
SUBVERTING PROBASE SECURITY FOR FUN AND PROFIT
A SOPHISTICATED ATM BLACKBOXING CASE
Frank Boldewin
ATRUVIA

Before diving into the details…

A brief introduction to ATM blackboxing

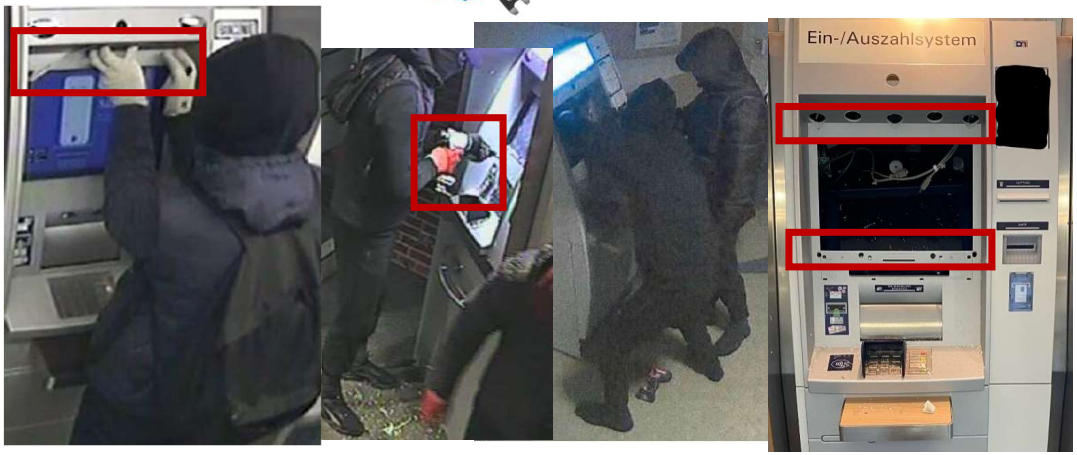
- **Attack vector**
  - In a blackboxing attack an **unauthorized device** (notebook or proprietary engineered device) is directly **connected to** the ATM **cash dispenser** in order to **issue commands** for cash out.
  - Most blackboxes are based on **notebooks with** an **operating system**, a **specific vendor stack** installed to **communicate** correctly **with** the **ATM** hardware and a **Malware for** the **cashout** itself.
- **Blackboxing attack types** (examples)
  - Connecting to **unencrypted dispensers**
  - **Brute forcing** weak keys
  - Attacking **firmware weaknesses**


Let‘s dive deeper
and subvert
Diebold Nixdorf‘s
ProCash security

Gaining physical access 1/2

- In **May 2021**, a series of **sophisticated** ATM **blackboxing attacks** started across Europe, where the perpetrators **targeted** Diebold Nixdorf **CINEO 4060 devices with RM3 cash dispensers**.
- **Physical access to** the **ATMs** was **gained** either **by using** a **cordless drill** with the aid of a **drilling template** or, alternatively, by **melting using** a **gas soldering iron**.

Drilling



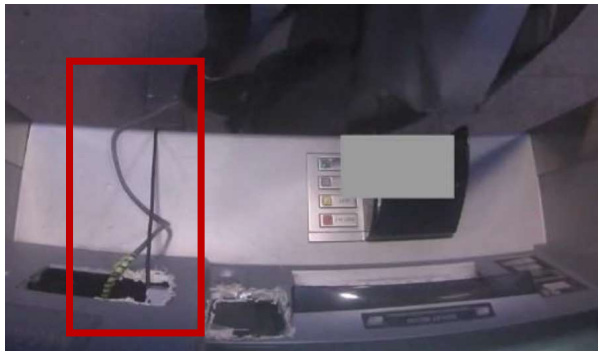
Melting

Gaining physical access 2/2

- The **perpetrators** then **connected** the **cables of** the ATM **dispenser and shutter to** their **blackbox**.
- Optionally, an **external control unit** was **connected** to locally **control power on/off** the ATM.

Connected notebook after drilling

Connections to shutter and dispenser after melting



External control unit

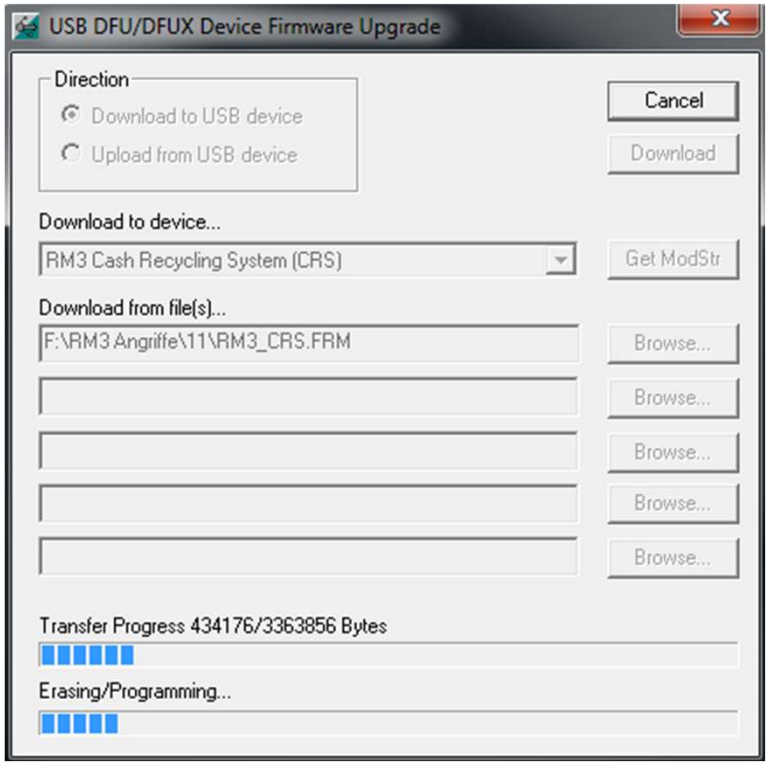
Seized equipment

- Due to the **complexity** of the **modus operandi**, the **perpetrators needed** an **average of 1,5 - 2 hours** for an attack. During this process, some of the **perpetrators were disturbed** while **others** were **arrested**, and **equipment** was **seized**.
- **Seized devices:**
  - **Mobile WiFi device** (used to establish a remote connection)
  - **USB Stick** (Fake CrypTA authentication dongle)
  - **Notebook** HP Elitebook 2570p (Acting as blackbox)
- **Notebook core software components:**
  - **Windows** 7
  - Infineon **TPM** Professional **Package**
  - **JDK** 1.8.0_181 + **Javassist**
  - **Teamviewer**
  - **Manipulated** Diebold Nixdorf **Platform Software** (ProBase)

- In order **to prevent unauthorized devices** from **communicating with** the ATM's cash **dispenser**, Diebold Nixdorf has **established** a number of **protection measures** to **make** a successful **blackbox attack** much more **difficult**.
- Through a number of **manipulations on** the **vendor stack** (ProBase) and **other components**, attackers were able to **bypass** these **protections** and generated a **cashout** under certain circumstances.

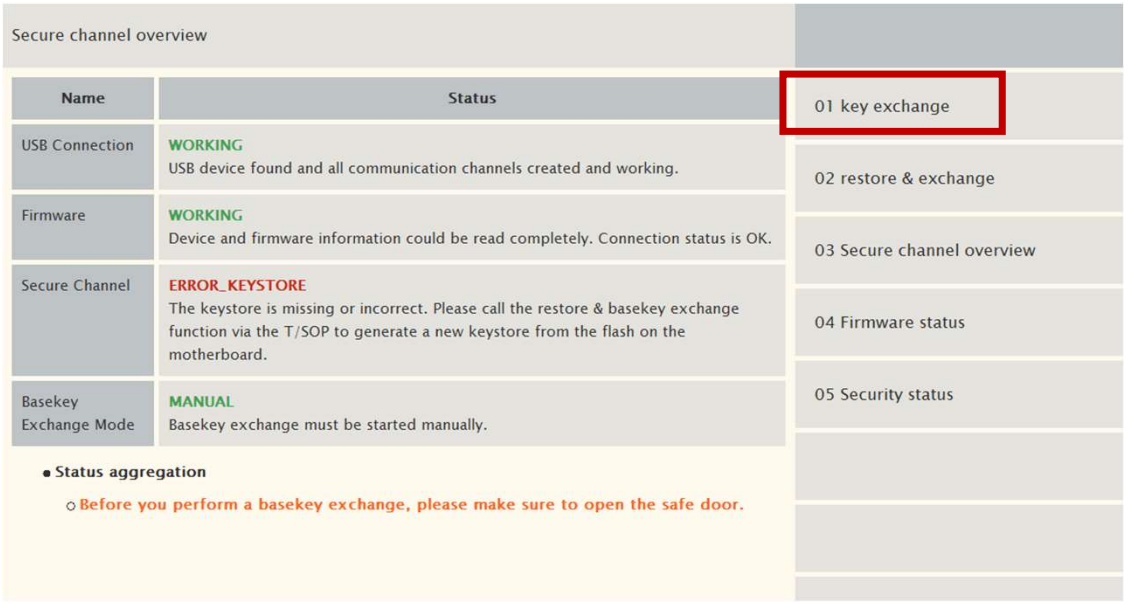
Firmware downgrading and a new key exchange

Perpetrators **loaded** an **old** original **firmware** (RM3 v1318 from 2012) that **does not require** an **open vault door to create** a **new key pairing** between the dispenser and the ATM PC.



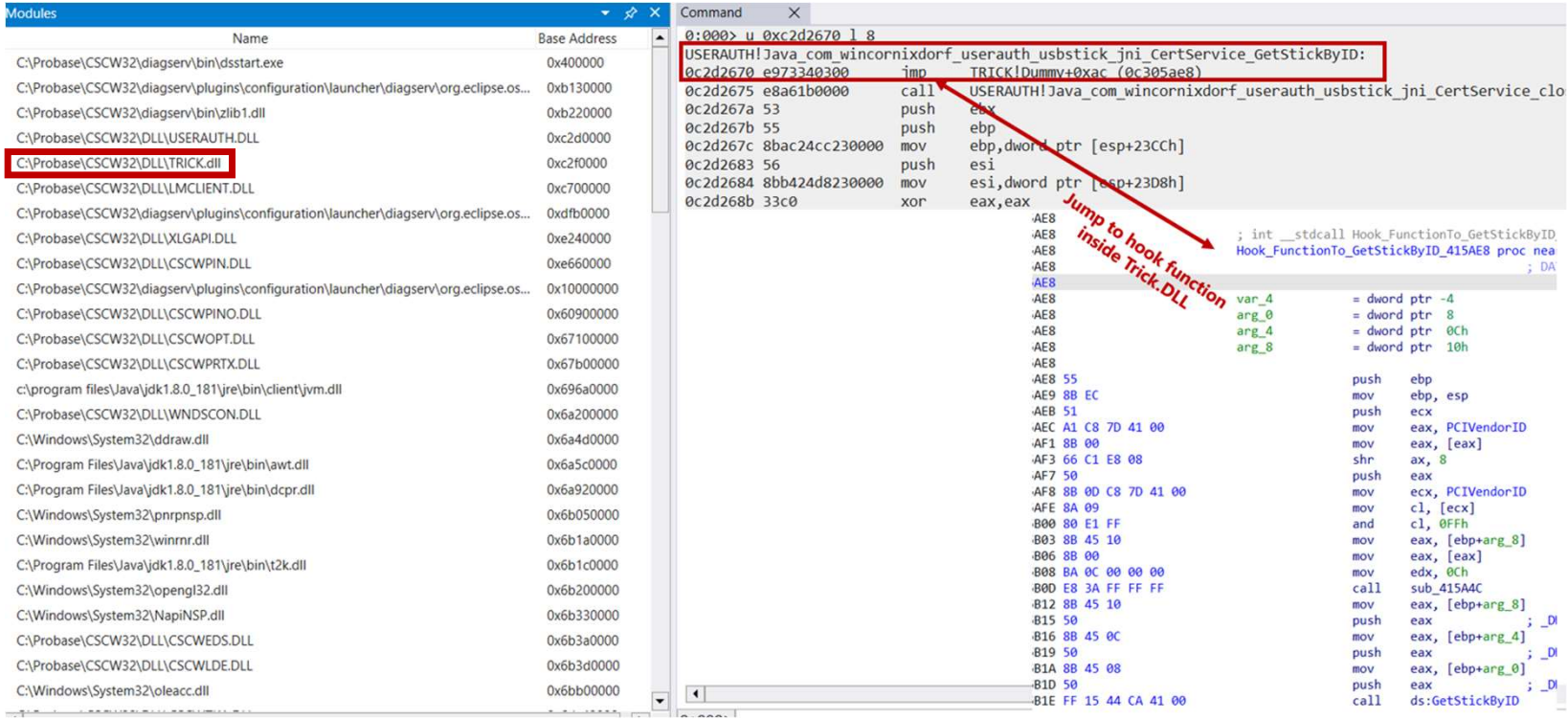
Afterwards a **new encryption key** is being **exchanged** between dispenser and attacker notebook **via** the Technical Service and Operator Panel (**T/SOP**).
In order **to achieve** a successful **key exchange** several **requirements** need to be fullfilled, like **access** to the **advanced T/SOP menu** functions and the **presence** of **original hardware**.

This is where several software manipulations come in place!



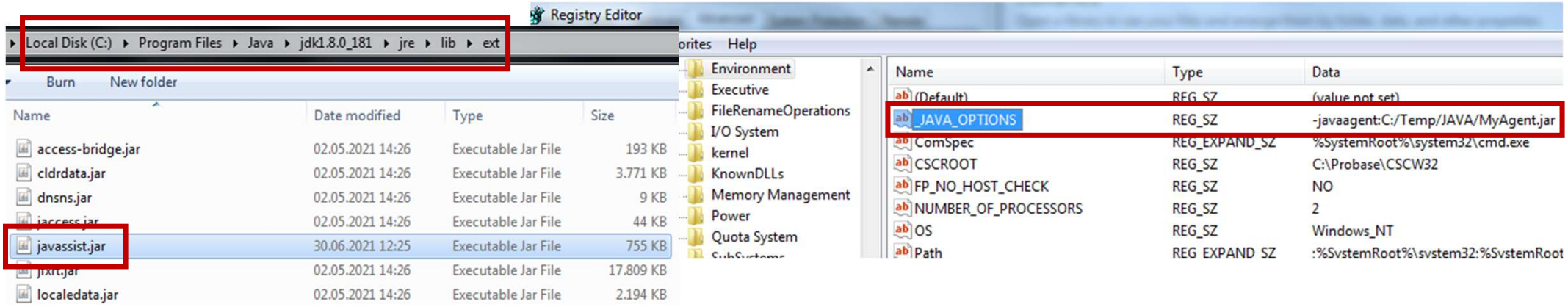
Faking the CrypTA USB stick 1/4

- To gain **access** to the **advanced T/SOP menu** a special USB stick called **CrypTA** is **required**. Usually **only certified service engineers have access** to such devices. The **stick validates against** the **T/SOP** by **check** a **PIN**, a **certificate** and a **specific USB Vendor ID**.
- In order to **simulate** an **original** Diebold-Nixdorf **stick**, the perpetrators build **tailored malware** and **embedded** it **in** the **ProBase** software stack.
- **Bypassing** the **CrypTA** stick USB **vendor ID checks** → A **malware** called **TRICK.DLL hooks** original functions **GetStickById()** and **GetAllSticksById()** inside **USERAUTH.DLL**
- **Seized** attacker **sticks** used **SanDisk** USB-Devices (VendorID 0x781) to **simulate** original **CrypTA** Sticks by **faking Vendor ID** to 0x64F.



Faking the CrypTA USB stick 2/4

- To **bypass** the **CrypTA** stick **Certificate and PIN validation** a **JAVA class** comes in place called **MyAgent.Jar** to **overwrite** JAVA **bytecode at runtime with** the help of a special JAVA **library** called **javassist.jar**.
- **Javassist provides** the **ability** to **change** the **implementation of** a **class at runtime**, where **bytecode** can be **manipulated when** the **JVM loads it**.
- In order to make this attack working **attackers installed** their **own JDK(1.8.0.181)** and **placed** the library **javassist.jar** in the **./jre/lib/ext** directory
- **MyAgent.jar** is **linked inside** the Windows **Registry** at →
  - HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment_JAVA_OPTIONS => -javaagent:C:\Temp\JAVA\MyAgent.jar



Faking the CrypTA USB stick 3/4

- **To bypass certificate validation MyAgent.jar overwrites InstrumentGetOIDValue() with its own function**

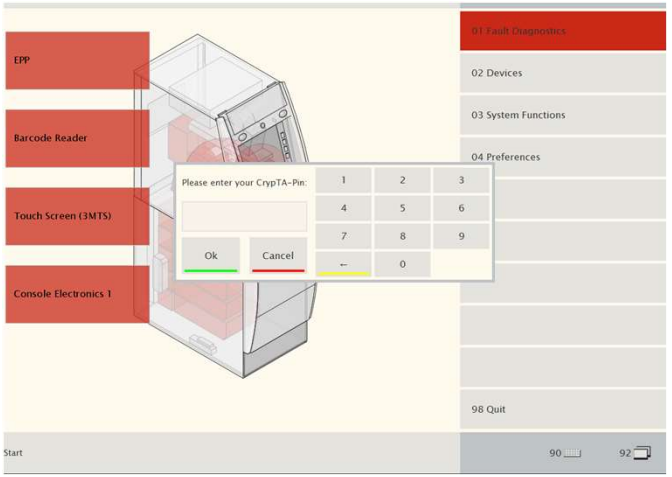
```
    if (dottedClassName.endsWith("com.wincornixdorf.userauth.verificationServer.CertCheckServer")) {
        return this.InstrumentGetOIDValue(dottedClassName, loader);
```

```
private byte[] InstrumentGetOIDValue(String className, ClassLoader loader) {
    try {
        System.out.println(className);
        ClassPool pool = ClassPool.getDefault();
        pool.appendClassPath((ClassPath)new LoaderClassPath(loader));
        CtClass cc = pool.get(className);
        CtMethod cm = cc.getDeclaredMethod("getOIDValue");
        MyUtils.AssertProblemMethod(cm);
        cm.setBody("{ String allowString = \"allow:1.3.6.1.4.1.14760.2.4.1,allow:1.3.6.1.4.1.14760.2.4.2,allow:1.3.6.1.4.1.14760.2.4.3.1
        return cc.toBytecode();
    }
    catch (IOException | CannotCompileException | NotFoundException throwable) {
        return null;
    }
}
```

Faking the CrypTA USB stick 4/4

- **To bypass PIN validation MyAgent.jar overwrites InstrumentCheckPin() and InstrumentVerify() with its own functions**

```
if (dottedClassName.endsWith("com.wincornixdorf.userauth.verificationClient.BaseClient")) {
    return this.InstrumentCheckPin(dottedClassName, loader);
}
if (dottedClassName.endsWith("com.wincornixdorf.diagserv.crypta.impl.CryptaService")) {
    return this.InstrumentVerify(dottedClassName, loader);
}
```

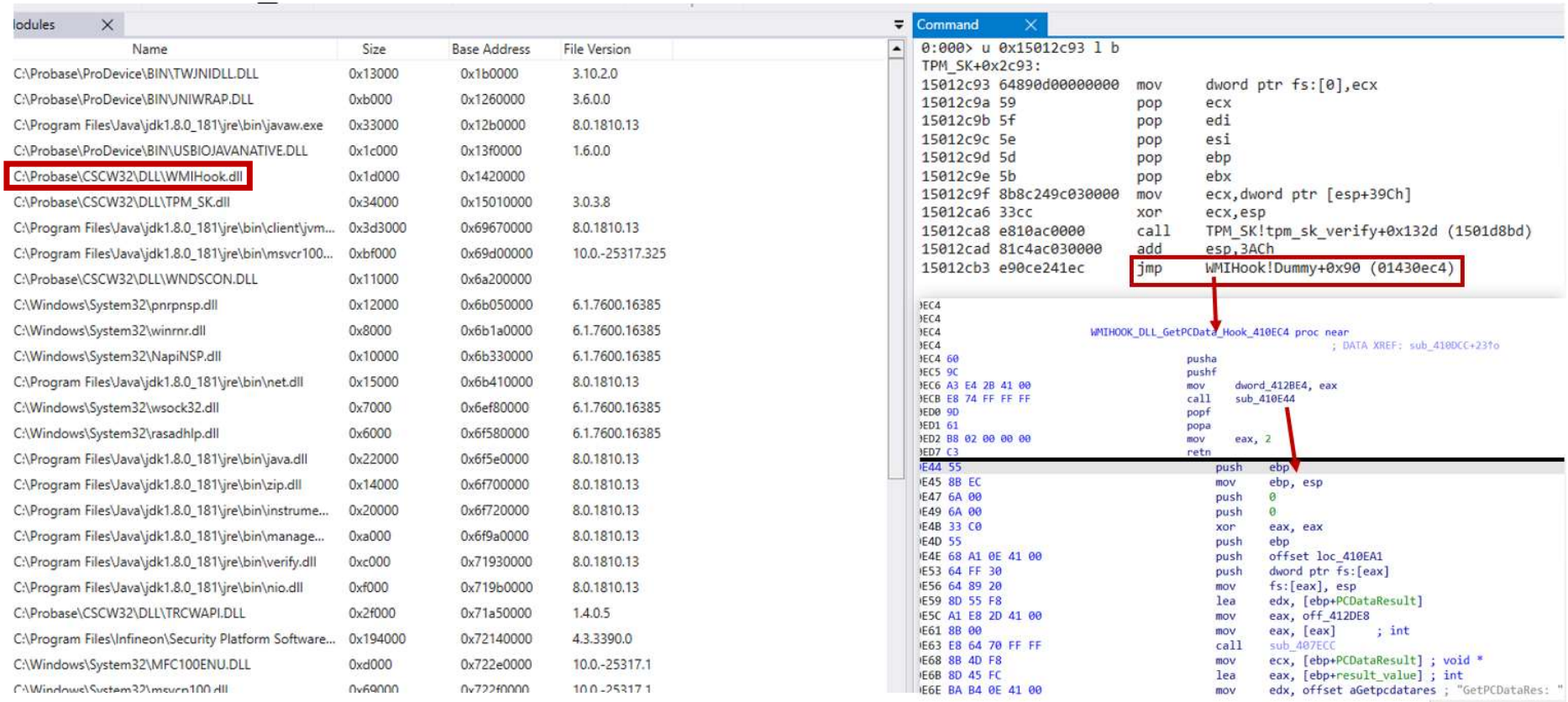



```
private byte[] InstrumentVerify(String className, ClassLoader loader) {
    try {
        System.out.println(className);
        ClassPool pool = ClassPool.getDefault();
        pool.appendClassPath((ClassPath)new LoaderClassPath(loader));
        CtClass cc = pool.get(className);
        CtMethod cm = cc.getDeclaredMethod("verify");
        MyUtils.AssertProblemMethod(cm);
        cm.setBody("{ return; }");
        return cc.toBytecode();
    }
    catch (IOException | CannotCompileException | NotFoundException throwable) {
        return null;
    }
}
```

```
private byte[] InstrumentCheckPin(String className, ClassLoader loader) {
    try {
        System.out.println(className);
        ClassPool pool = ClassPool.getDefault();
        pool.appendClassPath((ClassPath)new LoaderClassPath(loader));
        CtClass cc = pool.get(className);
        CtMethod cm = cc.getDeclaredMethod("checkPIN");
        MyUtils.AssertProblemMethod(cm);
        cm.setBody("{ return true; }");
        return cc.toBytecode();
    }
    catch (IOException | CannotCompileException | NotFoundException throwable) {
        return null;
    }
}
```

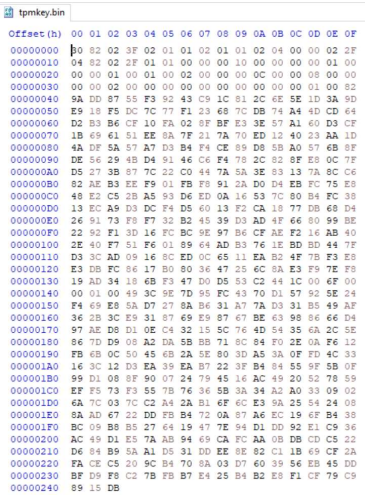
Getting around the original hardware check

- To **bypass** the **requirement** of **original hardware for** successful **key pairing** between dispenser and the blackbox the **malicious library WMIHOOK.DLL fakes** the system **responses** by **hooking** the **GetPcData()** function inside **TPM_SK.DLL**

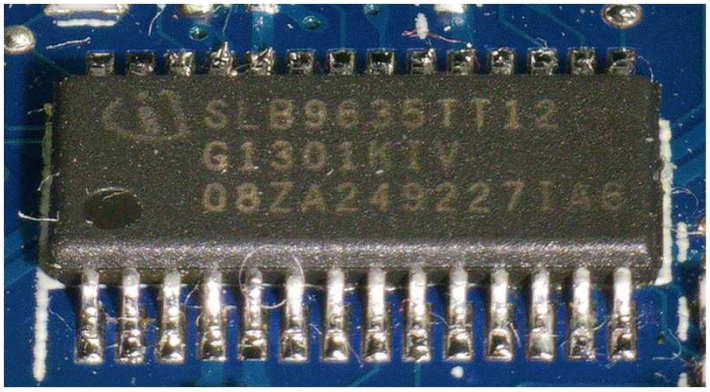


TPM chip migration for successful key exchange

- However, a simple **fake response by** the **WMIHOOK.DLL** is **not sufficient for** a valid **key exchange**, since important **secrets** are **stored in** a **TPM chip** on the **ATM PC**.
- **During** the **manufacturing process** of the ATM, a **keypair** is **generated in** the **TPM using** the **Tspi_Key_CreateKey()** function. This **also creates** a **keyblob** (**tpmkey.bin**) which is then **stored in** the **Probase directory**.
- **To open** the **keystore** the **blob** tpmkey.bin is **loaded into** the **TPM** using **Tspi_Context_LoadKeyByBlob()** and the **necessary** SHA1 based **password is generated** afterwards.
- Only **if** the **key pair in** the **TPM chip and** the **originally generated key blob match**, a new **key exchange** between ATM PC and dispenser **can take place** via T/SOP.
- For this purpose, an **original TPM chip** of an ATM was **desoldered and mounted on** the **blackbox notebook motherboard**. In **combination with** the **matching** tpmkey.bin **blob**, the **perpetrators managed** a successful **key exchange**.

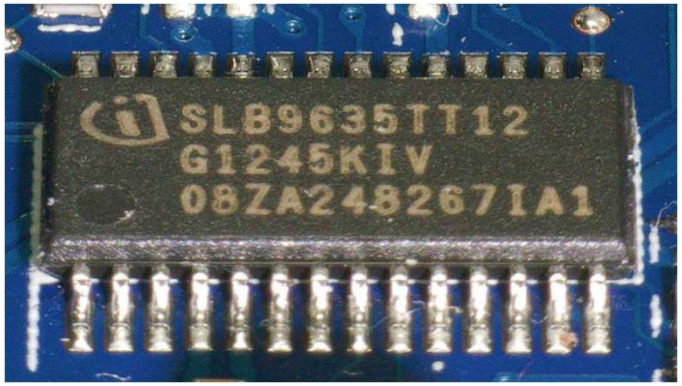


Sloppy applied **solder joints** make it **easy to spot** the migration of the TPM chip to the attacker's notebook.



VS

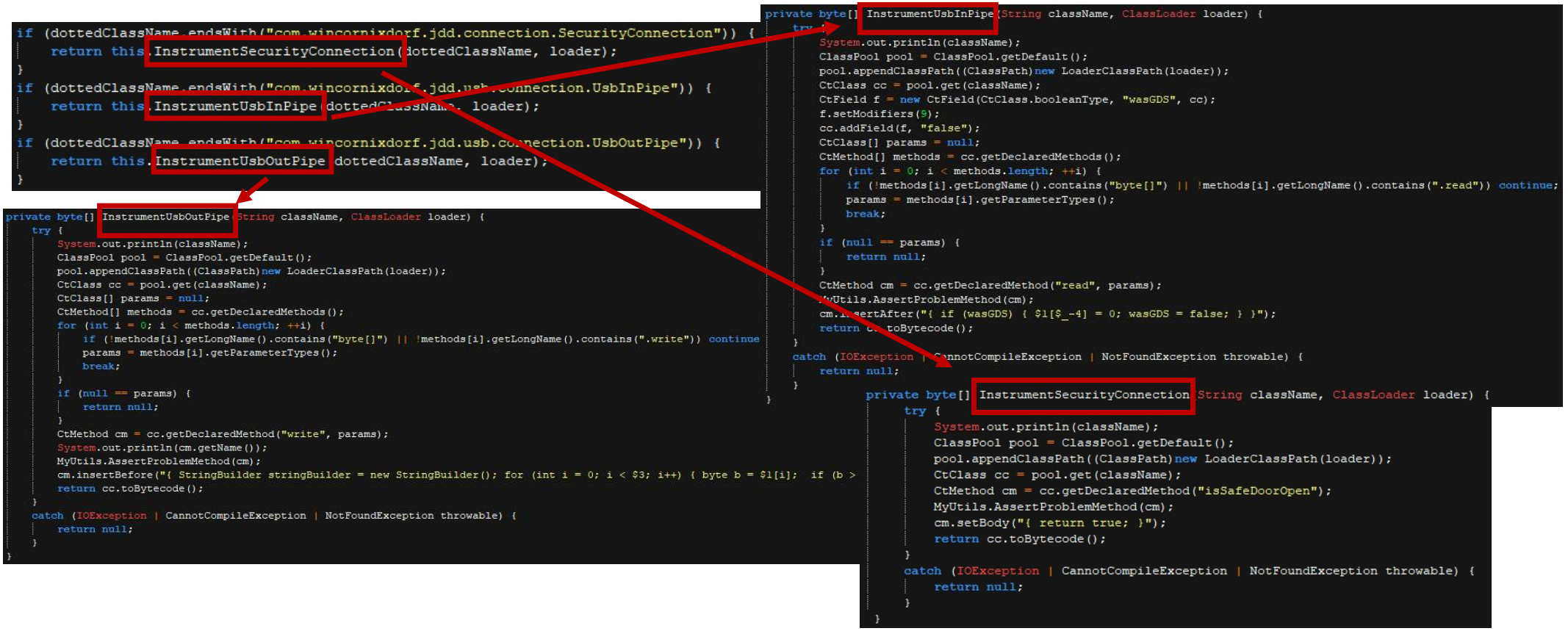
For comparison, a **TPM chip** on a motherboard **without manipulation**.



Standard equipment for chip desoldering or mounting

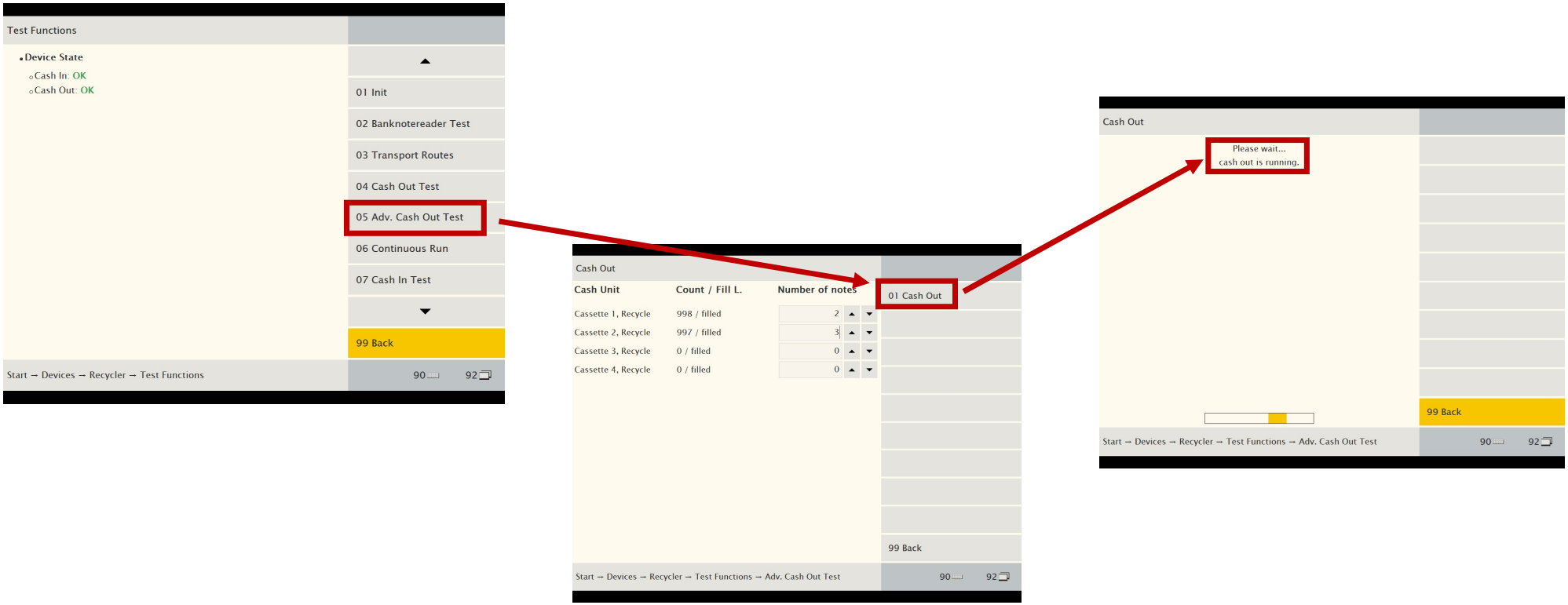
T/SOP open vault door check bypass

- To **bypass** the **T/SOP requirement** to have the **vault door open** in order **to** successfully **achieve** a **key pairing**, **three** original **functions get overwritten by MyAgent.jar →InstrumentSecurityConnection(), InstrumentUsbOutPipe(), InstrumentUsbInPipe()**



T/SOP Cash Out process

- **After** attackers managed a successful **key exchange**, a **newer firmware** (RM3 2632 from 2019) needs to be **installed** in order **to** successfully **cashout**.
- **For** the **cashout**, the **perpetrators did not use malware**, as usually common, **but** rather **functions within** the **T/SOP** that were **designed for cashout testing purposes**.



Known attacks and arrests

- Known **countries where attacks related to** this **modus operandi occurred** => Germany, Switzerland, Czech Republic, Netherlands, Spain, Austria, France, Slovakia, Slovenia, Poland, Denmark, Italy, Estonia and Latvia
- **In July 2021, a group of perpetrators was arrested:**
  - https://www.europol.europa.eu/newsroom/news/russian-speaking-hackers-arrested-in-poland-over-atm-jackpotting-attacks
  - https://therecord.media/belarusian-nationals-arrested-over-atm-black-box-attacks/

EUROPOL

ABOUT EUROPOL | ACTIVITIES & SERVICES | CRIME AREAS & TRENDS | PARTNERS & AGREEMENTS | CAREERS & PROCUREMENT | NEWSROOM | PUBLICATIONS & DOCUMENTS

HOME › NEWSROOM › RUSSIAN-SPEAKING HACKERS ARRESTED IN POLAND OVER ATM JACKPOTTING ATTACKS

Automated translation

RUSSIAN-SPEAKING HACKERS ARRESTED IN POLAND OVER ATM JACKPOTTING ATTACKS

THIS NEWS/PRESS RELEASE IS ABOUT
CYBERCRIME
View all crime areas

29 July 2021
Press Release

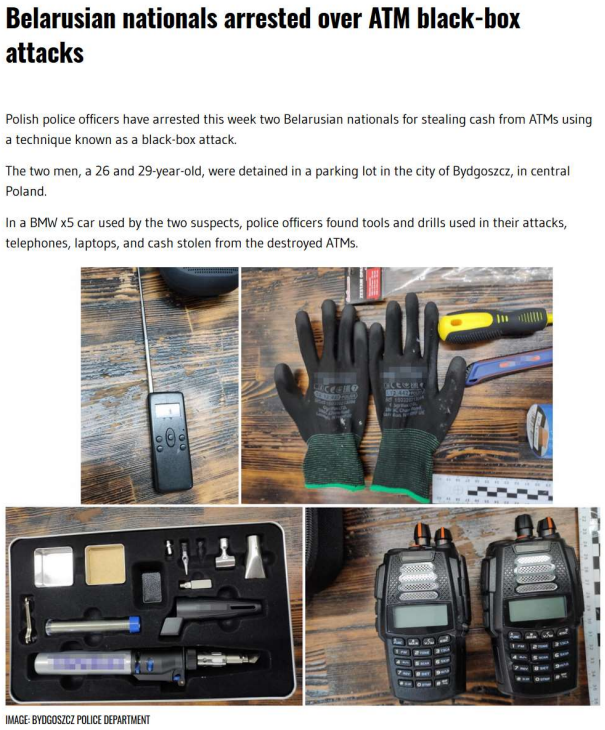
With the support of Europol, the Polish authorities have arrested two individuals committing so-called 'Black Box' attacks against ATMs, in which criminals connect electronic devices to a cash machine and remotely force it to spew out all its cash.

The two suspects - both Belarusian nationals, were arrested in Warsaw on 17 July.

The investigation uncovered that these criminals committed dozens of ATM attacks in at least seven European countries, stealing an estimated €230 000 in cash. The criminals were always targeting the same brand and model of ATM.

The criminals would gain access to the ATM wires by drilling holes or melting parts of it in order to physically connect the machine to a laptop which was then used to send relay commands that caused the machine to dispense all its cash.

EUROPOL SUPPORT



Belarusian nationals arrested over ATM black-box attacks

Polish police officers have arrested this week two Belarusian nationals for stealing cash from ATMs using a technique known as a black-box attack.

The two men, a 26 and 29-year-old, were detained in a parking lot in the city of Bydgoszcz, in central Poland.

In a BMW x5 car used by the two suspects, police officers found tools and drills used in their attacks, telephones, laptops, and cash stolen from the destroyed ATMs.

IMAGE: BYDGOSZCZ POLICE DEPARTMENT

Indicators of Compromise

Hash	Filename
32f5d89d4431c50f09cff7d9c21eb757537db4b3	MyAgent.jar
159de8d420d409546ed99d6d9244bf6f40598567	TRICK.DLL (Delphi)
e68c3bcd0fdd7439aecf668ad0960b8a1437e841	WMIHOOK.DLL (Delphi)

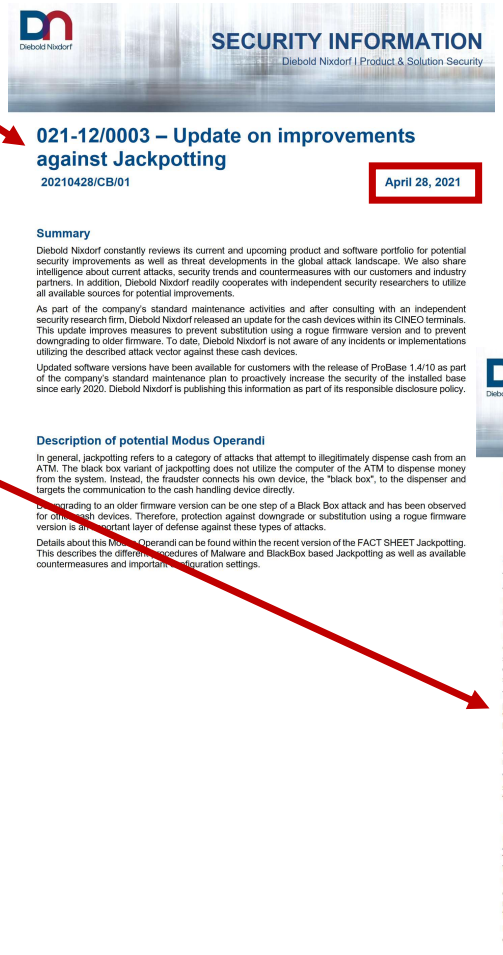
```
import "pe"
rule ATM_CINEO4060_Blackbox {
    meta:
        description = "Detects Malware samples for Diebold Nixdorf CINEO 4060 ATMs used in blackboxing attacks across Europe since May 2021"
        author = "Frank Boldewin (@r3c0nst)"
        date = "2021-05-25"
    strings:
        $MyAgent1 = "javaagentsdemo/ClassListingTransformer.class" ascii fullword
        $MyAgent2 = "javaagentsdemo/MyUtils.class" ascii fullword
        $MyAgent3 = "javaagentsdemo/SimplestAgent.class" ascii fullword
        $Hook = "### [HookAPI]: Switching context!" fullword ascii
        $Delphi = "Borland\\Delphi\\RTL" fullword ascii
        $WMIHOOK1 = "TPM_SK.DLL" fullword ascii
        $WMIHOOK2 = "GetPCData" fullword ascii
        $WMIHOOK3 = {60 9C A3 E4 2B 41 00 E8 ?? ?? ?? ?? 9D 61 B8 02 00 00 00 C3} //Hook function
        $TRICK1 = "USERAUTH.DLL"  fullword ascii
        $TRICK2 = "GetAllSticksByID"  fullword ascii
        $TRICK3 = {6A 06 8B 45 FC 8B 00 B1 4F BA 1C 00 00 00}  //Hook function
    condition:
        (uint16(0) == 0x4b50 and filesize < 50KB and all of ($MyAgent*)) or
        (uint16(0) == 0x5A4D and (pe.characteristics & pe.DLL) and $Hook and $Delphi and all of ($WMIHOOK*) or
        all of ($TRICK*))
}
```

Recommendations and countermeasures

- In a **security information** at the **end of April 2021**, **Diebold Nixdorf informed about** a **potential blackboxing attack vector on RM3 and CMD-V5 dispensers**.
- Interestingly, the **first attacks on CINEO 4060** ATM with RM3 dispensers were **launched only** a **short time after publication**.
- It can be **assumed** that **perpetrators read** this **alert** and **reacted** to it **immediately** because they were **aware** that the **recommendations** by Diebold Nixdorf would also **render their attack obsolete**.
  - DN Security Alert related to this excellent PTSecurity research → https://hardwear.io/netherlands-2021/presentation/Blackboxing-Diebold-Nixdorf-ATMs.pdf
- **Shortly after** the **first attacks** became public, **Diebold Nixdorf warned in** an **Active Security Alert** now to **implement** the **recommended measures** as **quickly** as possible.
- **Especially** the **firmware fusing protects** from **downgrading** to an older version and **makes** this **attack obsolete**.

Suggested measures by Diebold Nixdorf

- For all CINEO terminals with CMD-V5 dispenser or RM3 recycler, verify the implementation of the following:
  - ProBase/C 1.4/10
  - ProBase/C 1.4/xx with HotInfo 4564
  - ProBase/C 1.3/xx with HotInfo 4563
  - Enable firmware fuse (SECURITY_LEVEL=ENHANCED)
  - Enable physical authentication for TDC (HWAUTHMODE_BKE=EXTENDED)



SECURITY INFORMATION

Diebold Nixdorf | Product & Solution Security

021-12/0003 – Update on improvements against Jackpotting

20210428/CB/01

April 28, 2021

Summary

Diebold Nixdorf constantly reviews its current and upcoming product and software portfolio for potential security improvements as well as threat developments in the global attack landscape. We also share intelligence about current attacks, security trends and countermeasures with our customers and industry partners. In addition, Diebold Nixdorf readily cooperates with independent security researchers to utilize all available sources for potential improvements.

As part of the company's standard maintenance activities and after consulting with an independent security research firm, Diebold Nixdorf released an update for the cash devices within its CINEO terminals. This update improves measures to prevent substitution using a rogue firmware version and to prevent downgrading to older firmware. To date, Diebold Nixdorf is not aware of any incidents or implementations utilizing the described attack vector against these cash devices.

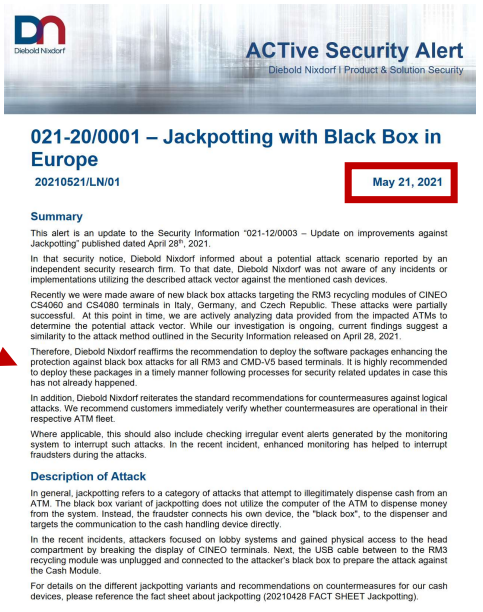
Updated software versions have been available for customers with the release of ProBase 1.4/10 as part of the company's standard maintenance plan to proactively increase the security of the installed base since early 2020. Diebold Nixdorf is publishing this information as part of its responsible disclosure policy.

Description of potential Modus Operandi

In general, jackpotting refers to a category of attacks that attempt to illegitimately dispense cash from an ATM. The black box variant of jackpotting does not utilize the computer of the ATM to dispense money from the system. Instead, the fraudster connects his own device, the "black box", to the dispenser and targets the communication to the cash handling device directly.

Downgrading to an older firmware version can be one step of a Black Box attack and has been observed for other cash devices. Therefore, protection against downgrade or substitution using a rogue firmware version is an important layer of defense against these types of attacks.

Details about this Modus Operandi can be found within the recent version of the FACT SHEET Jackpotting. This describes the different procedures of Malware and BlackBox based Jackpotting as well as available countermeasures and important configuration settings.



ACTive Security Alert

Diebold Nixdorf | Product & Solution Security

021-20/0001 – Jackpotting with Black Box in Europe

20210521/LN/01

May 21, 2021

Summary

This alert is an update to the Security Information "021-12/0003 – Update on improvements against Jackpotting" published dated April 28th, 2021.

In that security notice, Diebold Nixdorf informed about a potential attack scenario reported by an independent security research firm. To that date, Diebold Nixdorf was not aware of any incidents or implementations utilizing the described attack vector against the mentioned cash devices.

Recently we were made aware of new black box attacks targeting the RM3 recycling modules of CINEO CS4060 and CS4080 terminals in Italy, Germany, and Czech Republic. These attacks were partially successful. At this point in time, we are actively analyzing data provided from the impacted ATMs to determine the potential attack vector. While our investigation is ongoing, current findings suggest a similarity to the attack method outlined in the Security Information released on April 28, 2021.

Therefore, Diebold Nixdorf reaffirms the recommendation to deploy the software packages enhancing the protection against black box attacks for all RM3 and CMD-V5 based terminals. It is highly recommended to deploy these packages in a timely manner following processes for security related updates in case this has not already happened.

In addition, Diebold Nixdorf reiterates the standard recommendations for countermeasures against logical attacks. We recommend customers immediately verify whether countermeasures are operational in their respective ATM fleet.

Where applicable, this should also include checking irregular event alerts generated by the monitoring system to interrupt such attacks. In the recent incident, enhanced monitoring has helped to interrupt fraudsters during the attacks.

Description of Attack

In general, jackpotting refers to a category of attacks that attempt to illegitimately dispense cash from an ATM. The black box variant of jackpotting does not utilize the computer of the ATM to dispense money from the system. Instead, the fraudster connects his own device, the "black box", to the dispenser and targets the communication to the cash handling device directly.

In the recent incidents, attackers focused on lobby systems and gained physical access to the head compartment by breaking the display of CINEO terminals. Next, the USB cable between to the RM3 recycling module was unplugged and connected to the attacker's black box to prepare the attack against the Cash Module.

For details on the different jackpotting variants and recommendations on countermeasures for our cash devices, please reference the fact sheet about jackpotting (20210428 FACT SHEET Jackpotting).

Conclusion

- This **case** impressively **illustrates** the **intensity** with which **perpetrators attempt** to **crack even well-protected devices**.
- Especially the **reverse engineering** and **research work** performed to gain all these deep insights into Diebold Nixdorf's platform software **is remarkable**.
- However, it also **shows** that the **effort and time required** by the perpetrators **for** a **successful attack** is becoming **increasingly** complex Additionally it **raises** the **risk for money mules getting busted** when the **dwell time in** the **bank's foyer grows**.
- If **modern devices** are used and **manufacturers**' **security recommendations** are **implemented consistently**, the perpetrators' **success rate decreases significantly**.

Stay secure!