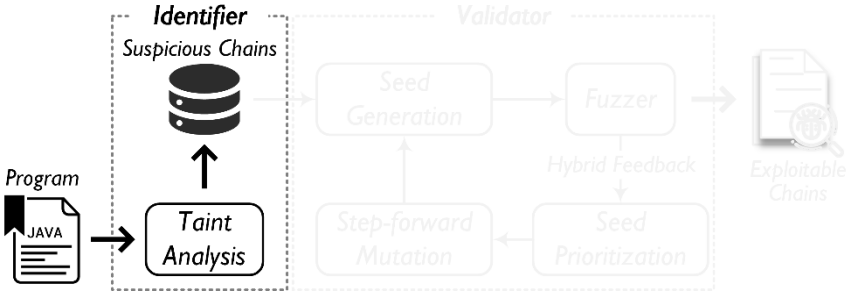
- *Summary*-based taint analysis + CHA
- *Structure-aware* seed generation
- *Hybrid* feedback-based prioritization
- *Step-forward* mutation rules



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



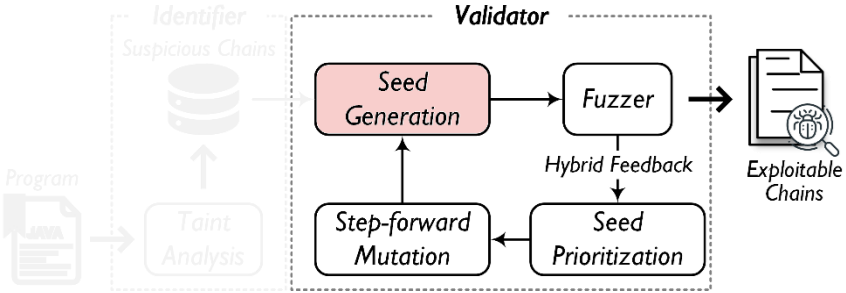
How to effectively and efficiently identify potential gadget chains?

- 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

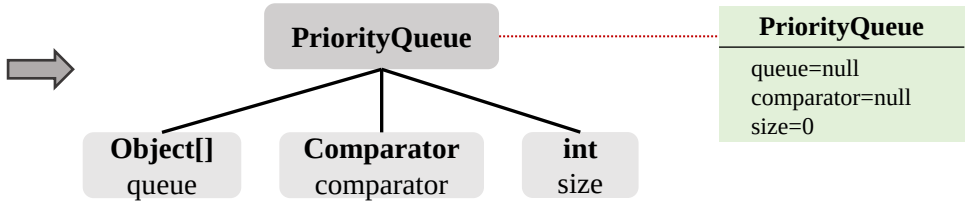
- *Summary*-based taint analysis + CHA
- *Structure-aware* seed generation
- *Hybrid* feedback-based prioritization
- *Step-forward* mutation rules

```
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)
                          {...} /*Source*/
5    private void heapify() {...} /*2nd Gadget*/
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<I, O>
             implements Comparator<I>, Serializable {
16   private final Transformer<? super I, ? extends O> transformer;
17   public int compare(I obj1, I obj2) {   /*5th Gadget*/
18     Object value2 = this.transformer.transform(obj2);
19     ...}}
```

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

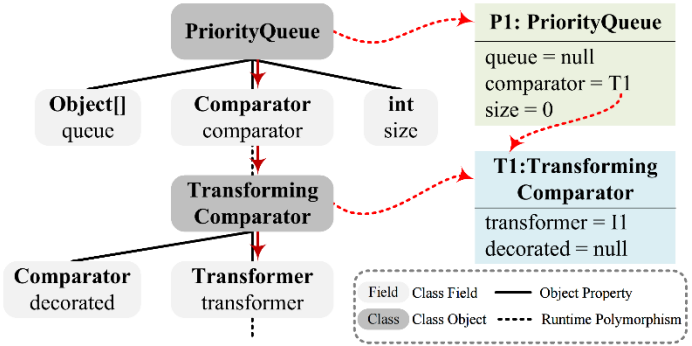
- *Summary*-based taint analysis + CHA
- *Structure-aware* seed generation
- *Hybrid* feedback-based prioritization
- *Step-forward* mutation rules

```
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)
                          {...} /*Source*/
5    private void heapify() {...} /*2nd Gadget*/
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<I, O>
             implements Comparator<I>, Serializable {
16   private final Transformer<? super I, ? extends O> transformer;
17   public int compare(I obj1, I obj2) {    /*5th Gadget*/
18     Object value2 = this.transformer.transform(obj2);
19     ...}}
```

How to handle the multilevel nested structure of injection objects?

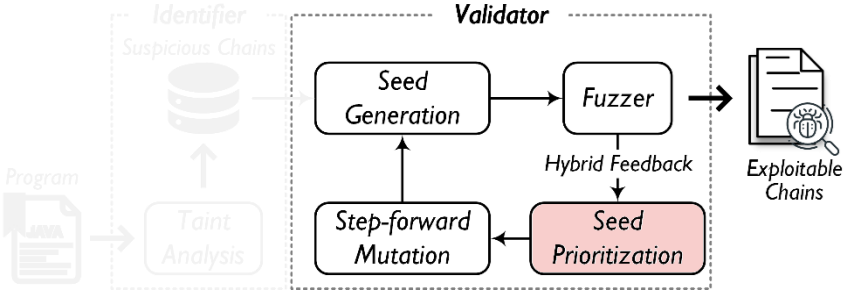
- Merge two property trees if the property type of a field node in tree A is an object inherited by tree B of which the class holds the next gadget in the target chain.



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?

Closer to sink

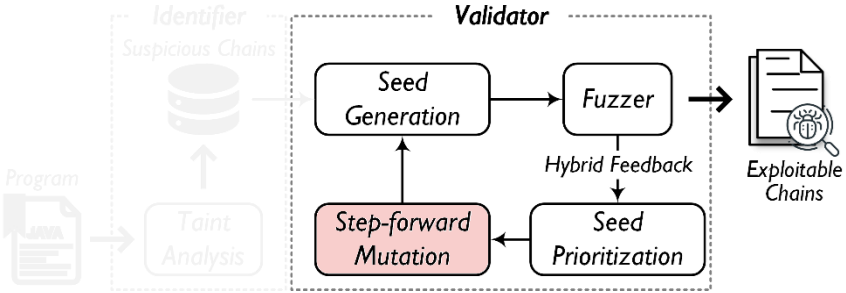
$$p(s, T_b) = \varphi(s) \cdot (1 - \tilde{d}(s, T_b))$$

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?

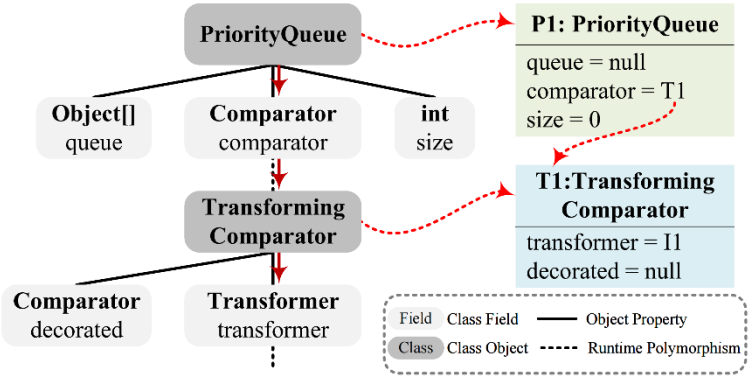
- Employ JQF[3], a parametric fuzzing framework which maps the structured inputs to a sequence of untyped bits (i.e., parameters), to mutate the generated seeds at the bit-level.

[3]https://github.com/rohanpadhye/JQF

Our approach: ODDFuzz

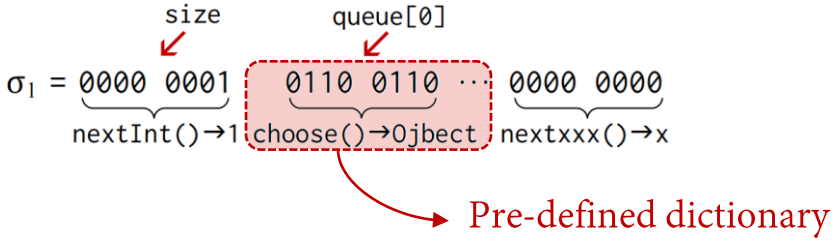
Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- *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 JQF[3], 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.

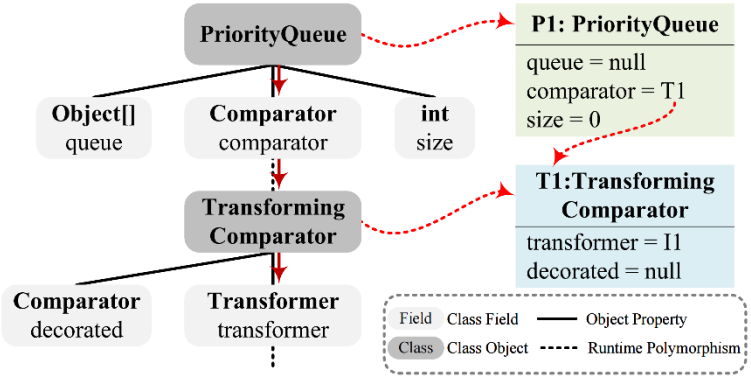


[3]https://github.com/rohanpadhye/JQF

Our approach: ODDFuzz

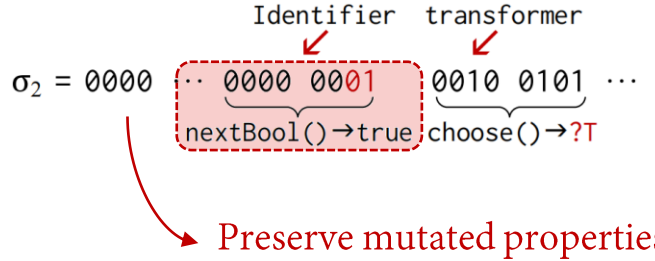
Lightweight Taint Analysis with *Directed* Greybox Fuzzing

- *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.



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
C3P0	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	*JRMPClient*	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 and introduces attack surfaces

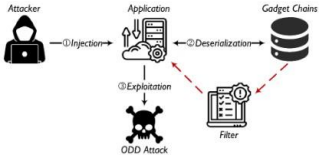
- D -- *Dynamic*

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

- D -- *Deserialization*

  Magic methods get executed *automatically* by the deserialiser, even before deserialisation finishes!

Controlling Data Types => Controlling Code!

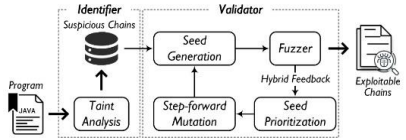


Threat Model

Our approach: ODDFuzz

Lightweight Taint Analysis with *Directed* Greybox Fuzzing

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



Performance of ODDFuzz

Application	Known Chains	GadgetInspector Identified Chains	GadgetInspector Confirmed Chains	GadgetInspector Analysis Time	SerHybrid Identified Chains	SerHybrid Confirmed Chains	SerHybrid Analysis Time	ODDFUZZ Identified Chains	ODDFUZZ Confirmed Chains	ODDFUZZ Analysis Time	ODDFUZZ 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 Identified	ODDFuzz Validated	GadgetInspector	SerHybrid
1	*AspectJWeaver*	aspectjweaver-1.9.2	9	✓	#	-	-
2	*BeanShell*	bsh-2.065	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	13	✓	✓	-	-
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	5	-	-	-	-
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	*JRMPClient*	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

44th IEEE Symposium on Security and Privacy

Thanks for listening!

DX120210088@yzu.edu.cn

https://github.com/ODDFuzz/ODDFuzz

RiSS Lab

Personal Page

Paper Link

YANGZHOU UNIVERSITY

蚂蚁集团 ANT GROUP

Tsinghua University

EAST CHINA NORMAL UNIVERSITY