Offensive Payment Security

Advanced Techniques and Process

whoami?

ROHITH RAJAN

CPT | CPTA v.2 | CSA | eCIR | CEHv.12 | SSCP | CCTP | OFSA | ADCD | DFE | DFIR | PMP | ISC2 (CC) | HOF in NASA | ISO27001 LA

CTO @ Intelligent Cyber Solutions , Technical Advisor @DEFCON bringing six years of cyber security expertise and a unique background as a reformed Blackhat hacker. With extensive experience working alongside intelligence agencies to lead innovative cyber defense strategies.

01

Introduction to Payment Security

The Growing Importance of Payment System Security

Payment systems are the backbone of the global economy, enabling billions of transactions every day across various platforms—whether through credit cards, mobile wallets, or online banking. As digital transactions increase, so does the risk of financial data theft and fraud. The importance of securing these systems cannot be overstated for several reasons:

- **Massive Transaction Volumes:** Global digital payments are expected to surpass $10 trillion annually, making them an attractive target for attackers.

- **Sensitive Data:** Payment systems handle highly sensitive information like credit card details, personal identities, and transaction histories, which can be exploited by malicious actors if not properly protected.

- **Reputational Damage:** A breach can destroy customer trust, leading to a loss of business, regulatory fines, and significant damage to the brand.

- **Legal and Compliance Requirements:** Organizations are subject to stringent regulations such as PCI-DSS, GDPR, and CCPA, which mandate the protection of customer data in payment systems.

Securing these systems is critical not only to prevent financial loss but also to maintain trust, comply with regulations, and avoid severe penalties.

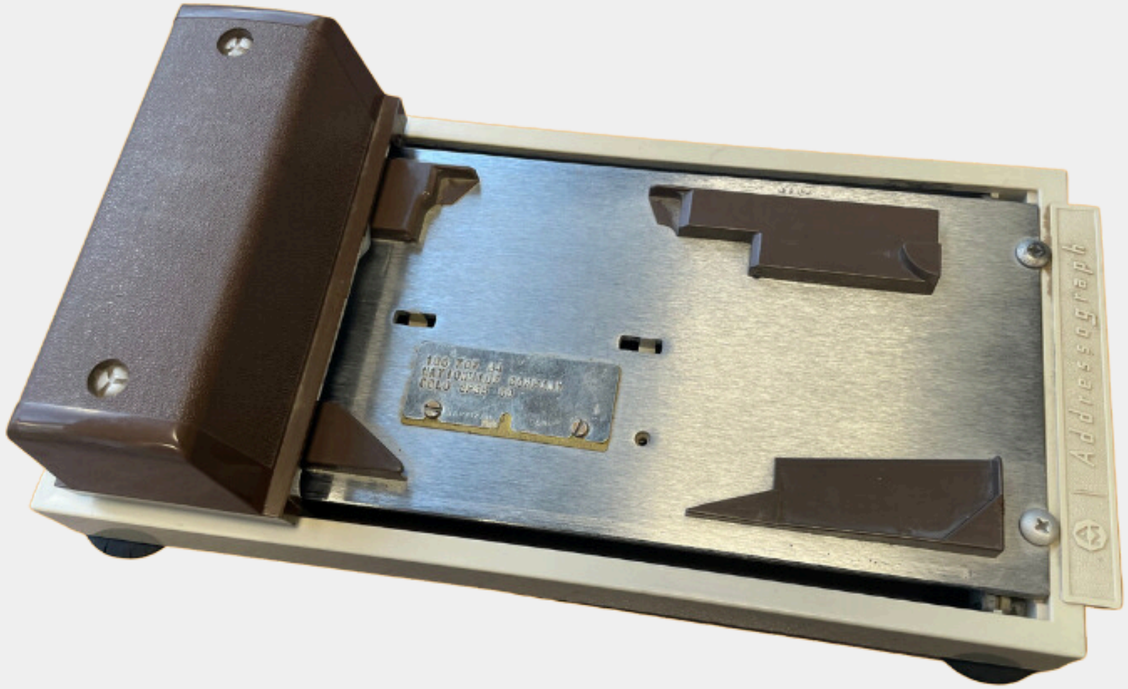
What’s in the cards?

Billions and billions of cards have been produced over the decades, and while the dimensions have largely remained the same, new technology, design trends and concerns about accessibility and sustainability have evolved how they look, feel and work.

The pandemic-fueled acceleration of contactless payments has led to a huge uptick in mobile payments, and now people can even choose to forgo physical cards entirely. More than 250 banks and other customers are offering digital-first issuance through Mastercard, where virtual cards can be issued in real time and used immediately, with a physical card optional. But physical card issuance remains robust — a tangible token in an increasingly virtual world.

Making an impression: Embossing

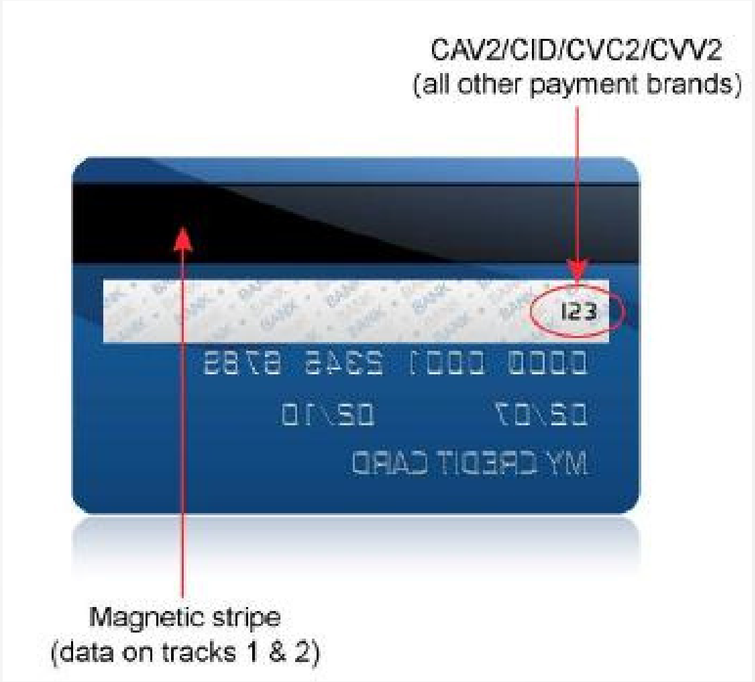
The earliest modern credit cards were made of cardboard and printed with the cardholder's name. As the concept of consumer credit expanded in the 1950s and 1960s, banks and merchants needed a more efficient way of processing transactions. Enter embossed plastic cards and the card imprinter, also called a zip-zap machine for the action of running the handle back and forth over the card to imprint the information onto carbon-paper packets. (It was also nicknamed the knuckle-buster by clumsy cashiers, for the layers of skin occasionally sacrificed to the gods of commerce.)



A swipe at security: The magnetic stripe

A 1960s invention largely credited to IBM, magnetic tape contained encoded card information that could be read and relayed by point-of-sale terminals with a simple swipe. Magnetic stripes remained in popular use for decades until the popularization of chip cards in the 1990s and 2000s. In fact, magnetic stripes now have an expiration date: In 2021, Mastercard became the first card network to announce it would no longer require newly issued credit and debit cards to have the stripe starting in 2024 in most markets

History of Payment Systems: From Magnetic Stripe to EMV and Contactless (NFC)



Magnetic Stripe Cards (1960s)

Introduction: Magnetic stripe cards (magstripe) were introduced in the 1960s and became the first widely adopted form of payment card technology.

Key Features:

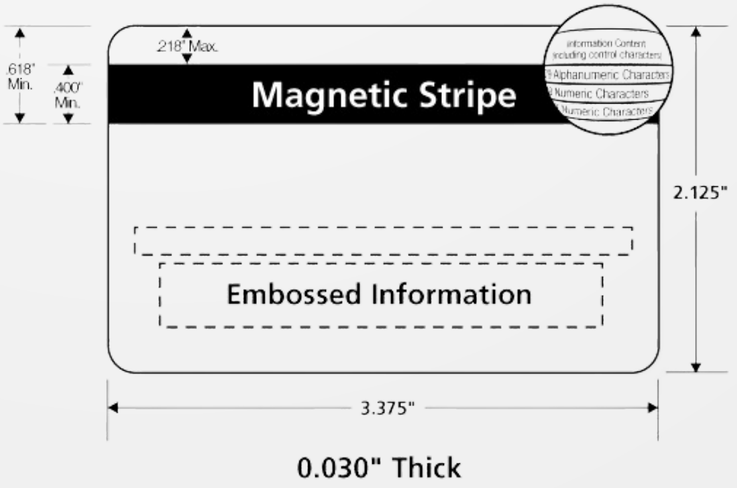
- Stores static data such as the cardholder's name, card number, and expiration date.
- Widely used in ATMs and point-of-sale (POS) systems.

Vulnerabilities:

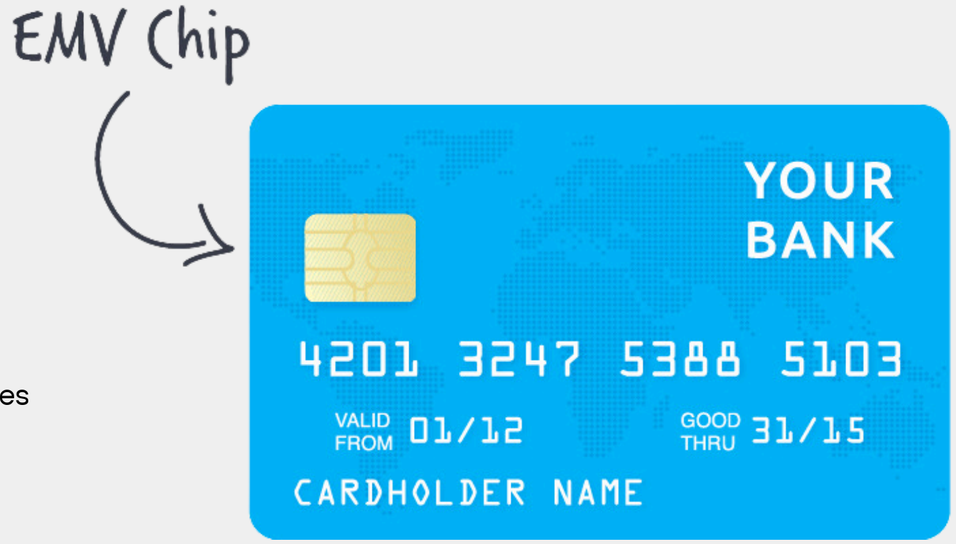
- Magstripe Skimming: One of the major vulnerabilities of magstripe cards is the ease with which data can be cloned. Attackers can install skimming devices on ATMs and POS terminals to capture the information stored on the magnetic stripe.

- Cloning: Because the data on magstripe cards is static (unchanging), it can be copied onto another card, allowing attackers to clone the card and make fraudulent transactions.

- Lack of Encryption: Magnetic stripe technology does not use encryption, leaving sensitive cardholder data exposed during transactions.



The Rise of EMV Smart Cards (1990s)



Introduction: EMV (Europay, MasterCard, Visa) smart cards were introduced in the early 1990s as a more secure alternative to magnetic stripe cards. EMV cards contain a microprocessor chip that stores and processes data, providing enhanced security features for payment transactions.

Key Features:

- Dynamic Data: Unlike magstripe cards, EMV cards generate a unique cryptogram for each transaction, making it difficult for attackers to clone the card or replay transaction data.

- Authentication Methods: EMV cards use various forms of authentication, including PIN verification and cryptographic challenge-response protocols.

Vulnerabilities:

- Cryptogram Replay Attacks: Although EMV chips generate dynamic data, weaknesses in certain implementations have allowed attackers to intercept and replay cryptographic messages in what's known as a "**cryptogram replay attack**."

- Cardholder Verification Bypass: Some vulnerabilities have been identified in the cardholder verification process, allowing attackers to bypass PIN requirements and authorize transactions without proper verification.

- POS Malware: Attackers can still target the POS terminals connected to EMV readers with malware to capture card data before it reaches the secure chip for processing.

Contactless Payments (NFC) – 2000s

The early 2000s saw the rise of Near Field Communication (NFC) technology, enabling contactless payments. This technology allows users to make payments by tapping their card or mobile device against a payment terminal, making transactions quicker and more convenient.

Key Features:

Convenience: NFC allows for quick, tap-to-pay transactions at short range (typically under 10 cm).
Integration with Mobile Wallets: NFC is integrated with mobile payment systems such as Apple Pay, Google Pay, and Samsung Pay, providing a seamless payment experience using smartphones or smartwatches.

Security: Contactless payments use dynamic cryptographic data (similar to EMV), and mobile wallets offer additional security features such as biometric authentication (fingerprint, face recognition).

Vulnerabilities:

- Relay Attacks: NFC technology is susceptible to relay attacks, where an attacker extends the communication range of an NFC card to relay the data to another terminal. This could allow a malicious actor to make unauthorized transactions using a contactless card from a distance.
- Man-in-the-Middle (MitM) Attacks: Although rare, an attacker could intercept NFC communications between a card and a terminal in a MitM attack, potentially manipulating or stealing transaction data.
- Mobile Wallet Exploits: Although mobile wallets are generally secure, vulnerabilities in the underlying software or operating system can expose payment credentials or allow attackers to bypass authentication mechanisms.

Comparative Discussion of Vulnerabilities Across Eras

- Magstripe vs. EMV: Magstripe technology's reliance on static, unencrypted data makes it far more vulnerable to skimming and cloning than EMV, which uses dynamic cryptographic protocols. However, EMV cards are not immune to attacks, as cryptogram replay and verification bypass vulnerabilities have been discovered in some implementations.

- EMV vs. NFC: While both EMV and NFC use dynamic data for securing transactions, NFC introduces additional risks due to its wireless nature. Relay attacks and MitM attacks are more plausible in contactless payments, but NFC's convenience and integration with mobile devices make it popular.

- Evolution of Attacks: As payment technology has evolved, so too have the attack vectors. Skimming and cloning were major issues in the magstripe era, while EMV has forced attackers to explore more sophisticated methods like replay attacks and POS malware. NFC payments, though relatively secure, introduce new risks related to wireless communication and mobile platforms.

Key Vulnerabilities in Payment Systems

Card-Present Transactions

These transactions occur when a physical payment card is presented at the point of sale (POS). Despite improvements in technology like EMV chips, card-present transactions remain vulnerable to several types of attacks.

A.Skimming

Definition: Skimming occurs when a malicious device (a skimmer) is secretly installed on an ATM, gas pump, or POS terminal to steal the information stored on a card's magnetic stripe.
How It Works: Skimmers capture card data during legitimate transactions and store it for later use. The data is typically cloned onto another card and used for fraudulent purchases.
Vulnerable Systems: Older POS systems using magnetic stripe readers are particularly at risk.
Preventive Strategies: Encourage merchants to upgrade to EMV chip-enabled devices and use encryption to secure card data during the transaction process.

B. POS Malware

Definition: POS malware is malicious software installed on point-of-sale systems that collects credit card data in real time.
How It Works: Attackers compromise the POS system and install malware to intercept card data, including magnetic stripe or EMV data, before it is encrypted and sent to the payment processor.
Case Study: The Target breach (2013) involved POS malware that compromised over 40 million credit and debit card numbers by targeting the retailer's payment systems.
Preventive Strategies: Regularly update POS software, implement network segmentation to isolate payment systems, and use point-to-point encryption (P2PE) to protect cardholder data during transmission.

C. Card Cloning

Definition: Cloning refers to copying data from the magnetic stripe of a payment card onto another card, effectively creating a duplicate.
How It Works: Once attackers have card data (often via skimming), they use it to create cloned cards. These clones can then be used for fraudulent purchases at physical stores.
Vulnerable Systems: Magnetic stripe cards are particularly vulnerable to cloning since the data on the stripe does not change between transactions.
Preventive Strategies: Shift to EMV chip cards, which use dynamic data for each transaction, making cloning much more difficult. Monitoring for suspicious transaction activity is also crucial.

Card-Not-Present (CNP) Transactions

CNP transactions occur when the physical card is not required to complete the payment, such as in online or mobile transactions. These transactions have grown significantly with the rise of e-commerce, but they also carry distinct vulnerabilities.

A. Phishing

Definition: Phishing involves tricking users into providing sensitive information, such as credit card numbers, by impersonating legitimate institutions or services.
How It Works: Attackers send emails, texts, or create fake websites that look like legitimate businesses, prompting users to input their card details. This information is then used for fraudulent transactions.
Real-World Example: Phishing schemes targeting popular online shopping sites or payment processors have led to significant fraud, especially during high-traffic shopping seasons like Black Friday.
Preventive Strategies: Educating consumers about phishing threats, implementing email filtering systems, and encouraging the use of multi-factor authentication (MFA) to verify account logins.

B. Insecure APIs

Definition: Payment APIs are software interfaces that allow communication between a website or mobile app and the payment gateway. An insecure API can expose sensitive data or allow unauthorized access to payment information.
How It Works: Attackers exploit weak authentication, improper input validation, or misconfigurations in payment APIs to steal data or perform unauthorized transactions.
Real-World Example: A misconfigured API allowed hackers to steal customer card data from a fintech platform by bypassing the authentication process.
Preventive Strategies: Implement strong authentication (e.g., OAuth), validate all API inputs, and apply encryption for data in transit and at rest. Regular security audits and penetration testing of APIs are also crucial.

C. Mobile Wallet Attacks

Definition: Mobile wallets, like Apple Pay or Google Pay, are digital platforms that store card information for tap-and-go or online payments. While secure, they can still be exploited if not properly configured or if vulnerabilities in the mobile device are leveraged.
How It Works: Attackers may exploit vulnerabilities in NFC communication, install malicious apps to access stored payment credentials, or launch social engineering attacks to gain access to a user's mobile wallet.
Real-World Example: Attackers have exploited flaws in mobile payment systems to make unauthorized transactions by cloning NFC data or intercepting mobile wallet credentials.
Preventive Strategies: Use device-level security features like biometric authentication (fingerprint, face ID) and tokenization, which ensures payment data is not stored on the device but is represented by a token during transactions. Additionally, mobile wallet providers must ensure regular patching and security updates for their apps.

Exploiting Card Readers and Skimming Attacks with Kali Linux Tools

MagSpoof

Purpose: MagSpoof is a hardware tool that emulates magnetic stripe data to spoof magstripes wirelessly. It can be used for card cloning and demonstration of how card skimming attacks work.
Platform: Requires microcontroller hardware (Arduino-based).
Use Case: Skimming and magnetic stripe card cloning demonstrations.

Stripe Snoop

Purpose: Stripe Snoop is a software tool designed to read and decode data from a magnetic stripe reader.
Platform: Runs on Linux, requires a physical magnetic stripe reader to capture data.
Limitations: Does not handle file-based input well, primarily used for hardware data capture.

Proxmark3

Purpose: Proxmark3 is a multi-purpose device for reading, writing, analyzing, and emulating RFID and NFC cards. It can also be used to emulate magnetic stripe functionality for RFID/NFC card systems.
Platform: Requires hardware and firmware setup.
Use Case: Useful for penetration testers, RFID hacking, and magnetic stripe-related attacks.

SnoopTrap

Purpose: SnoopTrap is a Linux-based software tool that emulates magnetic stripe readers and can be used to capture data from skimming attacks or simulate how card data is stolen via a compromised reader.
Platform: Works directly on Linux and doesn't require physical card readers.
Use Case: Simulating magnetic stripe reader attacks and understanding how skimming devices work.

SnoopTrap on Linux

git clone https://github.com/snooptrap/snooptrap.git

Install any dependencies (if required) using:
Compile the tool (if it's a compiled C-based program):
make
Once installed, you should be able to run it using:

./snooptrap

Simulate a Skimming Attack
SnoopTrap emulates a magnetic stripe reader and captures the data that would otherwise be sent to the payment processor. The tool can be used to simulate how a skimming device works.
Launch SnoopTrap: Run SnoopTrap to start listening for magnetic stripe data:
It should show that it’s listening for card reader input.
Input Data: You can feed magnetic stripe data by simulating card reader input. For example, simulate magnetic stripe data
echo "%B1234567890123456^DOE/JOHN^2403201000000000000000000000000?" | ./snooptrap

This command sends data to SnoopTrap, mimicking what would happen when a card is swiped at a skimming device.
Output: SnoopTrap should capture the data, showing the Primary Account Number (PAN), cardholder name, and expiration date on the terminal as if it were being skimmed from a real card.
The output -

Card Data Captured:
PAN: 1234567890123456
Cardholder: DOE/JOHN
Expiry: 03/2024

Analyze the Captured Data

Magnetic Stripe Data: The captured data would include the PAN, cardholder's name, and other relevant information.
Fraudulent Use: Attackers could use this data to clone cards or conduct unauthorized transactions.

Countermeasures

Chip-and-PIN (EMV): Chip technology makes it harder to skim data.
End-to-End Encryption: Ensuring that card data is encrypted during transmission can prevent interception.

img:of physical skimmer

much simpler way ...

Deep Dive into EMV and Cryptographic Attacks

EMV Smart Cards Overview:

- EMV **(Europay, Mastercard, Visa)** smart cards are used in secure payments and employ cryptographic techniques to protect transaction data.

- The key components of EMV involve both hardware and software to create secure payment systems that are difficult to replicate or tamper with.

- Cryptography plays a crucial role, utilizing algorithms like 3DES (Triple Data Encryption Standard) and RSA for encryption and securing communication between the card, the terminal, and the payment network.

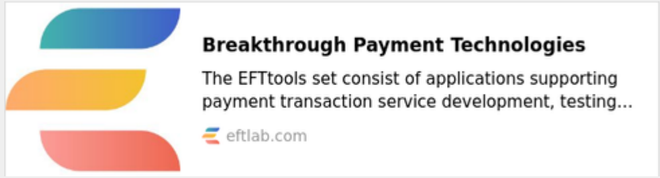
Role of Cryptography in EMV Transactions:

- **3DES**: Used for encrypting the cardholder's PIN and other transaction data.

- **RSA:** Employed in the card authentication process, particularly in generating and validating the dynamic data involved in the transaction.

- EMV cards generate **dynamic cryptograms** using **cryptographic keys,** ensuring that every transaction is unique. These cryptograms act as temporary digital signatures that authorize the payment.

The EFTLab's EMV Tool (or similar tools) provides a practical way to visualize and analyze how data flows during an EMV transaction. It allows you to examine the cryptographic elements of the transaction, giving insights into how the EMV process secures sensitive payment information.



The tool enables users to:

- Capture transaction data between the card and terminal.
- Analyze the creation and transmission of dynamic cryptograms.
- Understand how cryptographic keys (such as 3DES and RSA) are applied to secure transaction details.

Key Elements of the EMV Cryptographic EFTLab's EMV Tool

ARQC (Authorization Request Cryptogram) Generation:
The tool allows you to simulate the creation of an ARQC, which is a dynamic cryptogram generated by the card during a transaction.
This cryptogram is sent to the payment network for validation, and it contains transaction-specific data encrypted with the card's secret key.
By using the tool to generate an ARQC, users can see firsthand how cryptographic algorithms ensure each transaction is unique and secure.
ARPC (Authorization Response Cryptogram) Generation:
After the ARQC is validated, the payment network sends an ARPC back to the terminal to authorize or decline the transaction.
The EMV tool can simulate this process, showing how cryptographic keys are used to generate the response and how the terminal verifies the integrity of the transaction.
Session Key Derivation:
EMV transactions use session keys derived from the card's unique key and transaction-specific data. This ensures that each session (transaction) has its own distinct cryptographic keys, even if the same card is used for multiple transactions.
The tool helps visualize how session keys are derived, emphasizing the security provided by dynamic key generation.
Cryptogram Verification:
The tool offers a feature to verify cryptograms (such as ARQC and ARPC). Users can input the relevant data and see if the cryptogram matches what is expected.
This helps demonstrate the cryptogram validation process used by the payment network, highlighting how cryptograms prevent unauthorized transactions.
Issuer Script Processing:
The EMV Tool also simulates issuer script processing, where the issuer can send additional commands to the card (such as changing the card's settings or revoking it).
This feature is useful to explain how issuers can maintain control over the card even after it has been issued, providing an additional layer of security in the EMV system.

NFC Payment Attacks: Weaknesses in Contactless Payments

Overview of NFC Technology in Payments
NFC (**Near-Field Communication)** is widely used in contactless payment systems like Visa and MasterCard, enabling fast and convenient transactions. However, the ease of use comes with certain vulnerabilities, particularly in how payment data is transmitted wirelessly over short distances.

Weaknesses in Visa and MasterCard NFC Implementations

While NFC technology has security features, there are known weaknesses, especially in Visa and MasterCard implementations, that can expose users to risks. The primary vulnerabilities include:

- **Data Interception:** NFC communication can be intercepted by nearby attackers, allowing them to capture sensitive information such as the card's transaction data.
- **Relay Attacks:** In a relay attack, a malicious actor intercepts and relays NFC signals between a card and a payment terminal, enabling unauthorized transactions even when the cardholder is not physically present.
- **Man-in-the-Middle (MitM) Attacks:** In this scenario, an attacker positions themselves between the NFC card and terminal to manipulate or steal data without the user's knowledge.

Fact: NFC Data is Susceptible to Interception
NFC data is transmitted wirelessly within a short range (typically up to 4 cm), but sophisticated attackers can extend this range using tools to intercept or relay signals. This exposes users to potential relay and man-in-the-middle attacks, where transaction data can be manipulated or used to perform fraudulent payments.

Cardholder Verification Methods (CVM) and Bypass Techniques:

What are CVMs?

Cardholder Verification Methods (CVM) are techniques used by payment systems to verify the identity of the cardholder during a transaction. These methods ensure that the transaction is being authorized by the legitimate cardholder. Common CVMs include:

- PIN Verification: The most common method, where the cardholder enters a secret PIN to confirm their identity.
- Signature Verification: The cardholder signs a receipt or digital signature pad, and the merchant checks the signature.
- Contactless Verification: For small transactions, no additional verification might be required, but for larger amounts, a PIN or other method may be prompted.

Cardholder Verification Method (CVM) List 8E	16	000000000000000000002015E035F030403h
X	4	00000000h
Y	4	00000000h
CVM 1	2	> Fail cardholder verification if this CVM is unsuccessful: Enciphered PIN verified online - If unattended cash
CVM 2	2	> Apply succeeding CV rule if this rule is unsuccessful: Signature (paper) - If terminal supports the CVM
CVM 3	2	> Apply succeeding CV rule if this rule is unsuccessful: No CVM Required - If terminal supports the CVM
CVM 4	2	> Fail cardholder verification if this CVM is unsuccessful: Enciphered PIN verification performed by ICC - If terminal supports the CVM

CVM Bypass Attack Using APDU Commands

Step 1: Accessing the CVM List on the EMV Chip

APDU Command Overview:

- APDU (Application Protocol Data Unit) commands are used to communicate with smart cards, including EMV cards. These commands allow you to read data from the card or modify data like the CVM (Cardholder Verification Method) list.
- To access the CVM list, the attacker needs to send specific APDU commands to the EMV card, usually through an NFC-enabled reader or smart card reader.

Sending APDU Commands:

- Using tools like Proxmark3 or ACR122U NFC Reader, send a GET DATA APDU command to read the CVM list from the card.
- Example command format: 00 B2 XX YY, where the specific values of XX and YY depend on the structure of the data on the EMV chip.
- The card will respond with the CVM list, which outlines the card's verification methods (e.g., PIN required, no CVM required, signature required).

Step 2: Understanding the CVM List Format

The CVM list returned by the card contains a series of entries, each specifying a particular verification method (e.g., PIN, signature, or no verification). The entries also indicate under what conditions each method is used (e.g., for transactions above a certain amount).
A typical CVM list might look like:
0x01: Online PIN verification.
0x02: Signature verification.
0x03: No CVM required (contactless payments below a certain threshold).

Step 3: Modifying the CVM List Using APDU Commands
Sending a **WRITE APDU** Command:
Once the CVM list has been accessed, you can send a WRITE APDU command to modify the CVM list stored on the card.
Example format: **00 D6 XX YY** [New Data], where **XX** and **YY** specify the location to be written, and [New Data] contains the modified CVM list.
Modify the list by replacing the "PIN required" entry with a "No CVM required" entry. For instance:
Replace the Online **PIN (0x01) entry with No CVM (0x03).**

Example Modification:
Original CVM List: [0x01: PIN required for transactions > $100, 0x03: No CVM for contactless].
Modified CVM List: [0x03: No CVM for all transactions, 0x03: No CVM for contactless].
This modification ensures that the terminal no longer prompts for PIN verification, even for high-value transactions.

Step 4: Bypassing PIN Verification

Triggering a Transaction:
After modifying the CVM list, trigger a transaction using a payment terminal or a simulated environment (e.g., a terminal emulator or POS system).
Instead of requesting a PIN for a high-value transaction (e.g., above $100), the terminal will proceed without requesting verification due to the modified CVM list.

Observing the Impact:
The terminal no longer prompts for PIN input, even though the transaction exceeds the normal verification threshold.
The attacker can authorize high-value transactions without entering a PIN or providing any additional cardholder verification.

Step 5: Discussing the Real-World Implications

High-Value Fraud:
Attackers can exploit this CVM bypass technique to authorize fraudulent transactions, particularly in environments with weak or outdated terminal security.

ATM Attacks:
The same principles apply to ATMs, where bypassing PIN verification can allow unauthorized cash withdrawals without the cardholder's PIN.

Defensive Techniques Against CVM Bypass

Encrypted PIN Verification:

- Use encryption methods to secure PIN input, where the PIN is encrypted directly by the card and verified by the bank's backend systems (e.g., using an HSM), ensuring attackers cannot bypass the PIN at the terminal level.

Secure CVM Validation:

- Implement EMV protocols that ensure the integrity of the CVM list, using cryptographic verification to detect and reject any attempts to tamper with the CVM list.

Regular Software Updates:

- Ensure payment terminals are updated with the latest security patches to prevent vulnerabilities like the one exploited in the PIN OK Attack.

OTP and Authentication Bypass Attacks:

What is OTP?
An OTP (One-Time Password) is a security feature used in many systems (especially mobile banking, e-commerce, and two-factor authentication) to provide an additional layer of protection. It generates a temporary, single-use code that is typically sent to the user via SMS, email, or an authenticator app. While OTP systems are widely used, they are not without vulnerabilities. Attackers have developed various techniques to bypass these protections.

Common Weaknesses in OTP Systems
Brute Force Attacks:
OTP systems, especially those using short codes (e.g., 4-6 digits), can be vulnerable to brute-force attacks, where attackers attempt all possible combinations in rapid succession until the correct OTP is found.

Race Conditions:
A race condition occurs when multiple processes attempt to access and modify shared data simultaneously. In the context of OTP systems, attackers can exploit this flaw by sending multiple OTP verification requests simultaneously, sometimes allowing them to bypass OTP checks due to timing inconsistencies.

Insufficient Rate Limiting:
Without proper rate limiting (restrictions on how many attempts can be made within a given time), attackers can brute-force OTP codes more easily, sending numerous requests in quick succession without facing consequences, such as account lockouts.

Real-World Example: OTP Bypass in Mobile Banking Fraud
Attackers have targeted OTP systems on mobile banking platforms by exploiting the weaknesses mentioned above. In some cases, they have used SIM-swapping techniques to intercept OTPs sent via SMS, or they have brute-forced weak OTP codes due to the absence of rate-limiting or multi-factor authentication.
These attacks have led to significant financial fraud, where attackers have transferred funds without being detected until it’s too late.

Countermeasures Against OTP Bypass Attacks

Multi-Factor Authentication (MFA):
Instead of relying solely on OTP, combine it with other authentication factors (e.g., biometric verification, security questions) to make bypassing the system more challenging for attackers.

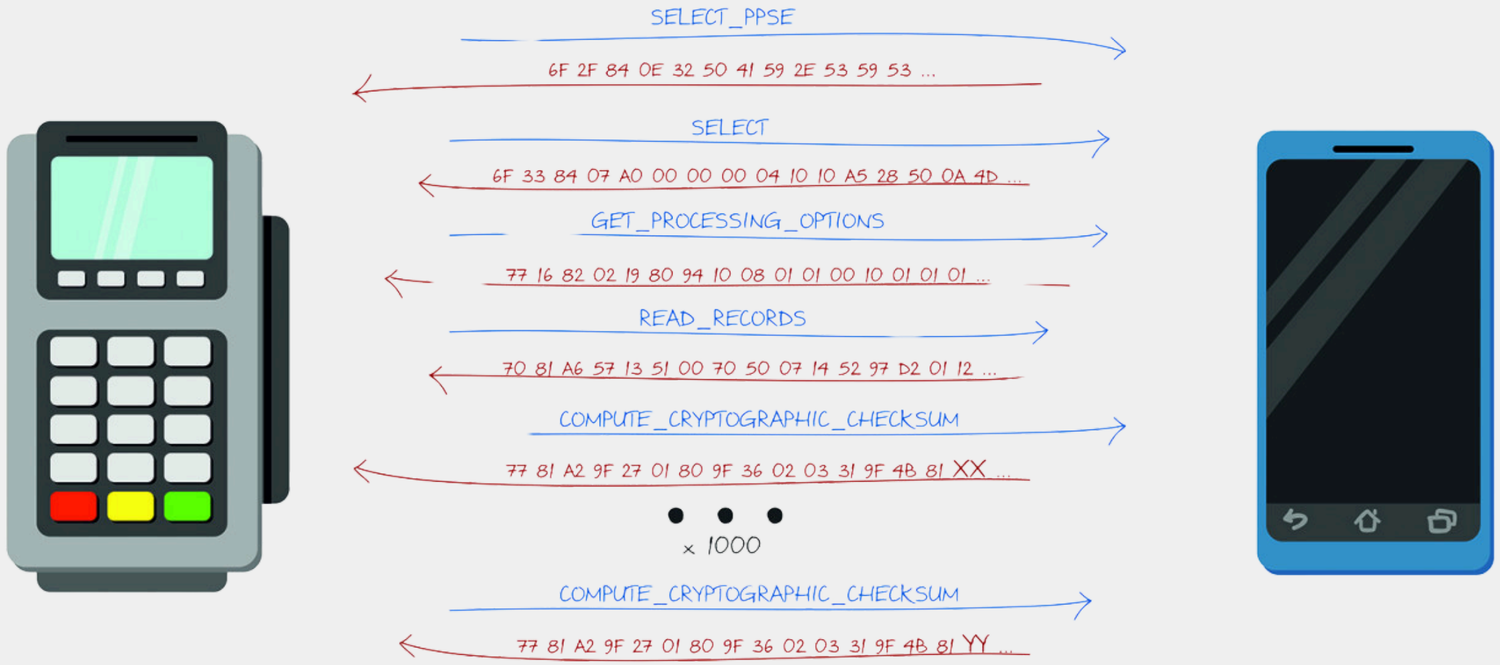
Longer and More Complex OTPs:
Increasing the length of OTPs (e.g., 8 digits or alphanumeric codes) can significantly increase the complexity of brute-force attempts, making it harder for attackers to guess the code within a reasonable time frame.

Rate Limiting and Account Lockouts:
Implement strict rate limiting, where users can only attempt OTP submissions a certain number of times (e.g., 3-5 attempts) before the account is temporarily locked or the OTP is invalidated.
Additionally, using CAPTCHA during OTP entry can help prevent automated brute-force attacks.

IP Blacklisting and Geolocation Restrictions:
Monitor for suspicious IP addresses or locations attempting OTP brute-force attacks, and blacklist those IPs after detecting multiple failed attempts.
Use geolocation restrictions to block login attempts from regions or countries not associated with the account.

So the attack plan looks like this:

- We read the card and find out the number of significant digits in UN that the terminal will provide
- We iterate over all UNS, get all possible values of the COMPUTE_CRYPTOGRAPHIC_CHECKSUM function, and store them in the corresponding table with the mapping UN - > Result
- We bring it to the POS-terminal, find out the number that the POS-terminal asks for.
- We select the desired result from the table and substitute it in the response to the terminal.
- The transaction goes away.
- PROFIT. However, the success of the transaction approval is not guaranteed, since the issuing bank may reject such a transaction.

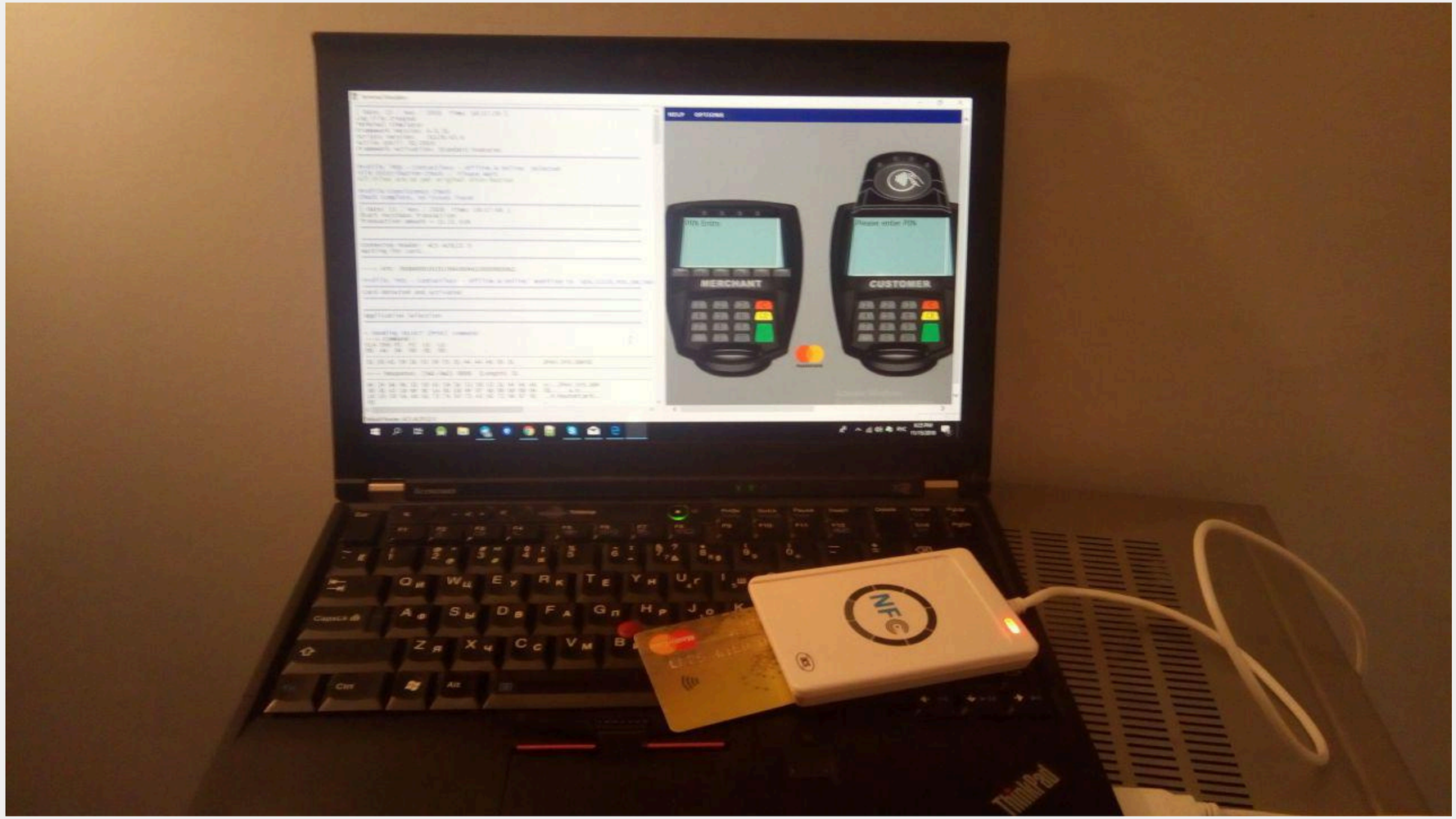


Terminal Simulator

```
+ Sending SELECT command
---> Command :
CLA INS P1  P2  LC  LE
00  A4  04  00  07  00
--------------------------------------------------------------------------------
A0 00 00 00 04 10 10                                .......
--------------------------------------------------------------------------------
<--- Response: (SW1-SW2) 9000  (Length) 35
--------------------------------------------------------------------------------
6F 33 84 07 A0 00 00 00 04 10 10 A5 28 50 0A 4D  o3.........(P.M
61 73 74 65 72 43 61 72 64 5F 2D 04 72 75 65 6E  asterCard_-.ruen
87 01 01 BF 0C 0F 9F 4D 02 0B 0A 9F 6E 07 06 43  .......M....n..C
00 00 30 30 00                                   ..00.
--------------------------------------------------------------------------------
Transaction time = 52 ms

--------------------------------------------------------------------------------
Tag   : 6F [File Control Information Template]
Length: 33
| Tag   : 84 [DF Name]
| Length: 07
| Data  : A0000000041010
| Tag   : A5 [File Control Information Proprietary Template]
| Length: 28
| | Tag   : 50 [Application Label]
| | Length: 0A
| | Data  : 4D617374657243617264
| | Tag   : 5F2D [Language Preference]
| | Length: 04
| | Data  : 7275656E
| | Tag   : 87 [Application Priority Indicator]
| | Length: 01
| | Data  : 01
| | Tag   : BF0C [File Control Information Issuer Discretionary Data]
| | Length: 0F
| | | Tag   : 9F4D [Log Entry]
| | | Length: 02
| | | Data  : 0B0A
| | | Tag   : 9F6E [Third Party Data]
| | | Length: 07
| | | Data  : 06430000303000
--------------------------------------------------------------------------------
+ SELECT processed ok

================================================================================
Initiate Application Processing
================================================================================

+ No PDOL supplied, using command data 8300
--------------------------------------------------------------------------------
+ Sending GET PROCESSING OPTIONS command
```

Default Reader: ACS ACR122 0

An ACR122 NFC reader can be used to read the card.



Q & A

Any queris

EAT
SLEEP
HACK
Everday

NO device is 100%
secure.