Motivation and Research Goal

Motivating Scenario
- Near-Field Communication (NFC) is widely used for security-critical applications like payment or access control systems.
- Assumption: Low range of wireless communication ensures physical proximity of card and reader.
- Some systems not protected against wormhole attacks.
- Existing wormholing solutions expensive, impractical, or complicated to set up.

Research Goal and Approach
- Leverage NFC capabilities of Android phones to build an inexpensive wormholing setup for future research.
- Avoid flashing the device with custom ROM.
- Gain access to transmitted data for other attacks.

Challenges
- HCE only supports APDU commands based on ISO 7816-4 and only forwards sessions starting with a SELECT command. This excludes popular protocols like (native) Mifare DESFire.
- HCE uses a random Unique Identifier (UID), which may be detected and blocked by readers performing UID verification and cannot be changed by the application.
- Android HCE implementation has no support for setting fixed UID.

Circumventing AID Verification
- AID detection and verification is performed by two parts of the system: libnfc (native C) and android.nfc.cardemulation (Android OS, Java).
- Modify Java part at runtime using XPosed [3] to always return a special AID, regardless of the received command.
- Register the special wildcard AID for our application.
- From XPosed, load a library into the NFC daemon and live-patch the binary ARM code to re-route function calls to our own code.
- Android will now route any received NFC command to our application, regardless of the used protocol or AID.
- This also ensures compatibility with other protocols like DESFire.

UID Emulation
- Analysis of libnfc-nci source code (libnfc for Broadcom chips) shows function for passing arbitrary config strings to NFC chip firmware.
- Further analysis reveals command bytes for setting UID of chip.
- At runtime, read UID from card, transmit to the other device, and set the UID of that device using an IPC command to the NFC daemon.
- Code inside the NFC daemon uploads the newly built config to the chip.
- Chip will now use the emulated UID instead of a random value.
- Emulation of ATQA, SAK and Historical Byte works the same way.

Limitations
- UID Emulation only possible for certain Broadcom Chips (used in Nexus 4 and 5, among other devices).
- Assorted bugs in Chip Firmware and Android.

Results and Discussion

Results
- Successful test of wormholing capabilities against a real-world contactless payment and access control system.
- NFC commands logged on the device for later inspection.
- Modification of NFC commands in transit possible using a Python client.
- Dedicated UID clone mode to bypass UID-based systems.

Discussion
- Relay introduces noticeable delay of 65 ± 38 ms when using a local WiFi, higher delays when routing over the public internet.
- Relay can be detected if the system checks the timing of commands or uses other methods for distance bounding.
- Delay could be reduced by using other technologies like Bluetooth.
- Modification of NFC traffic directly on the Android device would be more efficient. Infrastructure in place, but GUI currently missing.

For more information, visit seemoo.de/nfcgate

NFCGate is licensed under the Apache License v2.

Future Work

Further improvements
- Rule-based, on-the-fly modification of NFC traffic on the Android devices.
- Compatibility with other NFC Chips.
- More communication channels (Bluetooth, WiFi Direct, …).