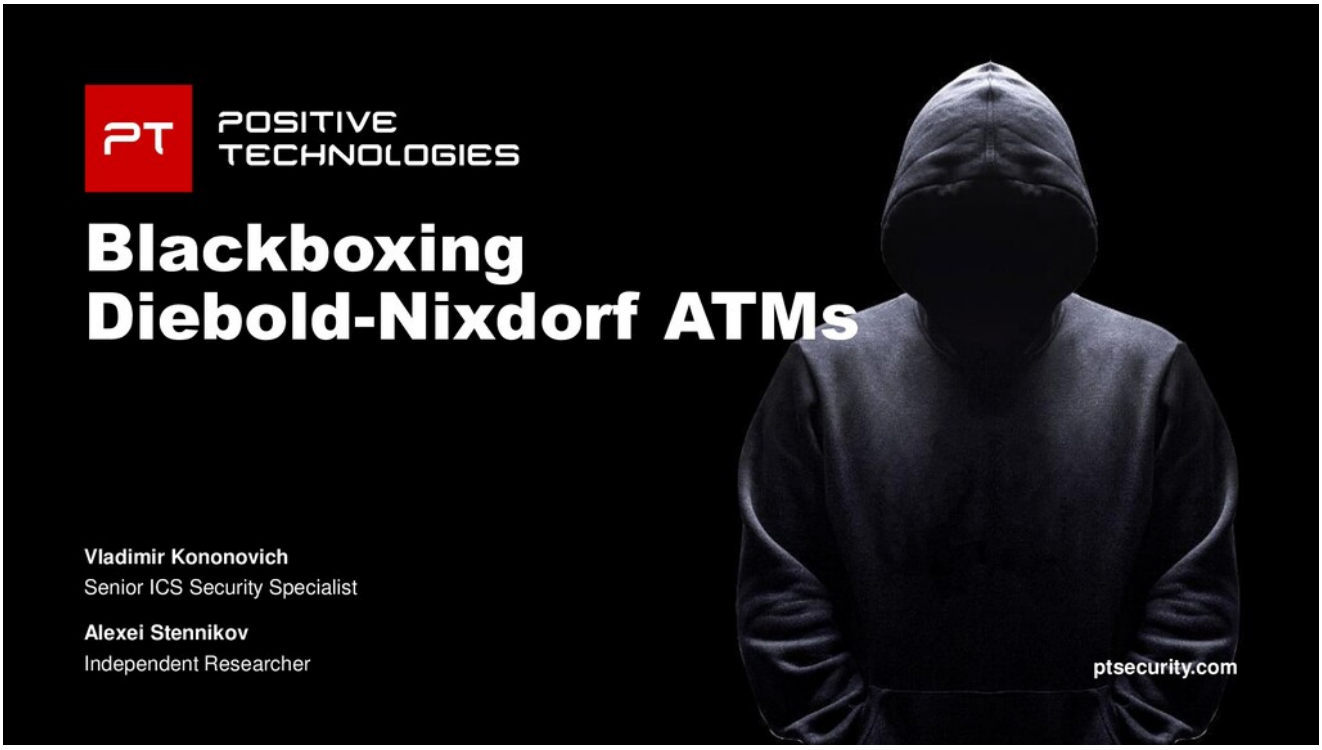
Blackboxing Diebold-Nixdorf ATMs

speakerdeck.com/ptswarm/blackboxing-diebold-nixdorf-atms



November 04, 2021

Other Decks in Research

See All in Research

ACTBE Inc. Company Deck FY2022

 actbeinc

0

120

素人発想玄人実行2.0

 hf149

0

1.2k

企業の業界分類予測における共変量シフト問題の抑制

 taro_masuda

2

660

#osc22on 中国オープンソースムーブメントの盛り上がりと 中国最大のオープンソー…

 takasumasakazu

0

160

AI最新論文読み会2022年4月

 ailaboocu

0

330

JGS594 Lecture 18

 javiergs

PRO

0

340

会社訪問アプリ「Wantedly Visit」のデータで見る相互推薦システム / deim2022-rrs-w…

 yuya4

0

750

ビジネス電話応対における音声感情認識

 ken57

0

240

ライティング支援のための文法誤り訂正

 chemical_tree

1

1k

AIOps研究録—SREのためのシステム障害の自動原因診断 / SRE NEXT 2022

 yuukit

5

1.8k

研究紹介2022年度版

sh01k

0

330

行政のオープンネスとフェアネスーデジタル庁でDXに取り組む『民間採用人材』の…


halsk

0

520

Featured

See All Featured

Clear Off the Table

cherdarchuk

79

280k

Happy Clients

 brianwarren

89

5.6k

Typedesign – Prime Four

 hannesfritz

33

1.3k

Optimizing for Happiness

 mojombo

365

63k

Robots, Beer and Maslow

 schacon

152

7.1k

Thoughts on Productivity

 jonyablonski

43

2.2k

Infographics Made Easy

 chrislema

233

17k

Web Components: a chance to create the future

 zenorocha

303

40k

WebSockets: Embracing the real-time Web

 robhawkes

57

5k

Documentation Writing (for coders)

 carmenhchung

48

2.5k

jQuery: Nuts, Bolts and Bling

 dougneiner

56

6.4k

Keith and Marios Guide to Fast Websites

 keithpitt

404

21k

Transcript

1. **ptsecurity.com Blackboxing Diebold-Nixdorf ATMs Vladimir Kononovich Senior ICS Security Specialist**

   Alexei Stennikov Independent Researcher

2. **Who are we? Vladimir Kononovich: •Reverse-engineering (since 2008) •Romhacking (my**

   hobby) •Writing tools for IDA/Ghidra •Ghidra ideologist

3. **Who are we? Alexei Stennikov: •Hardware expert •ICS/SCADA security researcher**

   •ATM/POS security researcher •Some skills of RE

4. **ATM hardware internals •Less-secure upper part •Safe-zone (lower part) Safe-zone**

   includes a dispenser controller

5. **Our previous talk at hw.io •ATM internals •ATM attacks types**

   •What is Blackbox attack? •NCR dispensers vulnerability Our hardwear.io 2018 talk (youtube) https://www.youtube.com/watch?v=L5yl4A1npVU

6. **Paderborn, we have a problem • FW downgrade • Modified**

   FW uploading • SmartCard DoS “feature” • Encryption bypass • Withdrawal

7. **RM3/CMDv5 firmware files •BTR (bootloader) •FRM (main firmware) Parts: •**

   RM3_CRS.BTR / CD5_ATM.BTR • RM3_CRS.FRM / CD5_ATM.FRM Files: • Device id • Product id • Vendor id • ? • “UFD” • ? • CRC32 • Some size • Firmware part name The rest is encrypted. No chance to decrypt. Thank you for watching! Bye:)

8. **But wait… eBay can help us! Again…**

9. **Demo**

10. **JTAG: Identifying connector & pins 1 • VREF • VSUPPLY**

    2 3 • nRST • GND 4 5 • TDI • GND 6 7 • TMS • GND 8 9 • TCK • GND 10 11 • RTCK • GND 12 13 • TDO • GND 14 15 • nRST • GND 16 17 • DBGRQ • GND 18 19 • DGBACK • GND 20

11. **Another interesting place: Smartcard •USB encryption keys generation •Session numbers/keys**

    storage •Different counters •Certificates storage •A whole system DoS Other “features”:)

12. **Powering and testing FW uploading • + USB connection •**

    + Java-based software (easy to decompile and modify)

13. **Firmware dumping (CMDv5) •Main CPU: STM STR710FZ2T6 •Image base: 0x60000000**

    Two other CPUs: •CollectorBooter: STR730FZ2T6 •DispenseBooter: STR730FZ2T6

14. **Firmware analysis (CMDv5) 1. Read 5 LE-dwords after a MOD**

    name (header-dwords, HD) 2. key[n] = KEY1[n] ^ HD[n]; // where n: 0..3 3. data[0] = KEY0[0] ^ HD[0] ^ HD[1]; data[1] = KEY0[1] ^ HD[2] ^ HD[3]; Encryption algo – XTEA mod. DELTA: 0xF27716BA. Rounds: 32 - KEY0 and KEY1 are unknown yet! Init:

15. **Firmware analysis (CMDv5) Decryption algo XTEA (Python): Our python implementation**

16. **Firmware analysis (CMDv5) Decryption result: •Sequential APLib archives (have AP32**

    header) •Ends with 0xFFFFFFFFs •Unpacked firmware

17. **Firmware analysis (CMDv5) KEY0 and KEY1: •Hardcoded! (base offset: 0x64000000)**

    •Ability to use OLD or ZEROed keys!

18. **Firmware analysis (CMDv5) Self-signing (bad practice) • 30-bit tokens count**

    (int length + 1) • 30-bit tokens count (int length) • sign = RSA(e=7, SHA1(data[0x360:])) • modulus = RSA.key.N •0x160 – sign •0x260 – modulus •0x360 - data

19. **Firmware analysis (CMDv5) Firmware uploading (DFU) •Uses special DFU device:**

    • - DFU_PID = PID | 0x8000 • - bInterfaceClass = 0xFF • - bInterfaceSubClass = 1 Normal state DFU-mode state

20. **Firmware analysis (CMDv5) Firmware encryption tricks Firmware header • Unpacked**

    FW size • Firmware part name • 5 header-dwords • 0xDEAD0000 | (KEY0_OFFSET / 8) Old keys checking code

21. **Firmware analysis (CMDv5) KEY0 and KEY1: •Hardcoded! (base offset: 0x64000000)**

    •Ability to use OLD or ZEROed keys!

22. **Firmware analysis (summary) What we know: 1. Self-signed firmware (public**

    key is in the same binary!) 2. APLib packed sequential blocks 3. Modified XTEA encryption algorithm (different DELTA) 4. XTEA encryption keys can be bypassed (VULN IS HERE!) 5. DFU protocol (uploading firmware into a dispenser)

23. **USB Communications (steps) 1.Basekey initialization 2.New session keys generation 3.Session**

    counters synchronizing

24. **USB Communications (Basekey init) To generate a new Basekey you**

    need: 1. ROOT-certificate 2. Intermediate CA-certificate 3. Terminal Encryption certificate (issued by CA) 4. Terminal Authentication certificate (issued by CA) We don’t have any of them… :((and don’t need them)

25. **USB Communications (session key) How to generate a new session**

key (PC): 1. BK = Read the Basekey from the Keystorage (its key in TPM) 2. SESSION_KEY_XXX = SHA1(BK) + session_counter + direction We have four directions: PC_FW_OUT, PC_FW_IN, FW_PC_OUT, FW_PC_IN SmartCard also checks for the same counter usage + makes its increment How to generate a new session key (Firmware): 1. SESSION_KEY_XXX = SmartCard(session_counter + direction)

26. **USB Communications (session sync) To synchronize session counters you need:**

1. ChannelID (server=2, client=1) 2. Basekey length 3. Basekey Check Value (KCV) (first 3 bytes of SHA1(Basekey) 4. Session counters for USB client/server IN/OUT Basekey can be read from the Keystorage file too Response has the same parameters so we can sync session counters

27. **Abusing session counters (DoS) Steps to reproduce: 1. session_counter =**

0xFFFFFFFF 2. SESSION_KEY_XXX = SmartCard(session_counter + direction) SmartCard generates a new key, but no new key can be generated after!

28. **USB comms analysis (summary) What we know: 1. TPM usage**

(awesome!) 2. Keystorage usage (awesome!) 3. Four encryption keys directions (awesome!) 4. SmartCard usage (awesome!) 5. SmartCard “feature” (can disable a whole ATM, but won’t allow to take the money!)

29. **USB Communications (withdrawal) Steps to perform a withdrawal: 1. Patch**

FW to skip asking SmartCard for a session key (use some dummy array) 2. Patch Java code to use the same dummy array as the key 3. Patch Java code to skip checks for cashIn and cashOut configs 4. Sync session counters (PC = SmarCard) 5. Write a new cassettes config to the dispenser’s EEPROM 6. Call prepareCashOut() 7. Call cashOut(cassetteNum=3, banknotesNum=5) 8. Call shutter.open() 9. Take the money! 10.Close the shutter

30. **Vulnerabilities disclosure timeline 1. Q3 2018 – vendor has been**

informed about vulnerabilities 2. Q4 2018 – official PoC tests were performed, vulnerabilities have been proven 3. Q4 2018 – CVE IDs were registered 4. Q1 2021 – vendor informed us that vulnerabilities were fixed in 2019 5. Q3 2021 – <Russian Mitre> IDs: - BDU:2021-04967 - BDU:2021-04968

31. **Thank you Contacts: vkononovich@ptsecurity.com**