QuickShell

Sharing is caring about an RCE attack chain on Quick Share

Or Yair

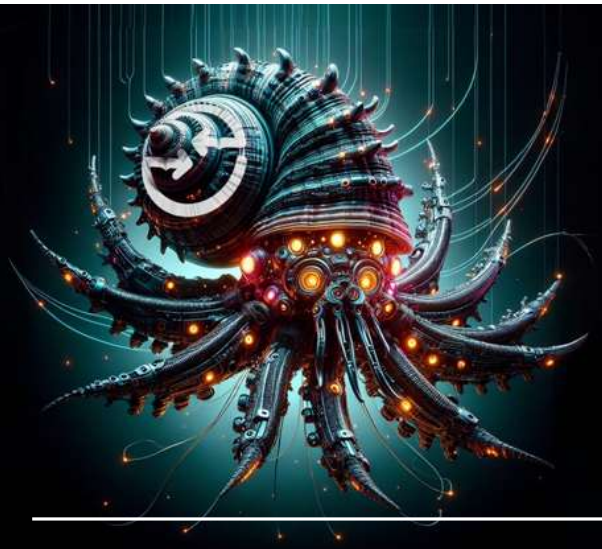
- Security Research Team Lead at SafeBreach
- 7+ years in Security Research
- Past research in Linux, embedded, Android
- 4 years Windows research

Shmuel Cohen - Contributer

- 6+ years in Security Industry
- Past APT Malware Researcher
- 4+ years Windows research

Agenda

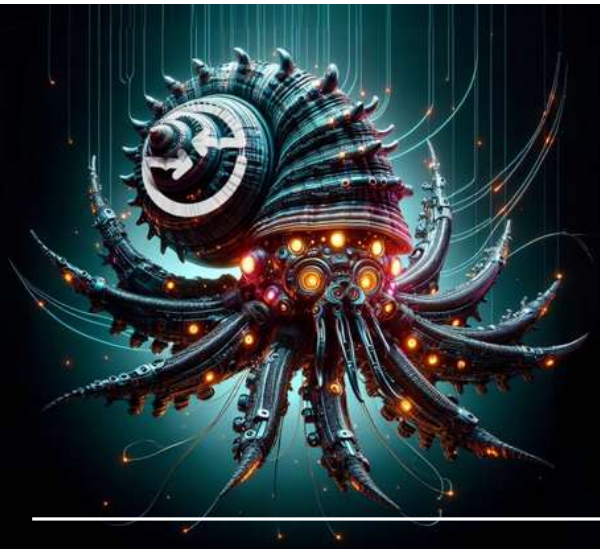
- Why Quick Share
- Protocol Overview
- Fuzzing
- Research Approach Shift + Vulnerability Discovery
- RCE Chain
- Takeaways
- GitHub + Q&A



What is Quick Share?

Quick Share


2:42
Quick Share
You'll share as
Pixel 8 Pro
Send to your devices
Pixel 8
Sending...
Send to nearby devices
Looking for nearby devices...
2:42
Tue, Jan 23
Receiving...
From Pixel 8 Pro
1 image
Cancel



Why Quick Share?

Quick Share Windows Version

Quick Share for Windows

Wireless sharing with your PC, made easy.

Send and receive photos, documents, and more between nearby Android devices[1] and Windows PCs[2].

To get started, install Quick Share for Windows to your PC. Send yourself the link to download it.
By downloading Quick Share for Windows, you agree to the **Google Terms of Service**. The **Google Privacy Policy** describes how Google handles information from Quick Share for Windows.

Download Quick Share

Quick Share Pre-installation

Google:

ANDROID

What we announced at CES 2024

Jan 09, 2024 · 6 min read

“we’re working with leading PC manufacturers like LG to expand Quick Share to Windows PCs as a pre-installed app.”

Quick Share Communication Methods

Various communication methods

1st time by Google on Windows

WebRTC

DIRECT

Wi Fi HOTSPOT

Wi Fi

Wi Fi

Previous Research

2019 by Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen:

“Nearby Threats: Reversing, Analyzing, and Attacking Google’s ‘Nearby Connections’ on Android”

- About Nearby Connections API
- Only Android
- No CVEs

https://francozappa.github.io/publication/rearby/paper.pdf

Nearby & Chromium Open-Source Repos

Contain part of the code for Quick Share for Windows

google / **nearby** Public

🐿️ A collection of projects focused on connectivity that enable building cross-device experiences.

developers.google.com/nearby

Apache-2.0 license

696 stars 151 forks Branches Tags

chromium / **chromium** Public

The official GitHub mirror of the Chromium source

chromium.googlesource.com/chromium/src/

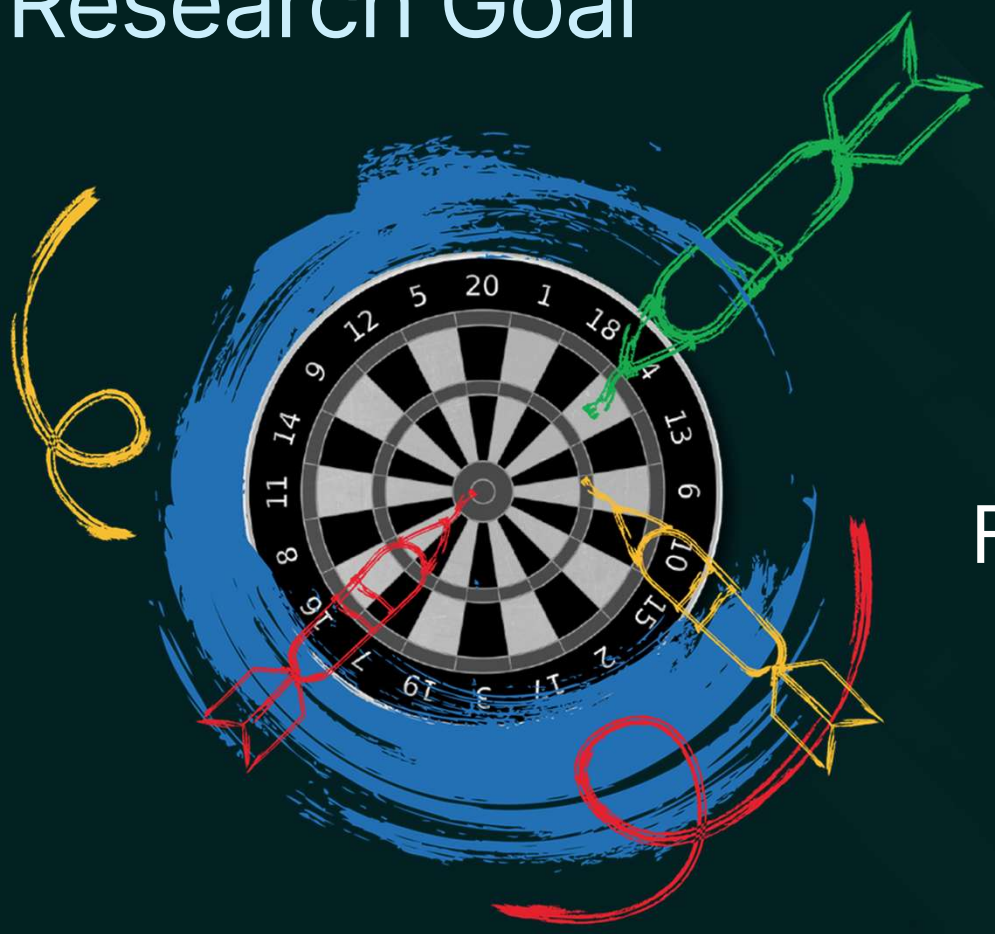
BSD-3-Clause, BSD-3-Clause licenses found

18.3k stars 6.8k forks Branches Tags

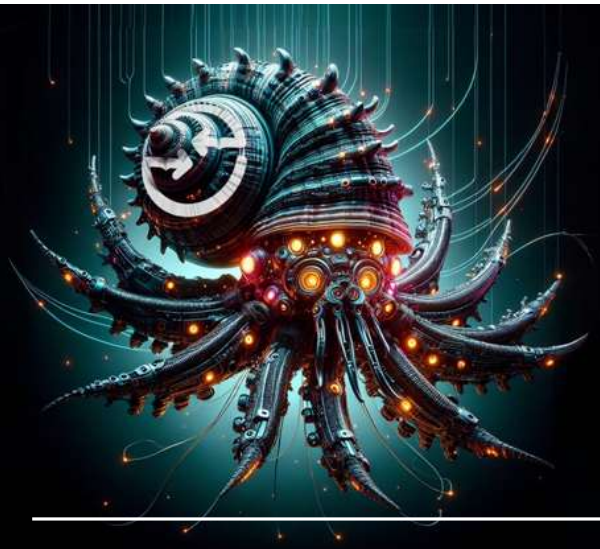
Why Quick Share

- New Windows App — New App New vulns
- Windows app will be pre-installed
- Various communication methods — Various attack vectors
- Google’s first Windows app to use these APIs
- Some of the code is open-source
- No CVEs

Research Goal



First RCE in Quick Share



Protocol Investigation

Investigating The “nearby” repo

Finding the communication functions – Send & Recv:

```
ExceptionOr<ByteArray> BaseEndpointChannel::Read(PacketMetaData& packet_meta_data)
```

```
Exception BaseEndpointChannel::Write(const ByteArray& data,
                                     PacketMetaData& packet_meta_data)
```

Protobuf and Offline Frames

```
OfflineFrame frame;
frame.ParseFromString(std::string(bytes))
```

Protobuf and Offline Frames

offline_wire_formats.proto

```
message OfflineFrame {
  enum Version {
    UNKNOWN_VERSION = 0;
    V1 = 1;
  }
  optional Version version = 1;

  // Right now there's only 1 version, but if there are more, exactly one of
  // the following fields will be set.
  optional V1Frame v1 = 2;
}
```

QuickSniff – 1st Tool

Hooking Quick Share to sniff sent and received Offline Frames on Windows


```yaml
initiator_to_responder:
v1:
  payloadTransfer:
    packetType: DATA
    payloadChunk:
      body:
        v1:
          introduction:
            fileMetadata:
            - id: '585290039179534374'
              mimeType: image/png
              name: TFMyMDI0MDYxMzEzMzM1Ni5wbmc=
              payloadId: '-8969229381597391197'
              size: '622679'
              type: IMAGE
          type: INTRODUCTION
        version: V1
      flags: 0
      index: 0
      offset: '0'
    payloadHeader:
      id: '-5778571142958742193'
      isSensitive: false
      totalSize: '85'
      type: BYTES
  type: PAYLOAD_TRANSFER
version: V1
```

Protocol Overview

Nearby Connections API

Nearby Connections API

+

Quick Share Implementation

=

Nearby Connections API

Discover and establish direct communication channels with other devices without having to be connected to the Internet. Enables seamless nearby interactions such as multiplayer gaming, realtime collaboration, forming a group, broadcasting a resource, or sharing content.

The Nearby Connections API is available for Android and iOS, and enables communication between the two platforms.

Nearby Connections API

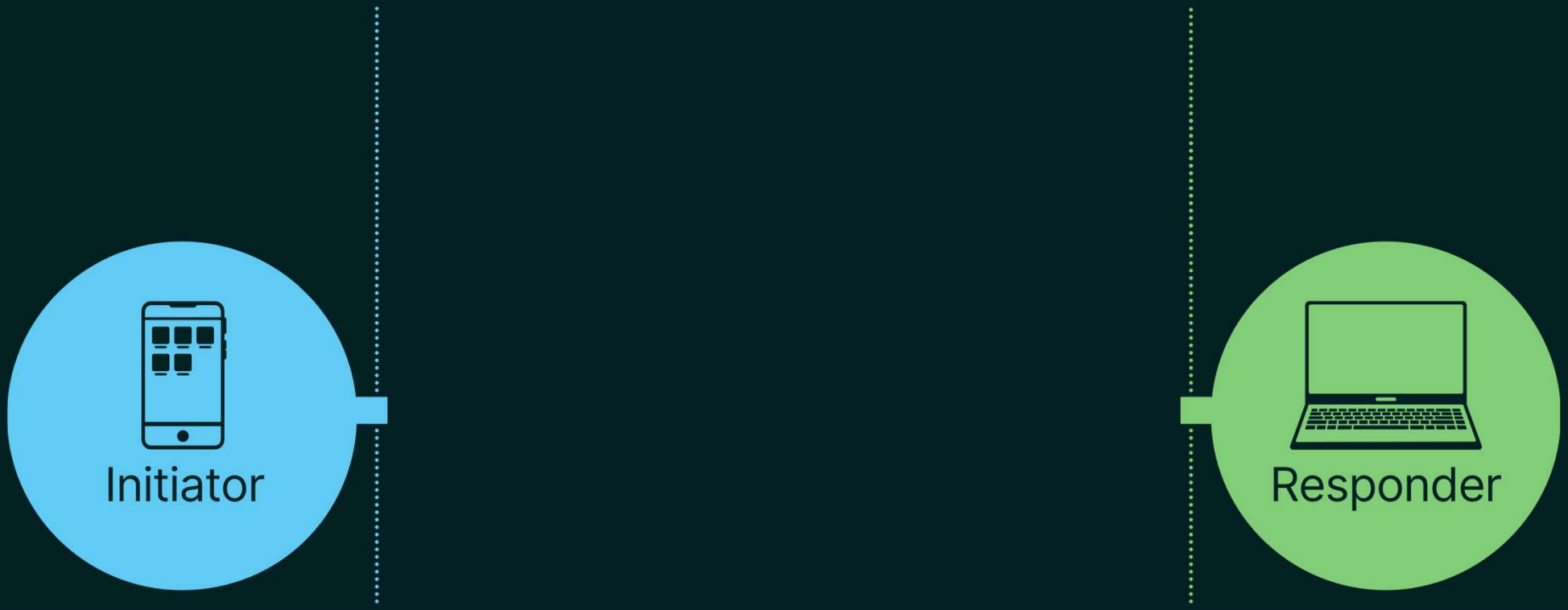
Protobuf Based

Encryption - Google's Ukey2

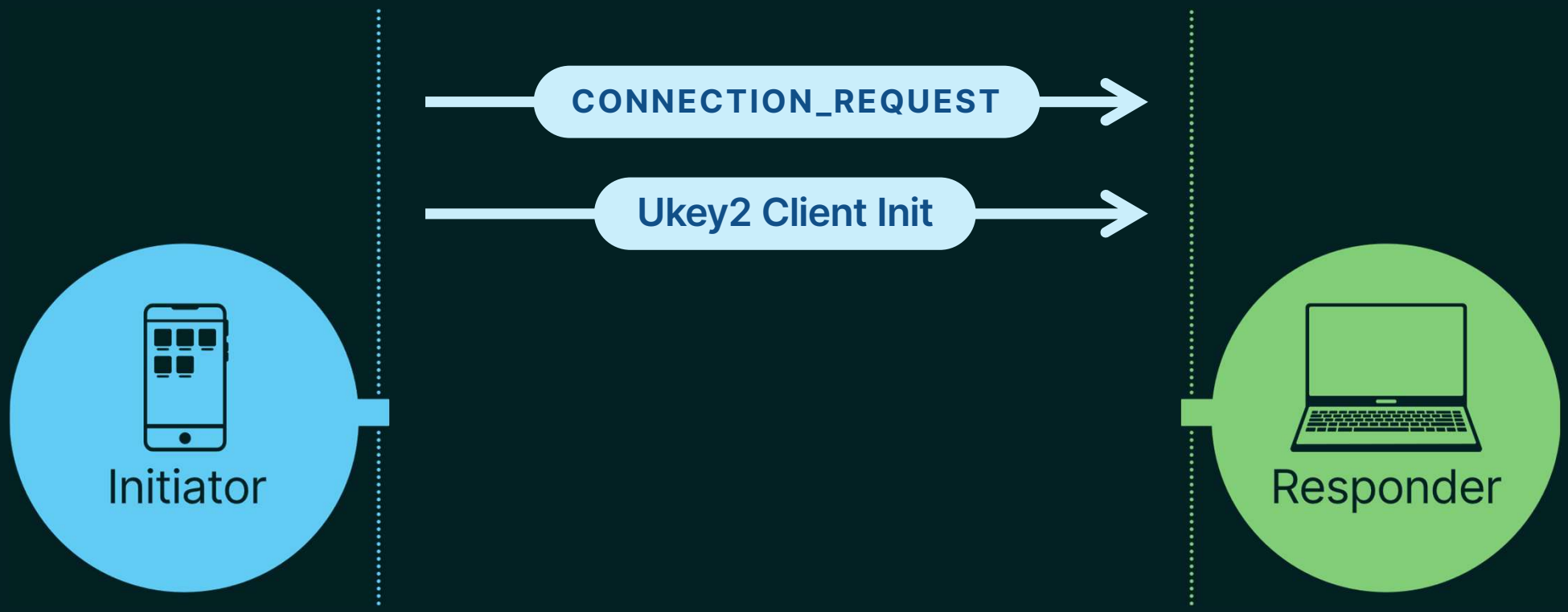
Advertisement based on Service ID

Multiple Connections Strategies
- P2P, Star, Cluster

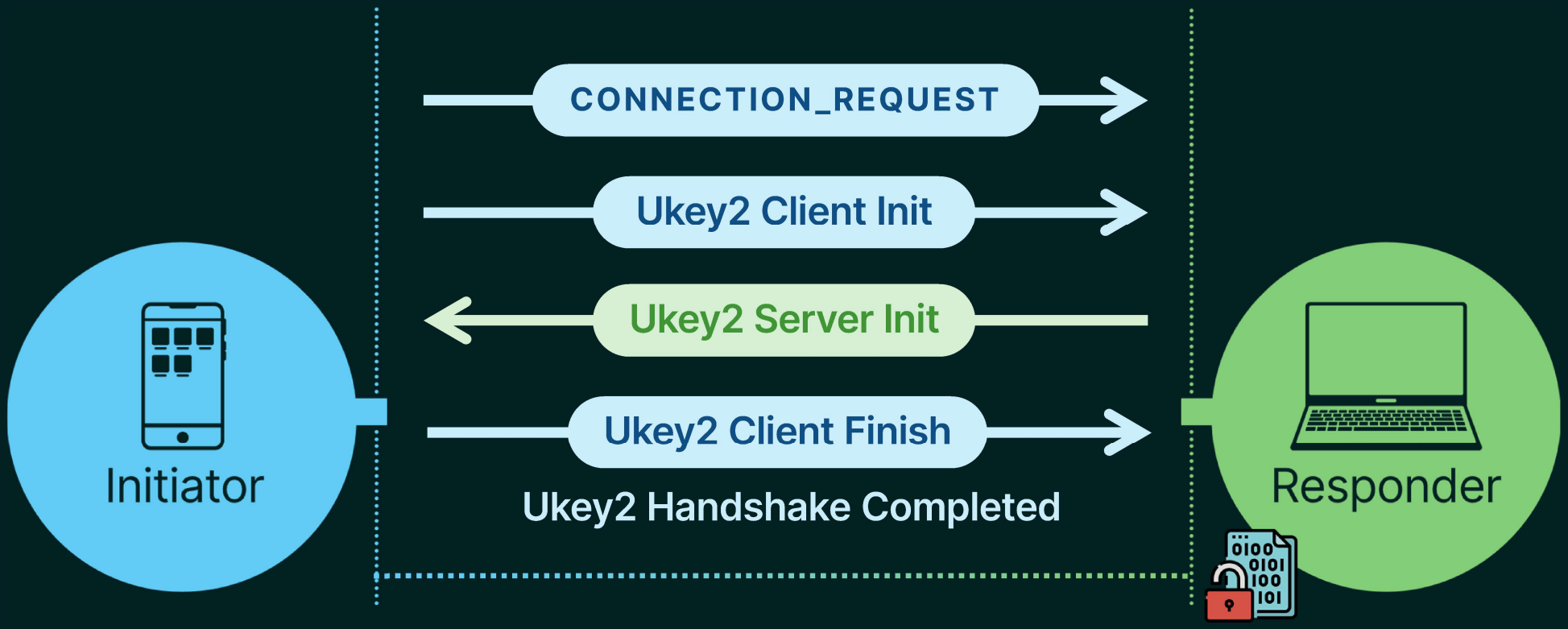
Quick Share Protocol Overview



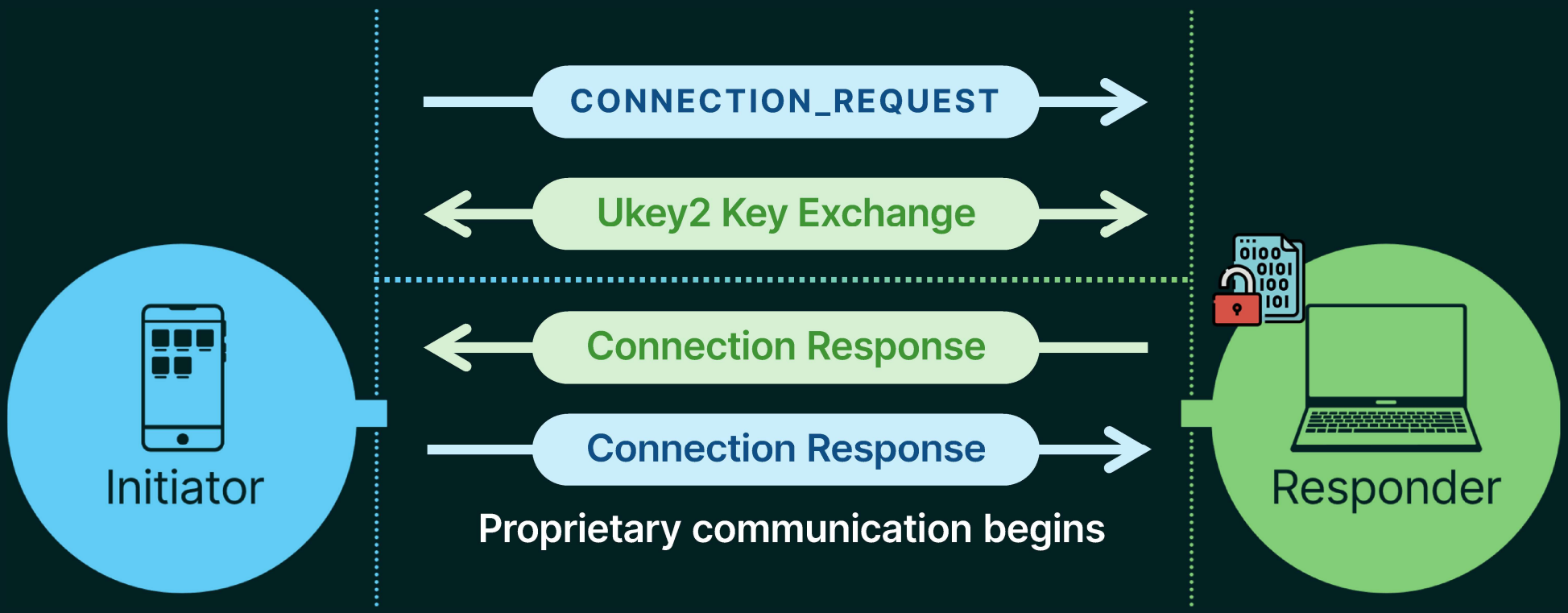
Quick Share Protocol Overview



Quick Share Protocol Overview



Quick Share Protocol Overview



Packet Types

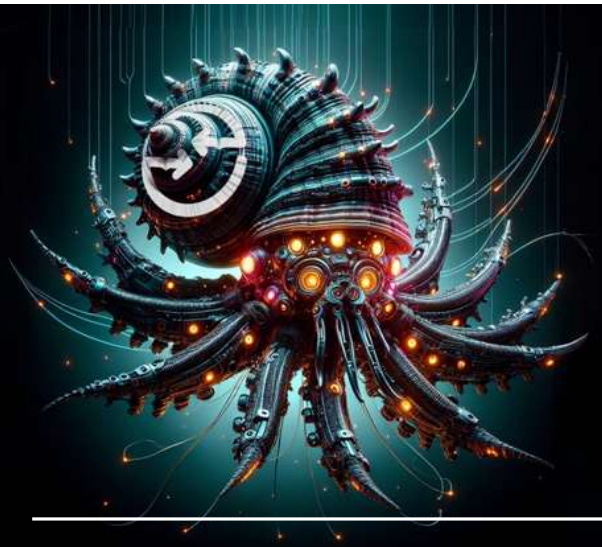
```
enum FrameType {
  UNKNOWN_FRAME_TYPE = 0;
  CONNECTION_REQUEST = 1;
  CONNECTION_RESPONSE = 2;
  PAYLOAD_TRANSFER = 3;
  BANDWIDTH_UPGRADE_NEGOTIATION = 4;
  KEEP_ALIVE = 5;
  DISCONNECTION = 6;
  PAIRED_KEY_ENCRYPTION = 7;
  AUTHENTICATION_MESSAGE = 8;
  AUTHENTICATION_RESULT = 9;
  AUTO_RESUME = 10;
  AUTO_RECONNECT = 11;
  BANDWIDTH_UPGRADE_RETRY = 12;
}
```

Packet Types

```
enum FrameType {
  UNKNOWN_FRAME_TYPE = 0;
  CONNECTION_REQUEST = 1;
  CONNECTION_RESPONSE = 2;
  PAYLOAD_TRANSFER = 3;
  BANDWIDTH_UPGRADE_NEGOTIATION = 4;
  KEEP_ALIVE = 5;
  DISCONNECTION = 6;
  PAIRED_KEY_ENCRYPTION = 7;
  AUTHENTICATION_MESSAGE = 8;
  AUTHENTICATION_RESULT = 9;
  AUTO_RESUME = 10;
  AUTO_RECONNECT = 11;
  BANDWIDTH_UPGRADE_RETRY = 12;
}
```

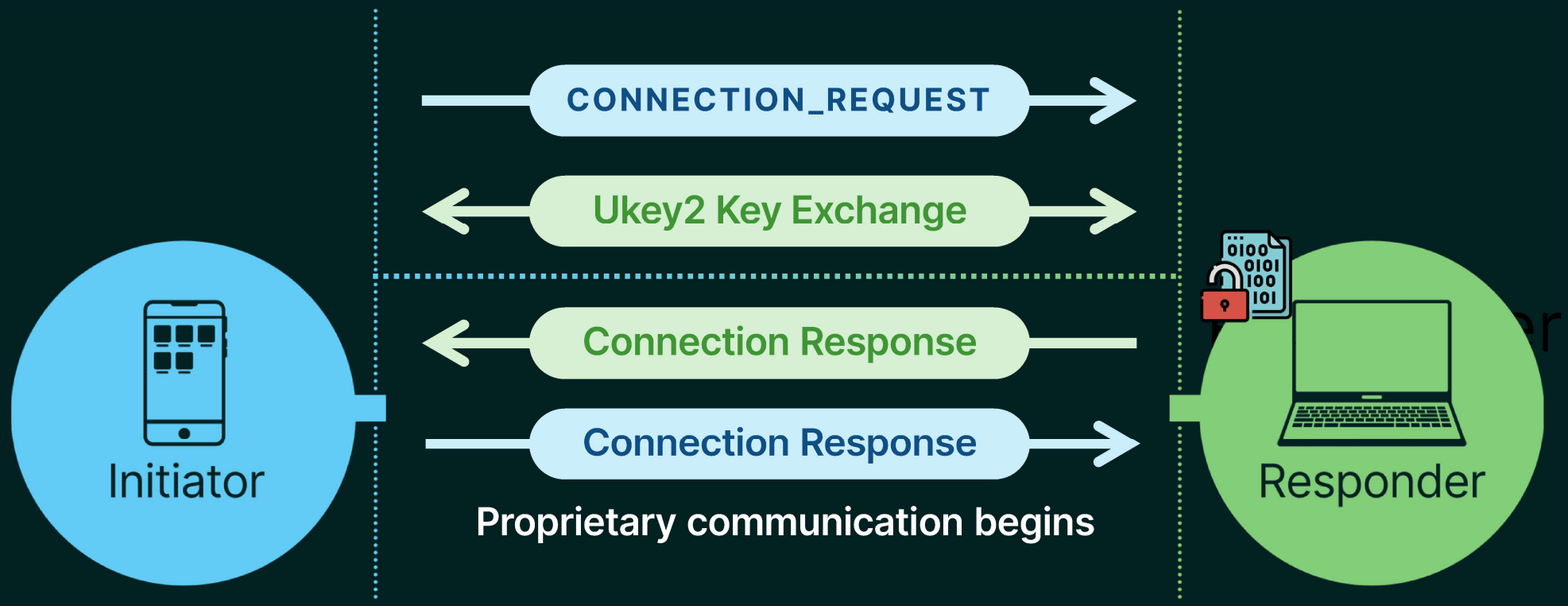
Packet Types

```
enum FrameType {
  UNKNOWN_FRAME_TYPE = 0;
  CONNECTION_REQUEST = 1;
  CONNECTION_RESPONSE = 2;
  PAYLOAD_TRANSFER = 3;
  BANDWIDTH_UPGRADE_NEGOTIATION = 4;
  KEEP_ALIVE = 5;
  DISCONNECTION = 6;
  PAIRED_KEY_ENCRYPTION = 7;
  AUTHENTICATION_MESSAGE = 8;
  AUTHENTICATION_RESULT = 9;
  AUTO_RESUME = 10;
  AUTO_RECONNECT = 11;
  BANDWIDTH_UPGRADE_RETRY = 12;
}
```



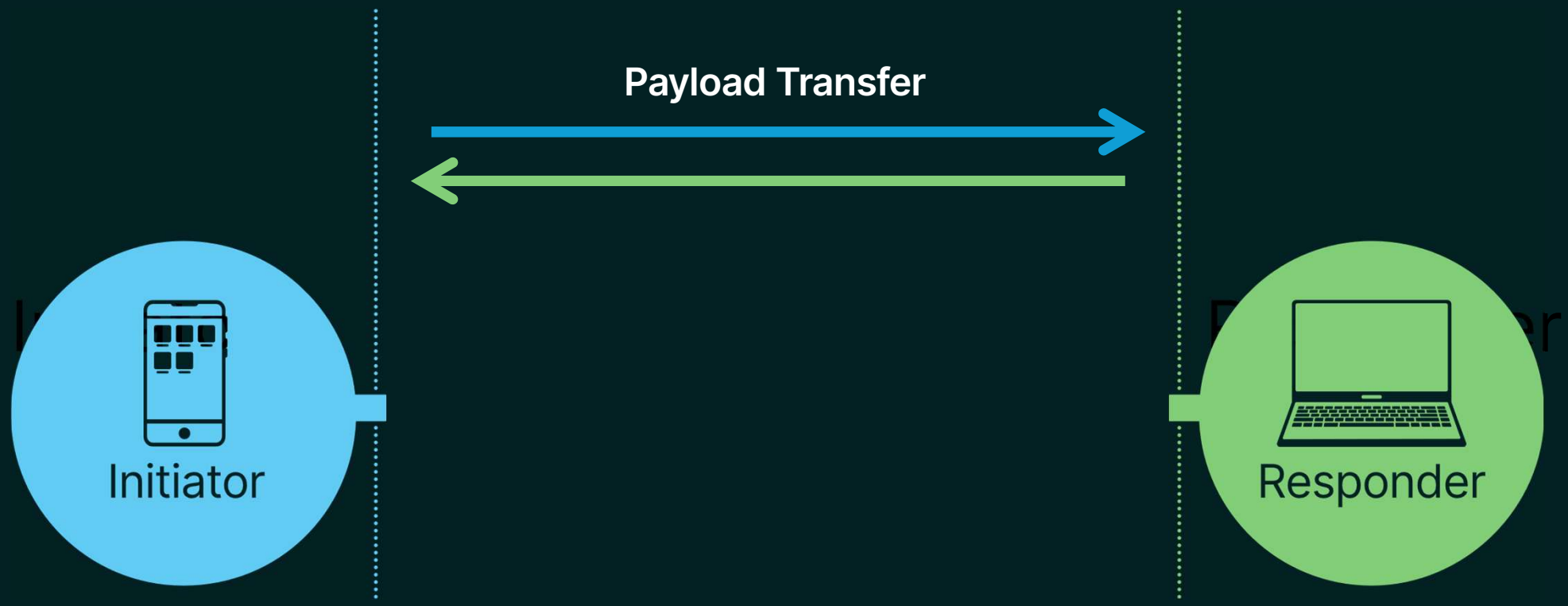
Quick Share Implementation

Recap


CONNECTION_REQUEST
Ukey2 Key Exchange
Connection Response
Connection Response
Proprietary communication begins
Initiator
Responder

Payload Transfer

Enforces “Contacts” and “Your Devices” modes



Quick Share Implementation

Custom protobuf data in Payload Transfer Payload


```yaml
v1:
  payloadTransfer:
    packetType: DATA
    payloadChunk:
      body: CAESYggBEl4KWgoIYmFzZS5hcGs ???
      flags: 0
      index: 0
      offset: '0'
    payloadHeader:
      id: '-5454771653010976901'
      isSensitive: false
      totalSize: '102'
      type: BYTES
  type: PAYLOAD_TRANSFER
version: V1
```

Quick Share Implementation

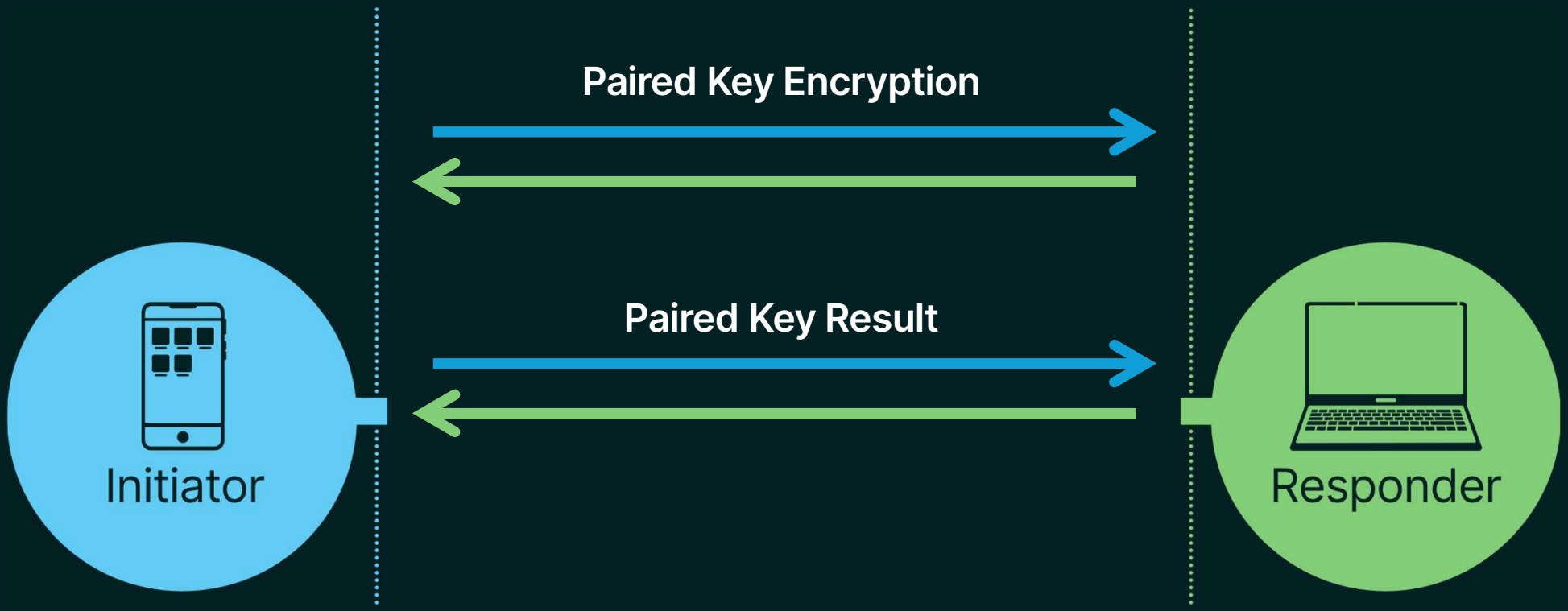
Custom protobuf data in
Payload Transfer OfflineFrame


```yaml
v1:
  payloadTransfer:
    packetType: DATA
    payloadChunk:
      body: CAESYggBEl4KWgoIYmFzZS5hcGs
      flags: 0
      index: 0
      offset: '0'
    payloadHeader:
      id: '-5454771653010976901'
      isSensitive: false
      totalSize: '102'
      type: BYTES
  type: PAYLOAD_TRANSFER
version: V1
```

```yaml
v1:
  payloadTransfer:
    packetType: DATA
    payloadChunk:
      body:
        v1:
          pairedKeyEncryption:
            secretIdHash: palHIL+r
            signedData: MEUCIQC24J1oLakk1ypg7
          type: PAIRED_KEY_ENCRYPTION
        version: V1
      flags: 0
      index: 0
      offset: '0'
    payloadHeader:
      id: '-5454771653010976901'
      isSensitive: false
      totalSize: '102'
      type: BYTES
  type: PAYLOAD_TRANSFER
version: V1
```

Payload Transfer

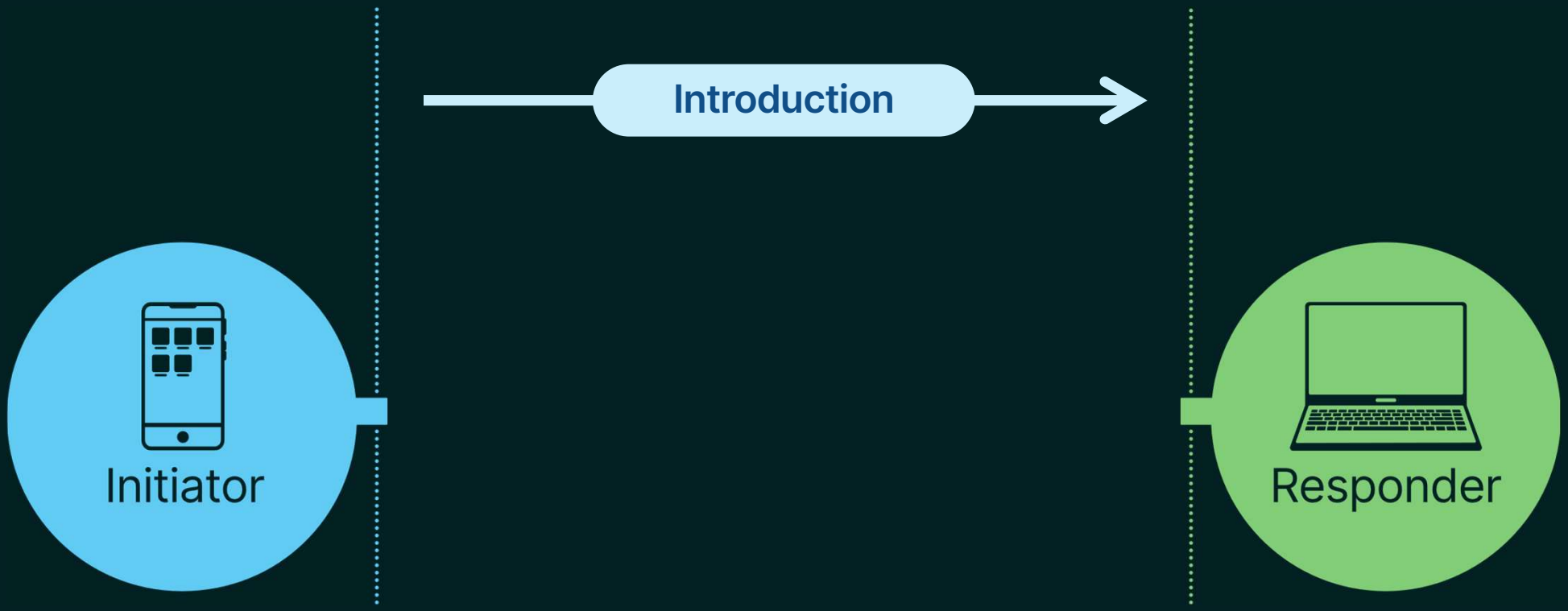
Enforces “Contacts” and “Your Devices” modes



Payload Transfer

INTRODUCTION & ACCEPT

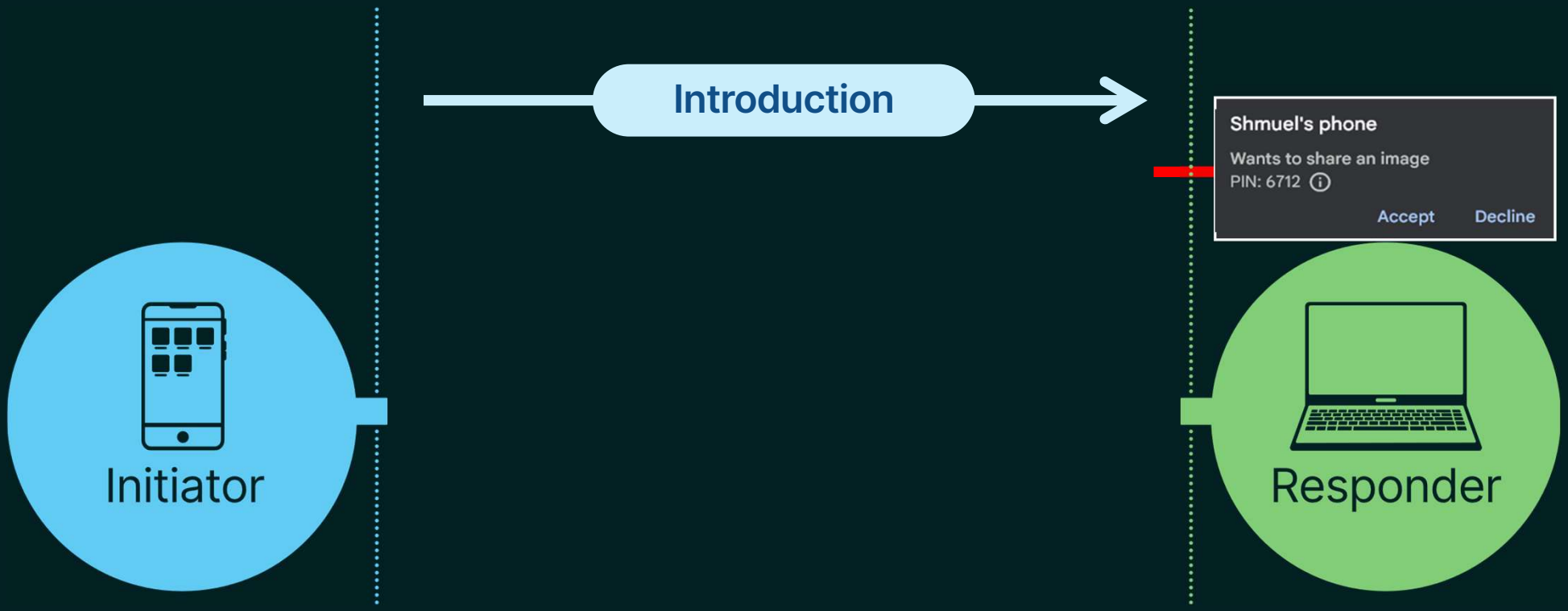
After paired Key Encryption:



Payload Transfer

INTRODUCTION & ACCEPT

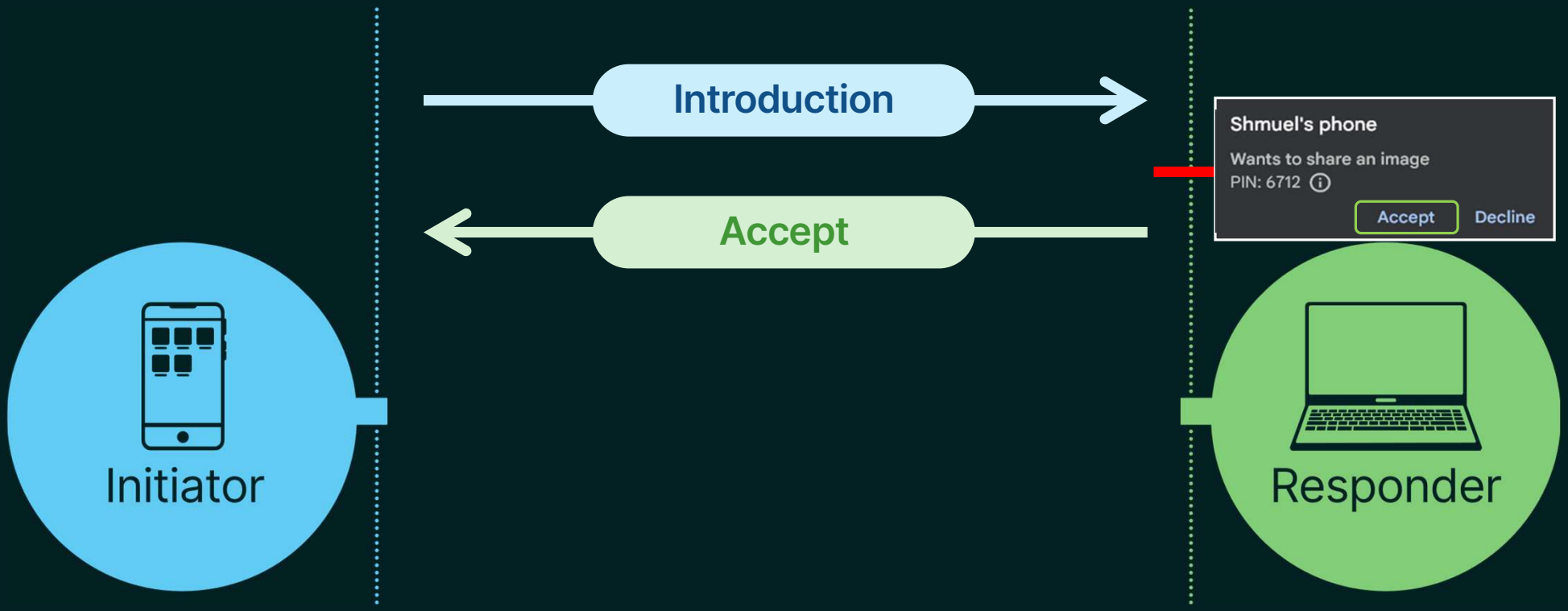
After paired Key Encryption:



Payload Transfer

INTRODUCTION & ACCEPT

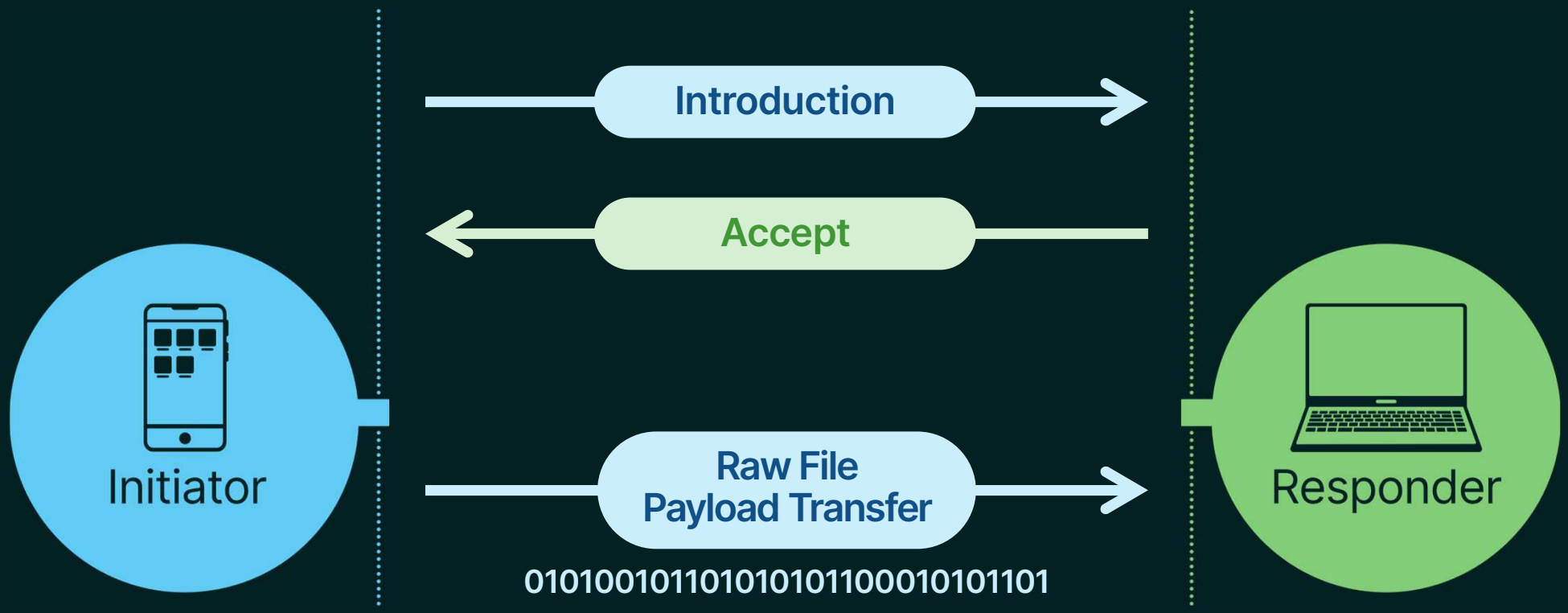
After paired Key Encryption:



Payload Transfer

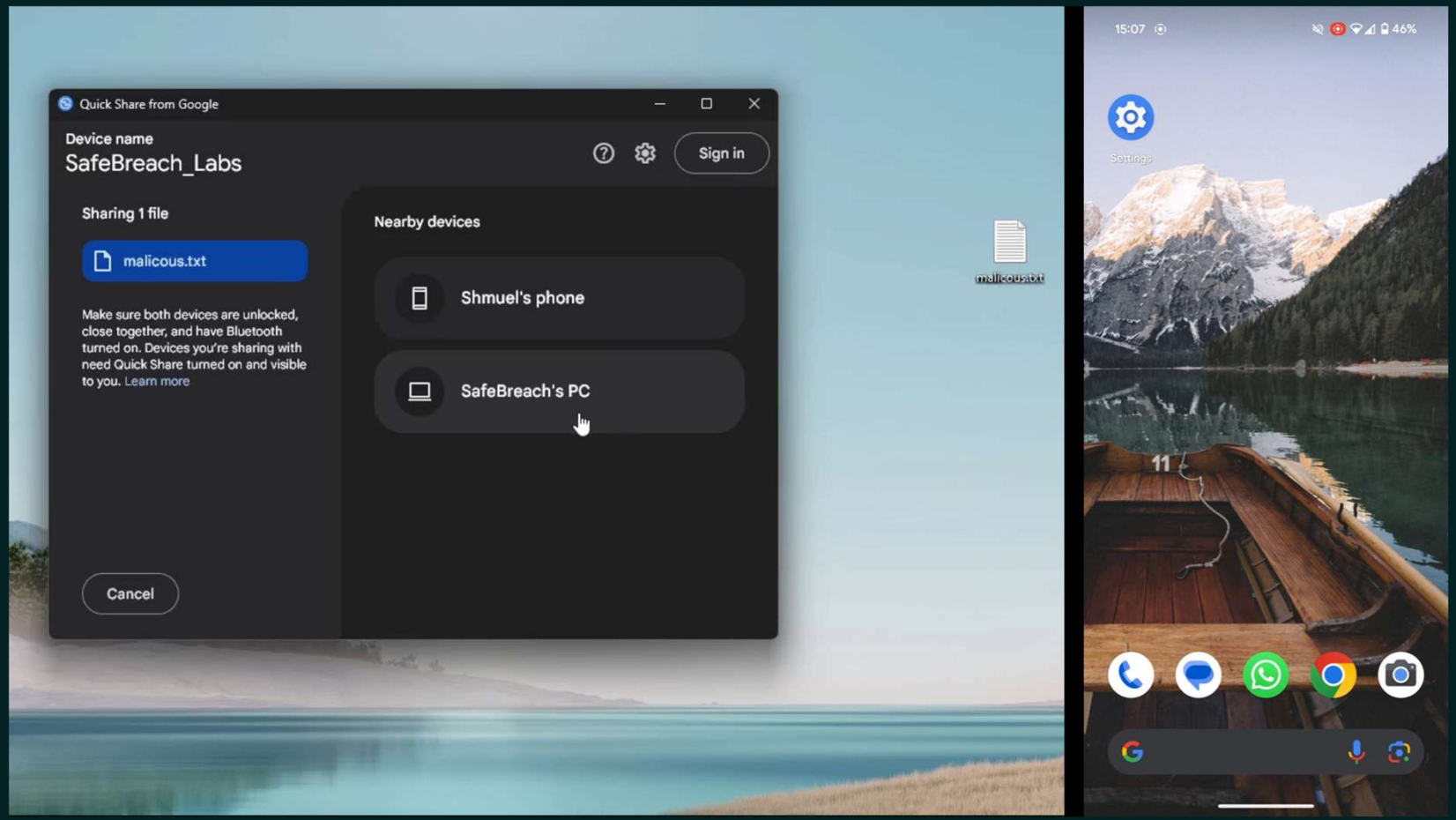
INTRODUCTION & ACCEPT

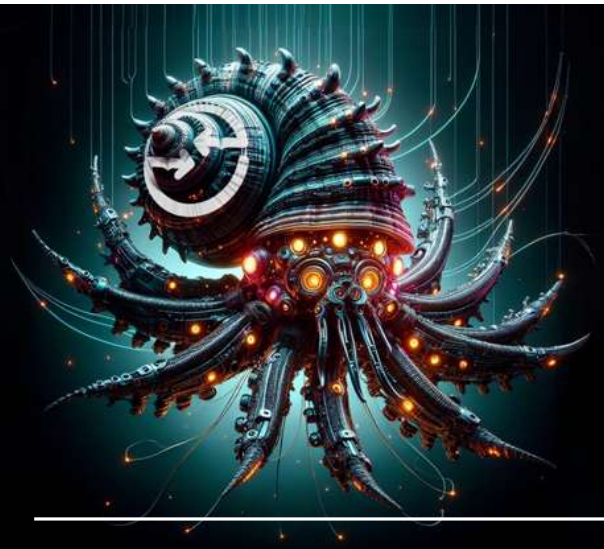
After paired Key Encryption:



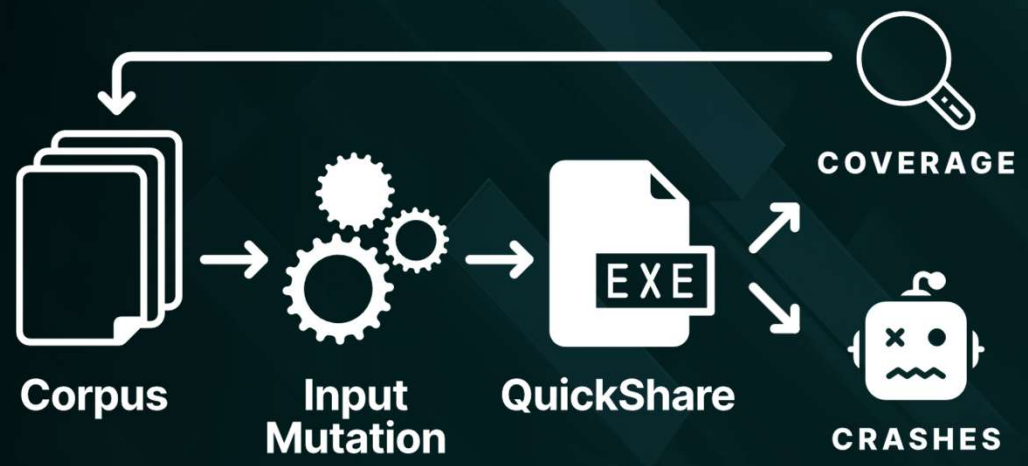
Payload Transfer

Introduction & Accept

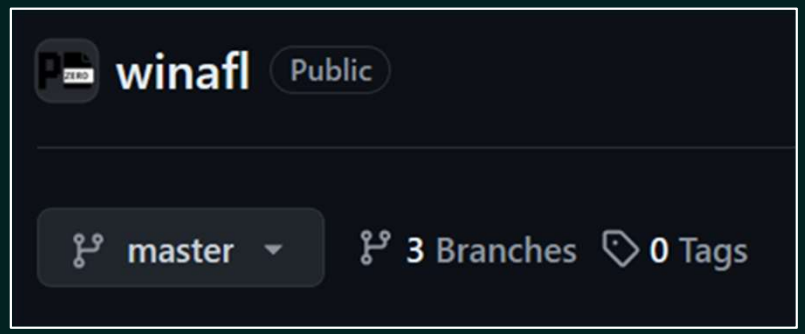




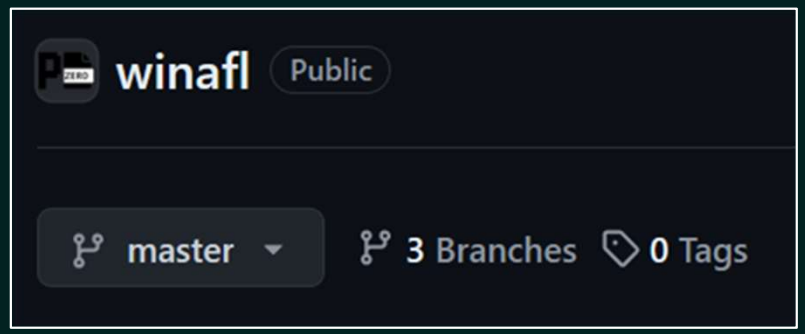
Fuzzing Quick Share



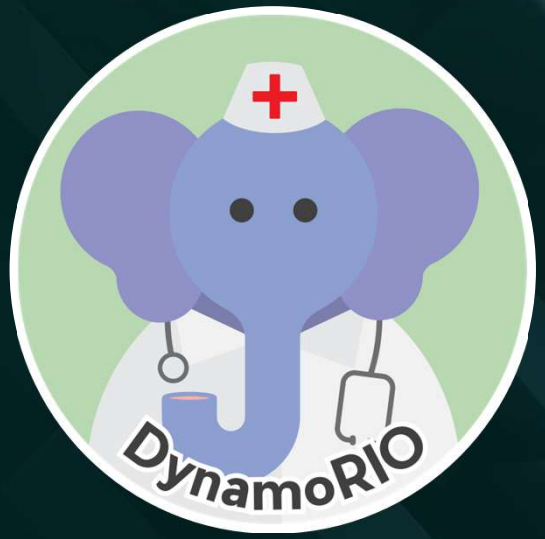
Fuzzing Infrastructure



Fuzzing Infrastructure



+



Fuzzing Infrastructure

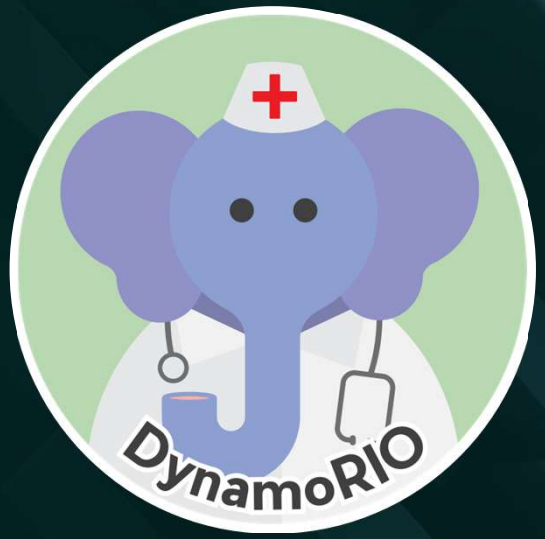
winafl Public

master 3 Branches 0 Tags

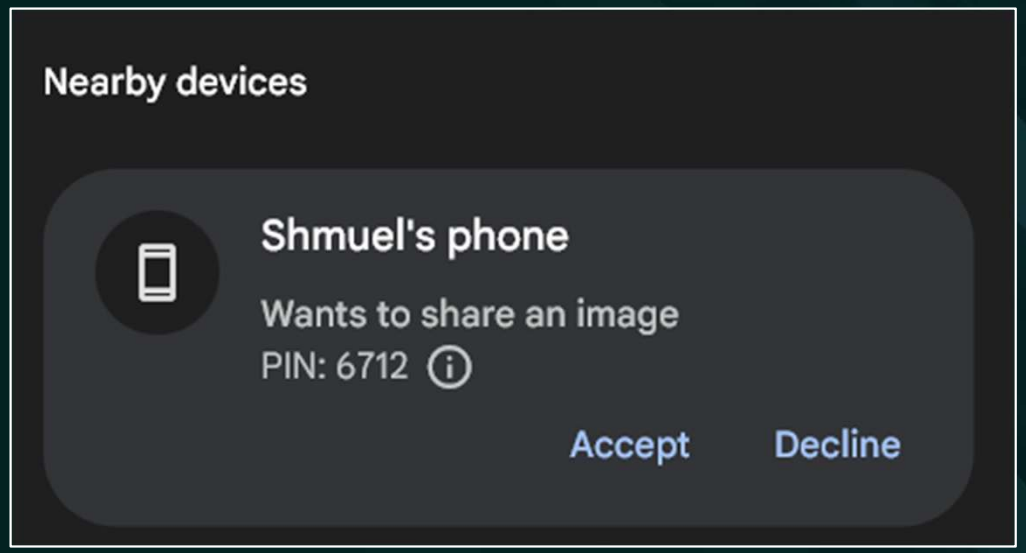
libprotobuf-mutator Public

master 1 Branch 4 Tags

+



“Accept” Patch



Auto-Accept feature

```
if (base::FeatureList::IsEnabled(features::kNearbySharingSelfShare)) {
  // Auto-accept self shares when not in high-visibility mode.
  if (share_target.for_self_share && !IsInHighVisibility()) {
    NS_LOG(INFO) << __func__ << ": Auto-accepting self share.";
    Accept(share_target, base::DoNothing());
  }
}
```

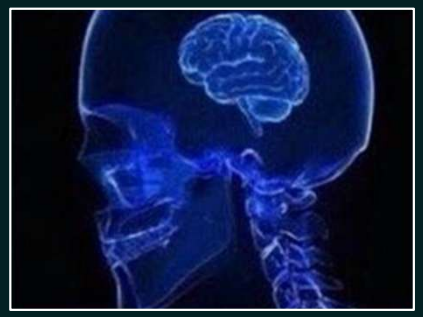
Auto-Accept feature

```
  // Auto-accept self shares when not in high-visibility mode.
  if (share_target.for_self_share && !IsInHighVisibility()) {
    NS_LOG(INFO) << __func__ << ": Auto-accepting self share.";
    Accept(share_target, base::DoNothing());
  }
}
```

Stateless to Stateful

Custom format to hold all packets of an entire session.

`[DWORD Length]` `[Serialized Offline Frame]`



Stateless

Stateful

```
                WinAFL 1.17 based on AFL 2.43b (nearby_share.exe)

- process timing ------------------------------------+- overall results ----+
        run time : 0 days, 0 hrs, 0 min, 19 sec       |  cycles done : 0      |
   last new path : 0 days, 0 hrs, 0 min, 18 sec       |  total paths : 2      |
 last uniq crash : none seen yet                      | uniq crashes : 0      |
  last uniq hang : none seen yet                      |   uniq hangs : 0      |
- cycle progress --------------------+- map coverage -+----------------------+
  now processing : 0 (0.00%)         |    map density : 28.53% / 34.41%      |
 paths timed out : 0 (0.00%)         | count coverage : 1.31 bits/tuple      |
- stage progress --------------------+ findings in depth --------------------+
  now trying : calibration           | favored paths : 1 (50.00%)            |
 stage execs : 23/40 (57.50%)        |  new edges on : 2 (100.00%)           |
 total execs : 64                    | total crashes : 0 (0 unique)          |
  exec speed : 1.28/sec (zzzz...)    |  total tmouts : 0 (0 unique)          |
- fuzzing strategy yields -----------+---------------+- path geometry -------+
   bit flips : 0/0, 0/0, 0/0                          |    levels : 2         |
  byte flips : 0/0, 0/0, 0/0                          |   pending : 2         |
 arithmetics : 0/0, 0/0, 0/0                          |  pend fav : 1         |
  known ints : 0/0, 0/0, 0/0                          | own finds : 0         |
  dictionary : 0/0, 0/0, 0/0                          |  imported : n/a       |
       havoc : 0/0, 0/0                               | stability : 67.35%    |
        trim : n/a, n/a                               +-----------------------+
-----------------------------------------------------+   [cpu000001:  12%]stage
times
```

Reproducible Crashes

4 non exploitable DoS vulnerabilities:

Invalid UTF8 continuation byte

Empty “Endpoint ID”

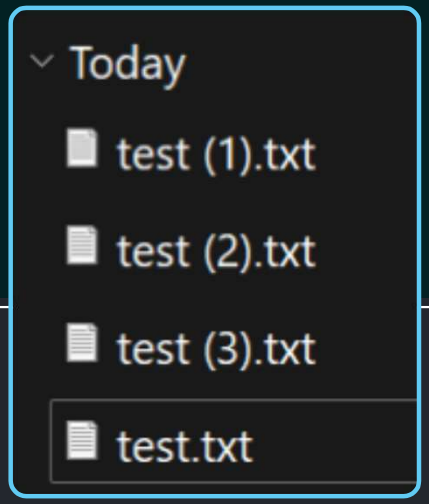
“Payload ID” set to 0

Fast 2 connections:

- same “nonce” in Connection Request
- UNKNOWN_VERSION set in Connection Response

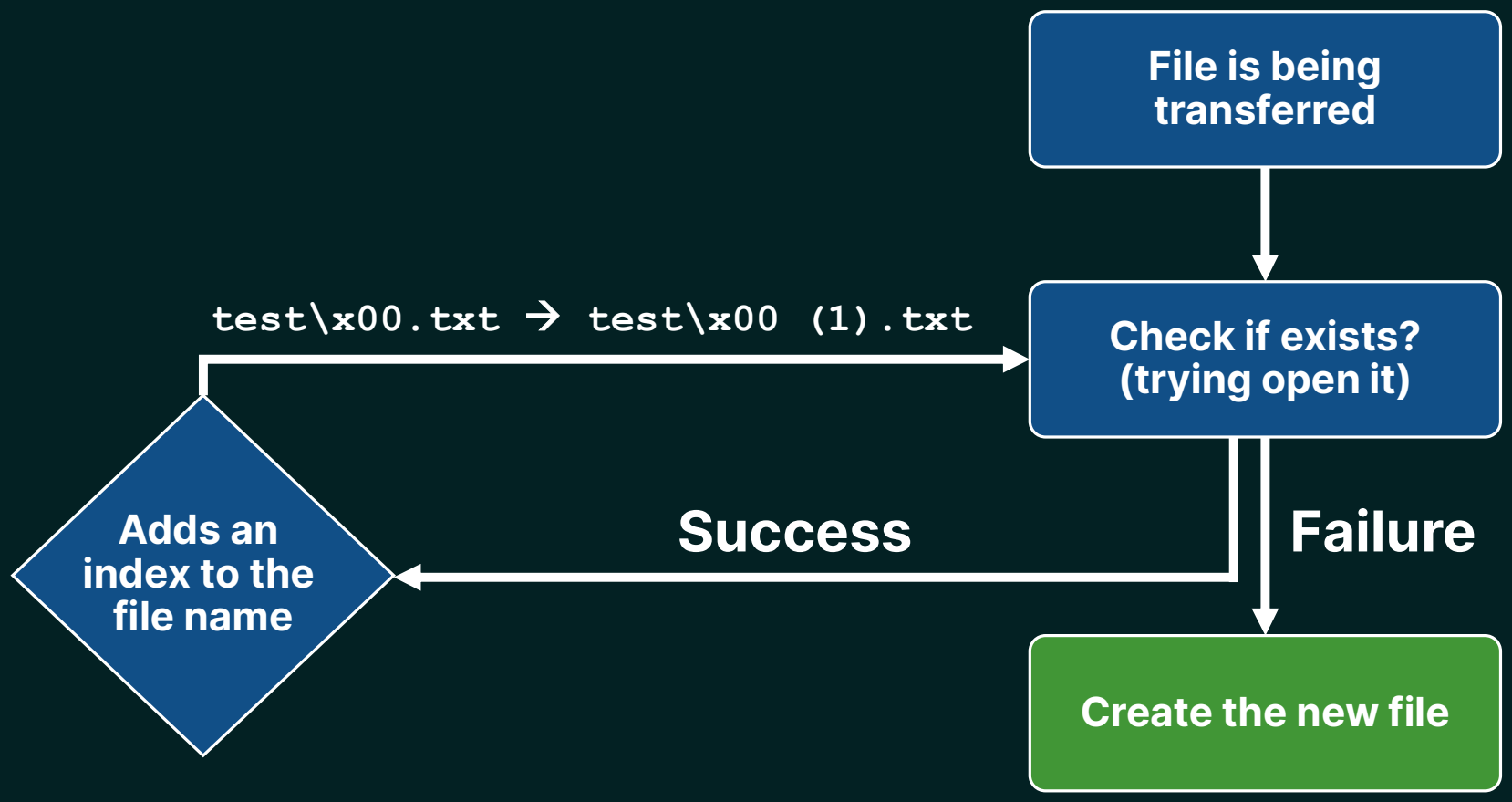
Reproducible Timeout

test.exe → test(1).exe




```
// Break the string at the dot.
auto file_name1 = file_name.substr(0, first);
auto file_name2 = file_name.substr(first);
...
// While we successfully open the file, keep incrementing the count.
int count = 0;
while (!(file.rdstate() & std::ifstream::failbit)) {
  file.close();
  target = (folder + file_name1 + L" (" + std::to_wstring(++count) + L")" + file_name2);
  ...
  file.open(target, std::fstream::binary | std::fstream::in);
}
```

Reproducible Timeout

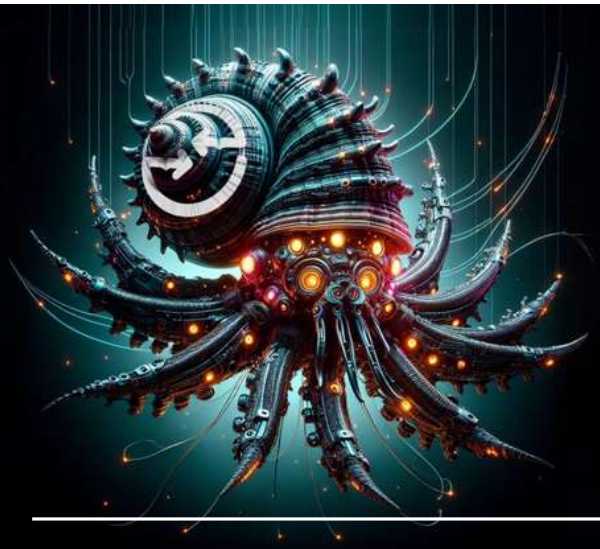


Time For Reflection

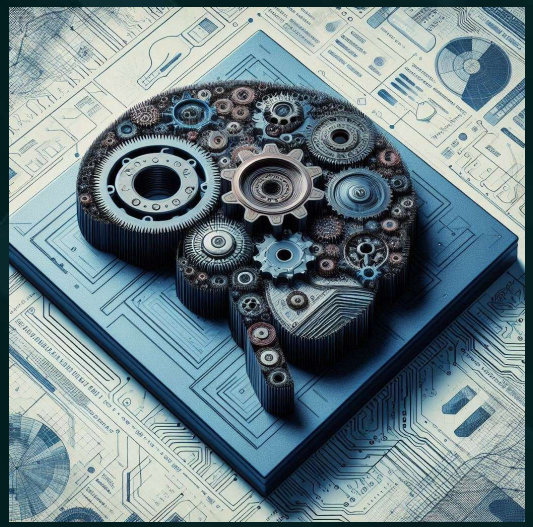
Fuzzer is running (slow but works)

Some unexploitable findings

Moving on to search for logic vulnerabilities, instead of creating the perfect fuzzer

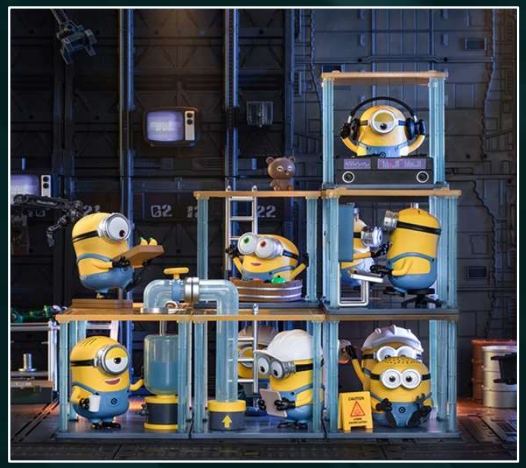


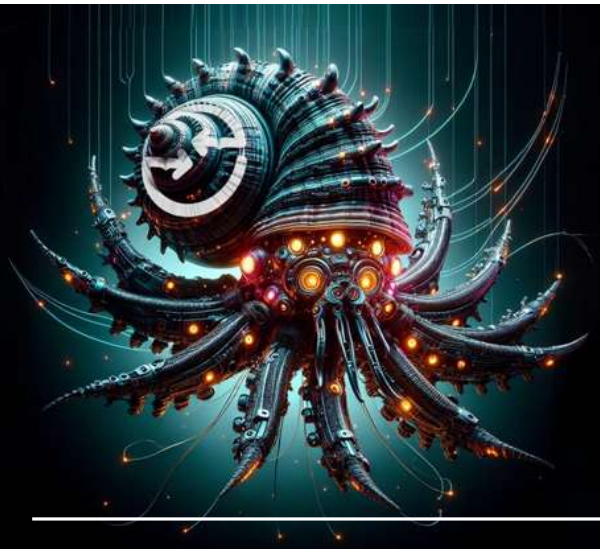
Logic Vulnerabilities



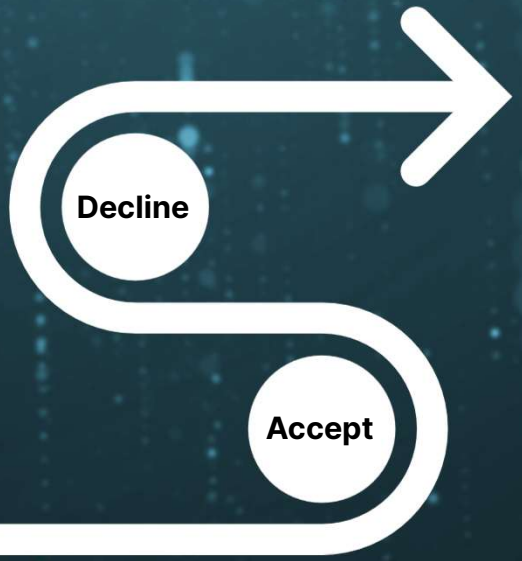
Google Quick Share's Code Design

- Extremely generic
- Handler class for each packet type
- Code is full of thread creations all over the place





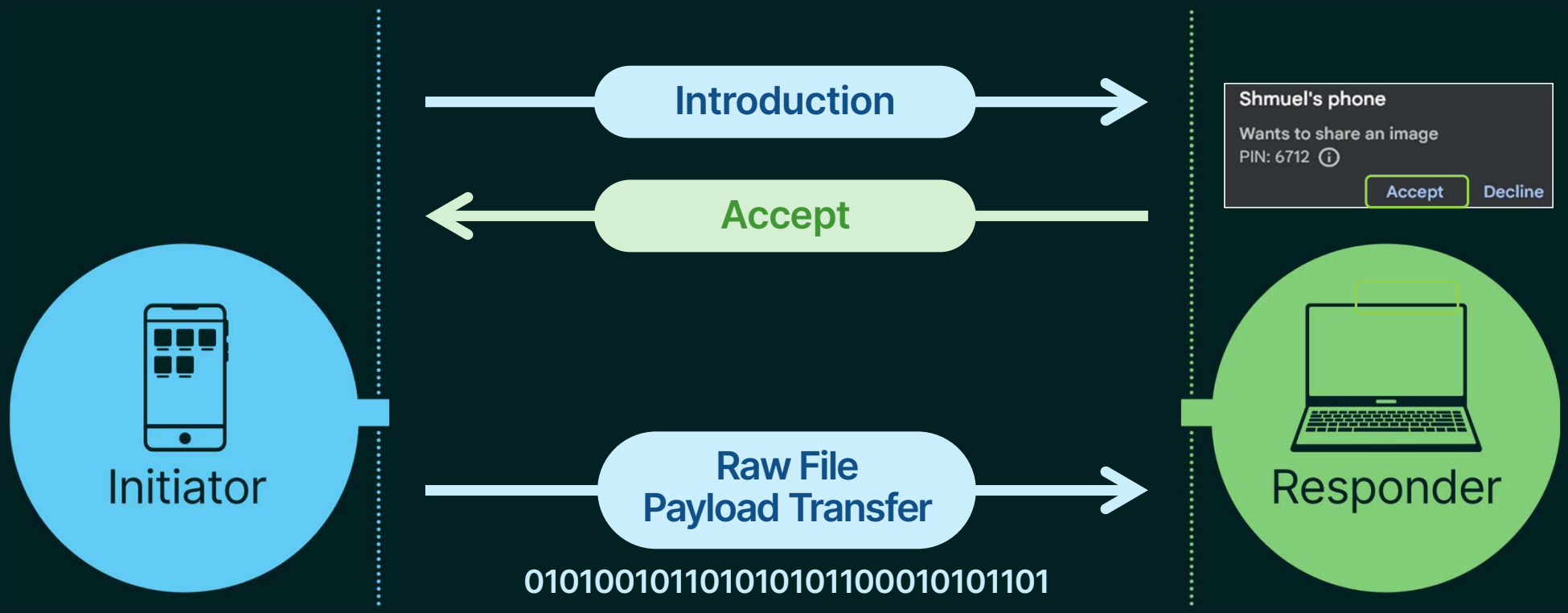
File Acceptance Bypass



Payload Transfer

INTRODUCTION & ACCEPT

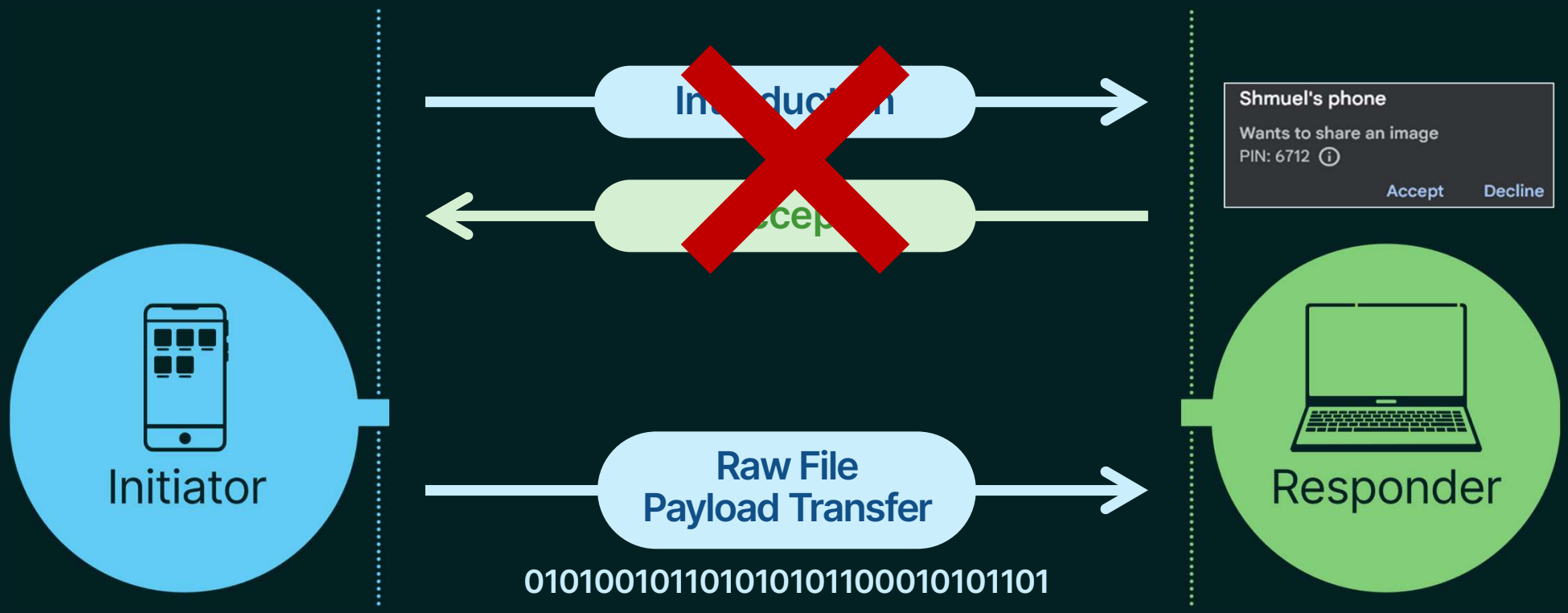
After paired Key Encryption:



Payload Transfer

INTRODUCTION & ACCEPT

After paired Key Encryption:



File Transfer Acceptance Bypass

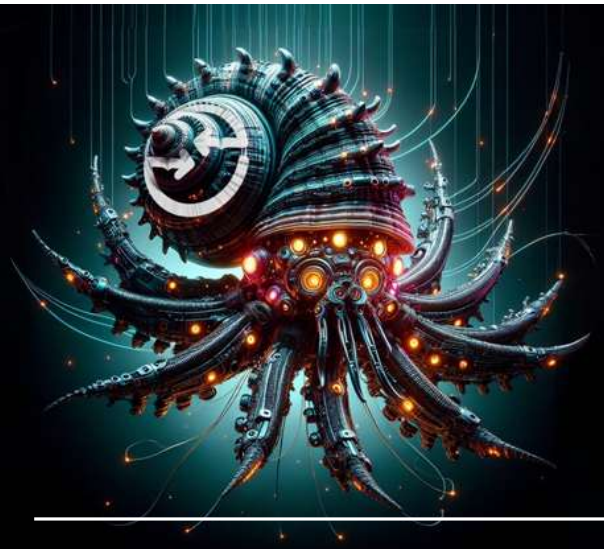
Bypasses all “Accept” in all visibility modes:

Your Devices

Contacts

Everyone

Acceptance Bypass - Demo



Forcing WiFi Connection

Connecting endpoints to our own AP

Bandwidth Upgrade Negotiation

Medium can be changed during the session.

```
// Accompanies Medium.WIFI_HOTSPOT.
message WifiHotspotCredentials {
  optional string ssid = 1;
  optional string password = 2;
  optional int32 port = 3;
  optional string gateway = 4 [default = "0.0.0.0"];
  // This field can be a band or frequency
  optional int32 frequency = 5 [default = -1];
}
```

```
// Accompanies Medium.WIFI_AWARE.
message WifiAwareCredentials {
  optional string service_id = 1;
  optional bytes service_info = 2;
  optional string password = 3;
}
```

```
// Accompanies Medium.WIFI_LAN.
message WifiLanSocket {
  optional bytes ip_address = 1;
  optional int32 wifi_port = 2;
}
```

```
// Accompanies Medium.BLUETOOTH.
message BluetoothCredentials {
  optional string service_name = 1;
  optional string mac_address = 2;
}
```

```
// Accompanies Medium.WEB_RTC
message WebRtcCredentials {
  optional string peer_id = 1;
  optional LocationHint location_hint = 2;
}
```

Past Research

Android devices forced to connect to a WiFi network

~30 seconds' max

Mitigated by Google

- Android devices no longer connect to internet through a Quick Share Bandwidth Upgrade WiFi network

Quick Share on Windows

Internet access is permitted through a Bandwidth Upgrade WiFi network!

Quick Share on Windows

Internet access is permitted through a Bandwidth Upgrade WiFi network!

We can now sniff responder internet traffic

Standard stones may sometimes be forged into deadly drones

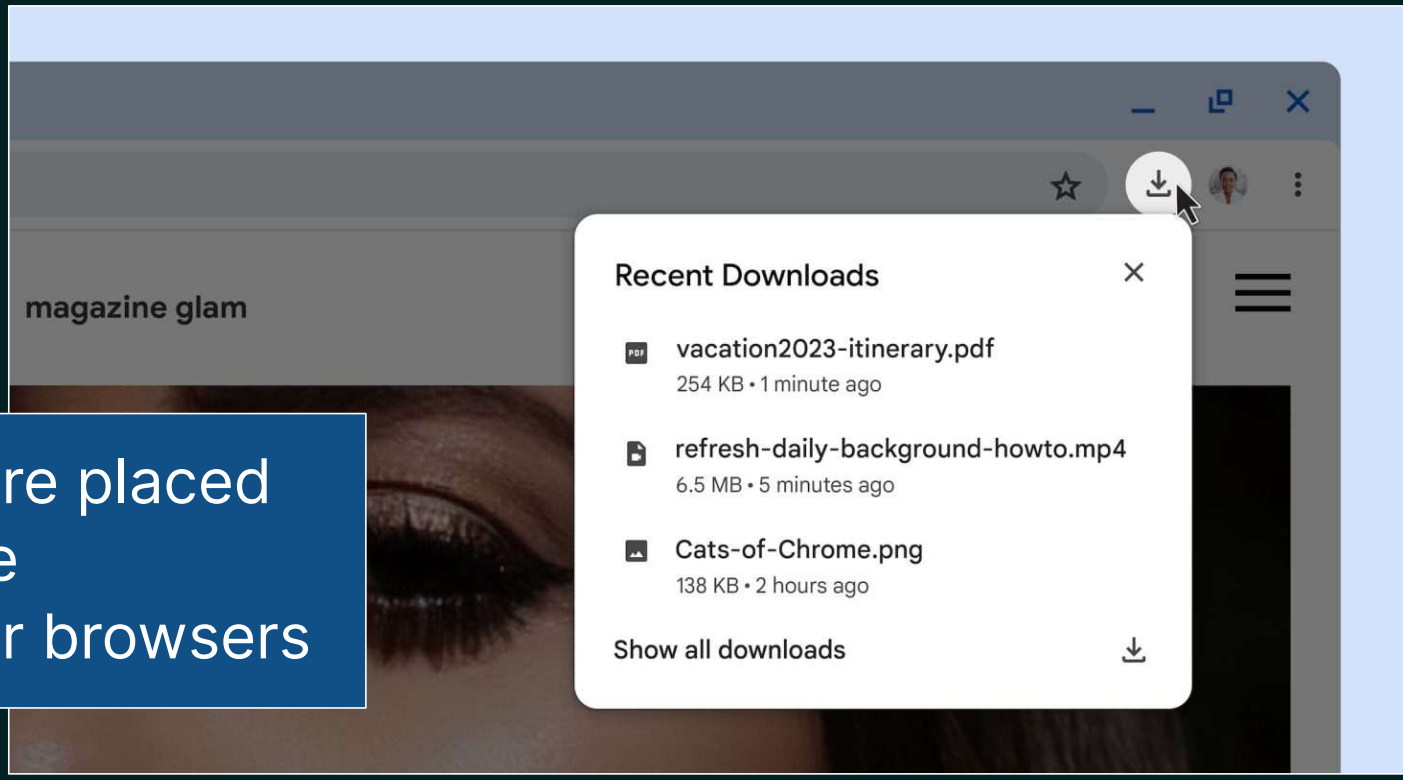
Primary Abilities We Achieved

- Create files in “Downloads” without approval
- WiFi MITM (30 sec max)
- Crash Quick Share
- Force Quick Share to continuously open a file

WiFi MITM —
Application Layer Encryption

Encrypted application layer is a standard.
Leveraging MITM for straight forward RCE won't work for most use cases.

Downloading an Insight



Quick Share's files are placed in "Downloads" - the downloads folder for browsers

Potential RCE Goal

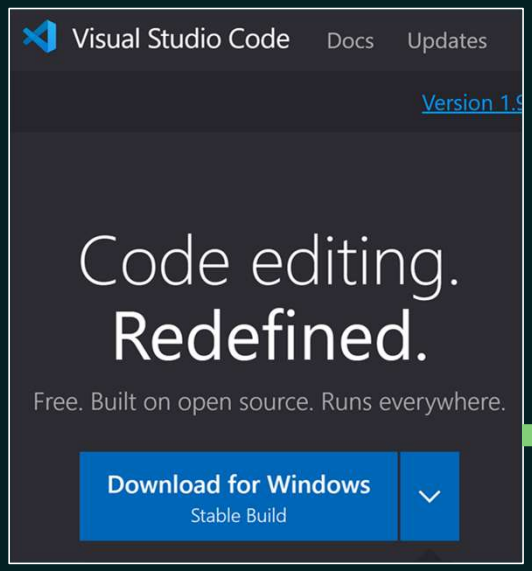
Goal:

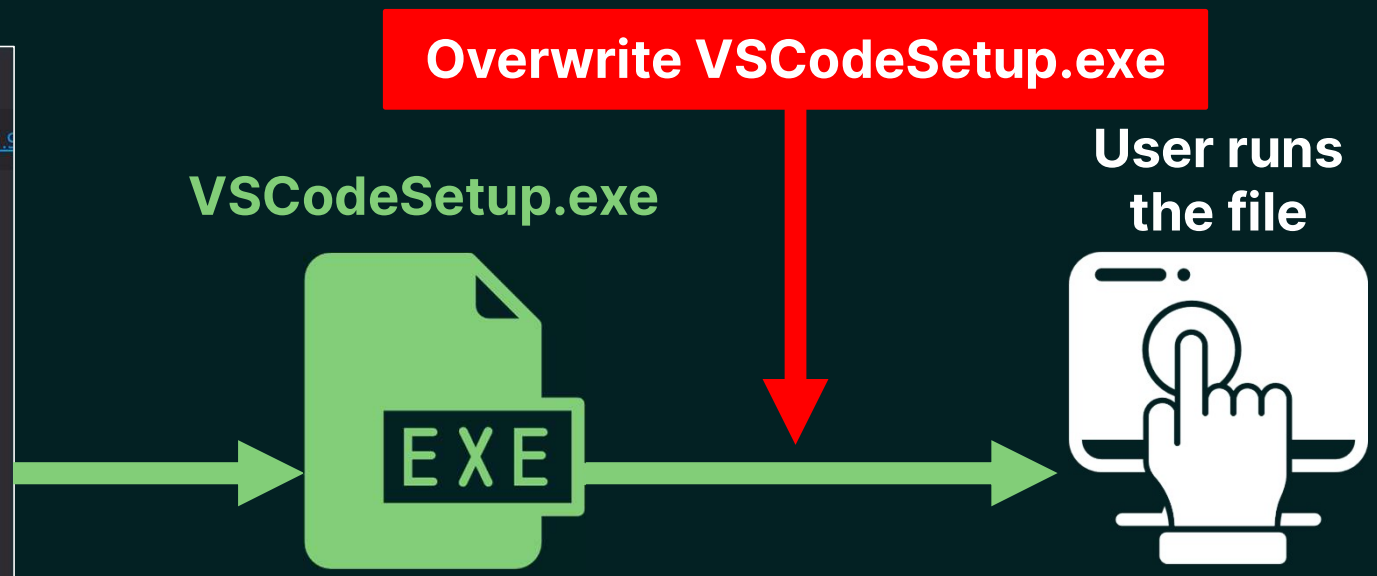
Overwrite an executable downloaded by a victim before it runs

Needed Abilities:

- ? Know downloaded executable file names
- ? Overwrite files (not just create)

Potential RCE Goal





Lasting WiFi MITM

Trying to bridge the gaps anyway, starting with making the WiFi connection last

Lasting WiFi Connection

Quick Share’s Relaunch Scheduled Task

Quick Share Relaunch Scheduled Task –
Every 15 minutes

Quick Share Relaunch	At 1:27 AM on 7/2/2024 - After triggered, repeat every 15 minutes indefinitely.

Force
WiFi
Connection

Crash

We're now
MITM

Knowing Downloaded File Names

Downloaded Files Metadata - Domain

TLS Client Hello - Server Name Indication Extension

Current filter: ip.dst == 13.107.246.62

No.	Time	Source	Destination	Protocol	Length	Info
1…	135.…	10.…	13.10…	TLSv1.3	439	Client Hello

```
Extension: server_name (len=26)
  Type: server_name (0)
  Length: 26
  Server Name Indication extension
    Server Name list length: 24
    Server Name Type: host_name (0)
    Server Name length: 21
    Server Name: code.visualstudio.com
```

Downloaded Files Metadata - Size

Single TCP session per download

↓

Approximate download size

Knowing Downloaded File Names

Installer Domain

+

Approximate Size

=

File Name Accurate Guess

code.visualstudio.com

+

95 MB

=

VSCodeUserSetup-x64-1.91.0.exe

Knowing Downloaded File Names

Installer Domain

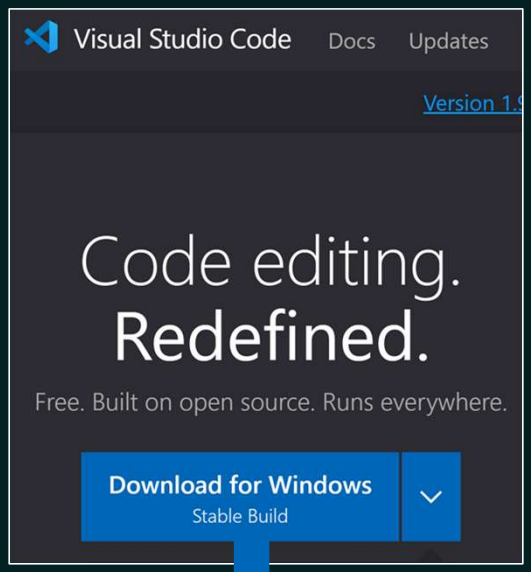
+

Approximate Size

=

File Name Accurate Guess

code.visualstudio.com



95 MB

VSCodeUserSetup-x64-1.91.0.exe

Improvement – Monitoring Domain Paths

notepad-plus-plus.org

Improvement – Monitoring Domain Paths

notepad-plus-plus.org

github.com/.../npp.8.6.9.Installer.x64.exe

Improvement – Monitoring Domain Paths

notepad-plus-plus.org

github.com/.../npp.8.6.9.Installer.x64.exe

objects.githubusercontent.com

New MITM Technique

Map “Domain Paths” to executables + their sizes

Wait for “Domain Path” hit

Count TCP data

If - TCP data <= actual executable size + 15%:

We know it’s the executable

Force WiFi Connection

Detect EXE Download Name

Crash

Potential RCE Goal

Goal:

Overwrite an executable downloaded by a victim before it runs

Needed Abilities:

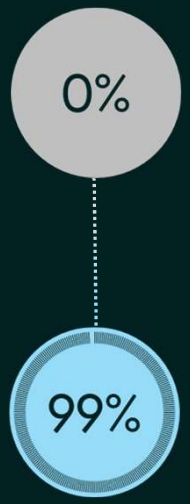
- ✓ Know downloaded executable file names
- ? Overwrite files (not just create)

Chrome’s Download Process

0%

Check if VSCodeSetup.exe exists

Chrome’s Download Process



Check if VSCodeSetup.exe exists

Unconfirmed 550383.crdownload

Chrome's Download Process

0%

Check if VSCodeSetup.exe exists

99%

Unconfirmed 550383.crdownload

100%

VSCodeSetup.exe

Overwriting Files

0%

Check if VSCodeSetup.exe exists

99%

Unconfirmed 550383.crdownload

100%

VSCodeSetup.exe

Overwriting Files

0%

`Check if VSCodeSetup.exe exists`

99%

`Unconfirmed 550383.crdownload`

Hold last TCP packet

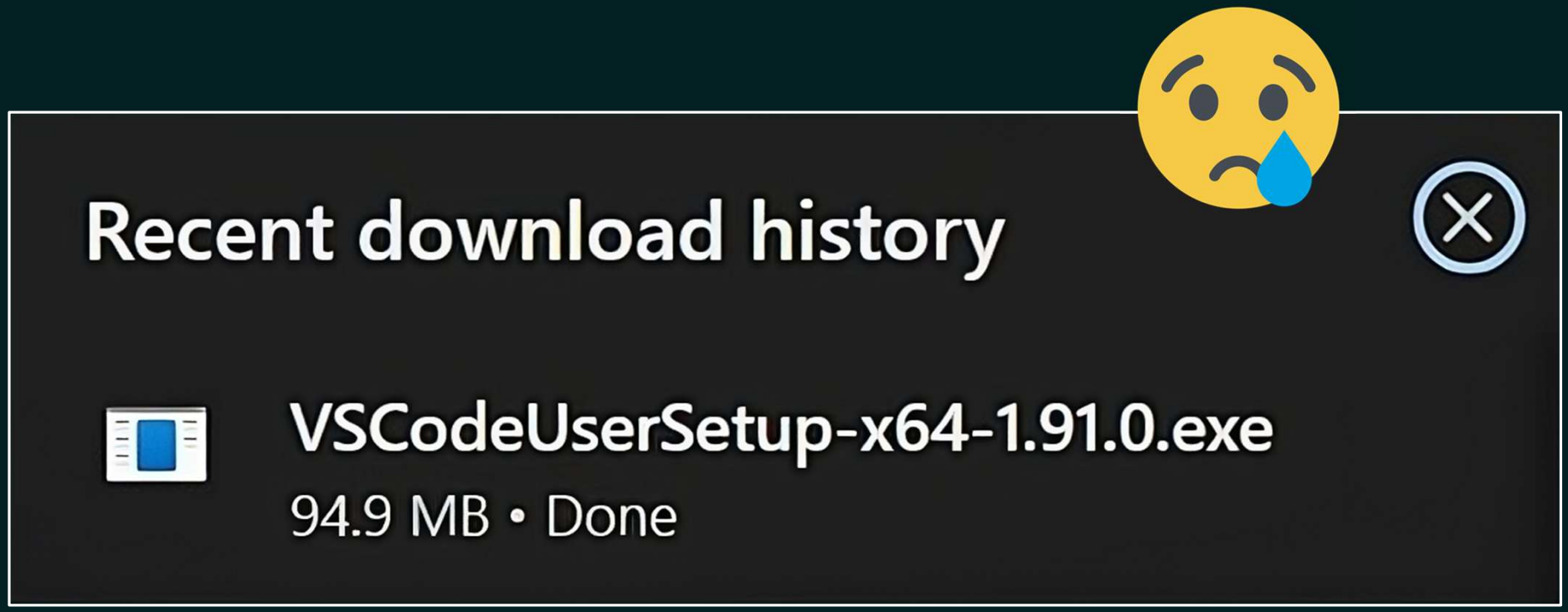
Send malicious `VSCodeSetup.exe`

100%

`VSCodeSetup.exe`

Overwriting Chrome's Download

.crwd is renamed and our file is deleted



Overwriting Chrome's Download

Can we maybe prevent our file from being deleted?

1. Send malicious `VSCodeSetup.exe`
2. Make Quick Share continuously open `VSCodeSetup.exe`

Overwriting Chrome’s Download

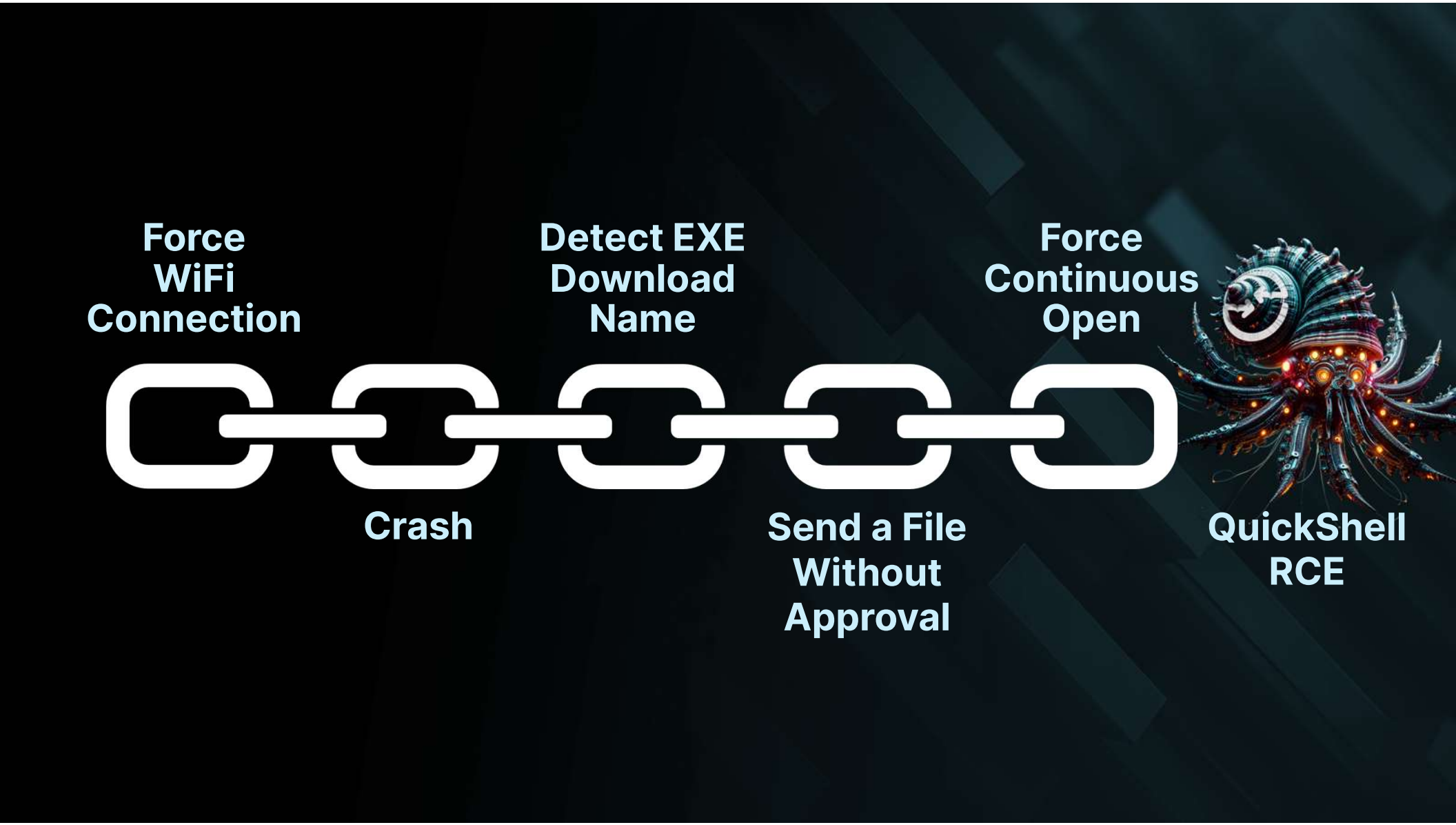
Result:

Chrome deletes the .crdownload file

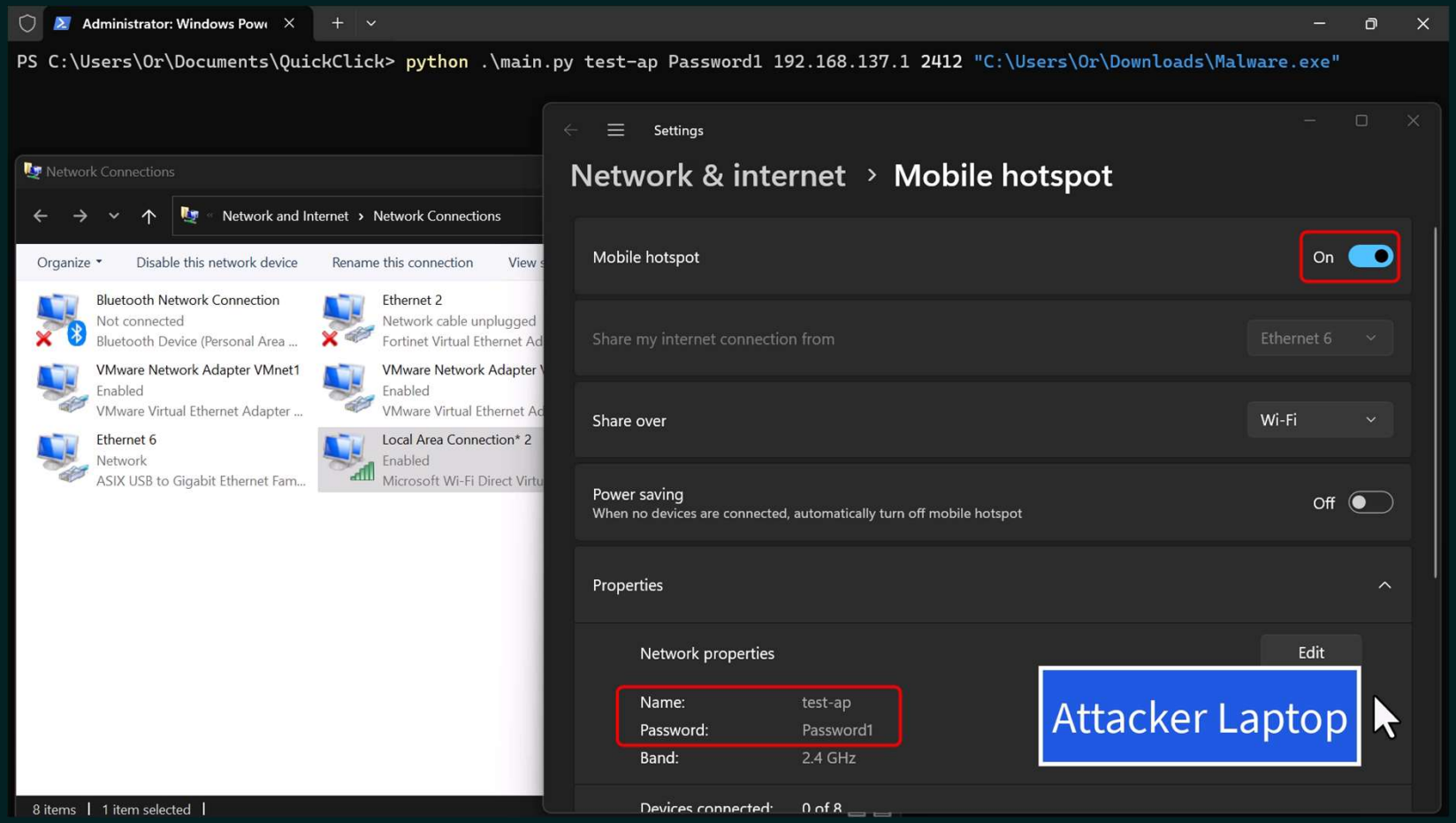
Leaves our malicious file in place

Reports successful download completion

Refers to our malicious file


Force WiFi Connection
Crash
Detect EXE Download Name
Send a File Without Approval
Force Continuous Open
QuickShell RCE

Demo


Administrator: Windows Pow
PS C:\Users\Or\Documents\QuickClick> python .\main.py test-ap Password1 192.168.137.1 2412 "C:\Users\Or\Downloads\Malware.exe"
Network Connections
Network and Internet › Network Connections
Organize
Disable this network device
Rename this connection
Bluetooth Network Connection
Not connected
Bluetooth Device (Personal Area ...
Ethernet 2
Network cable unplugged
Fortinet Virtual Ethernet Ad
VMware Network Adapter VMnet1
Enabled
VMware Virtual Ethernet Adapter ...
VMware Network Adapter
Enabled
VMware Virtual Ethernet Ad
Ethernet 6
Network
ASIX USB to Gigabit Ethernet Fam...
Local Area Connection* 2
Enabled
Microsoft Wi-Fi Direct Virtu
8 items | 1 item selected |
Settings
Network & internet › Mobile hotspot
Mobile hotspot
On
Share my internet connection from
Ethernet 6
Share over
Wi-Fi
Power saving
When no devices are connected, automatically turn off mobile hotspot
Off
Properties
Network properties
Edit
Name: test-ap
Password: Password1
Band: 2.4 GHz
Attacker Laptop
Devices connected: 0 of 8

Extra Vulnerabilities

Limited path traversal (only to parent folder)

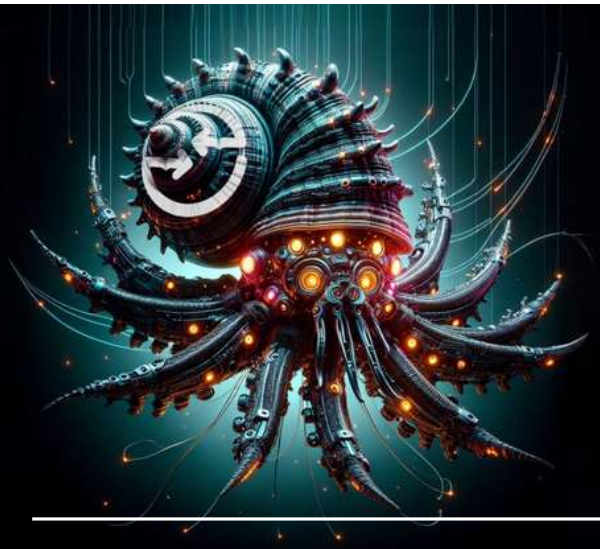
DoS (based on the MagicDot research)

10 Vulnerabilities

1. Remote Unauthorized File Write in Quick Share for Windows
2. Remote Unauthorized File Write in Quick Share for Android
3. Remote Forced WiFi Connection in Quick Share for Windows
4. Remote Directory Traversal in Quick Share for Windows
5. Remote DoS in Quick Share for Windows – Endless Loop
6. Remote DoS in Quick Share for Windows – Assert Failure
7. Remote DoS in Quick Share for Windows – Assert Failure
8. Remote DoS in Quick Share for Windows – Unhandled Exception
9. Remote DoS in Quick Share for Windows – Unhandled Exception
10. Remote DoS in Quick Share for Windows – Unhandled Exception

Fixed Vulnerabilities

- Remote Unauthorized File Write in Quick Share for Windows
- Remote Unauthorized File Write in Quick Share for Android
- Remote Forced WiFi Connection in Quick Share for Windows
- Remote Directory Traversal in Quick Share for Windows
- Remote DoS in Quick Share for Windows – Endless Loop
- Remote DoS in Quick Share for Windows – Assert Failure
- Remote DoS in Quick Share for Windows – Assert Failure
- Remote DoS in Quick Share for Windows – Unhandled Exception
- Remote DoS in Quick Share for Windows – Unhandled Exception
- Remote DoS in Quick Share for Windows – Unhandled Exception



Fixes & Patch Bypasses

Patch – Invalid UTF8 Continuation Byte

Reported to Google about Invalid UTF8 continuation bytes crashing Quick Share

Example we provided – “\x00FileName”

Google’s patch – Verifies file names don’t start with “\x00”

Bypass – Invalid UTF8 Continuation Byte

Instead of “\x00”, setting a different invalid UTF8 continuation byte in file names

Example – “\xc5\xffFileName”

Result:

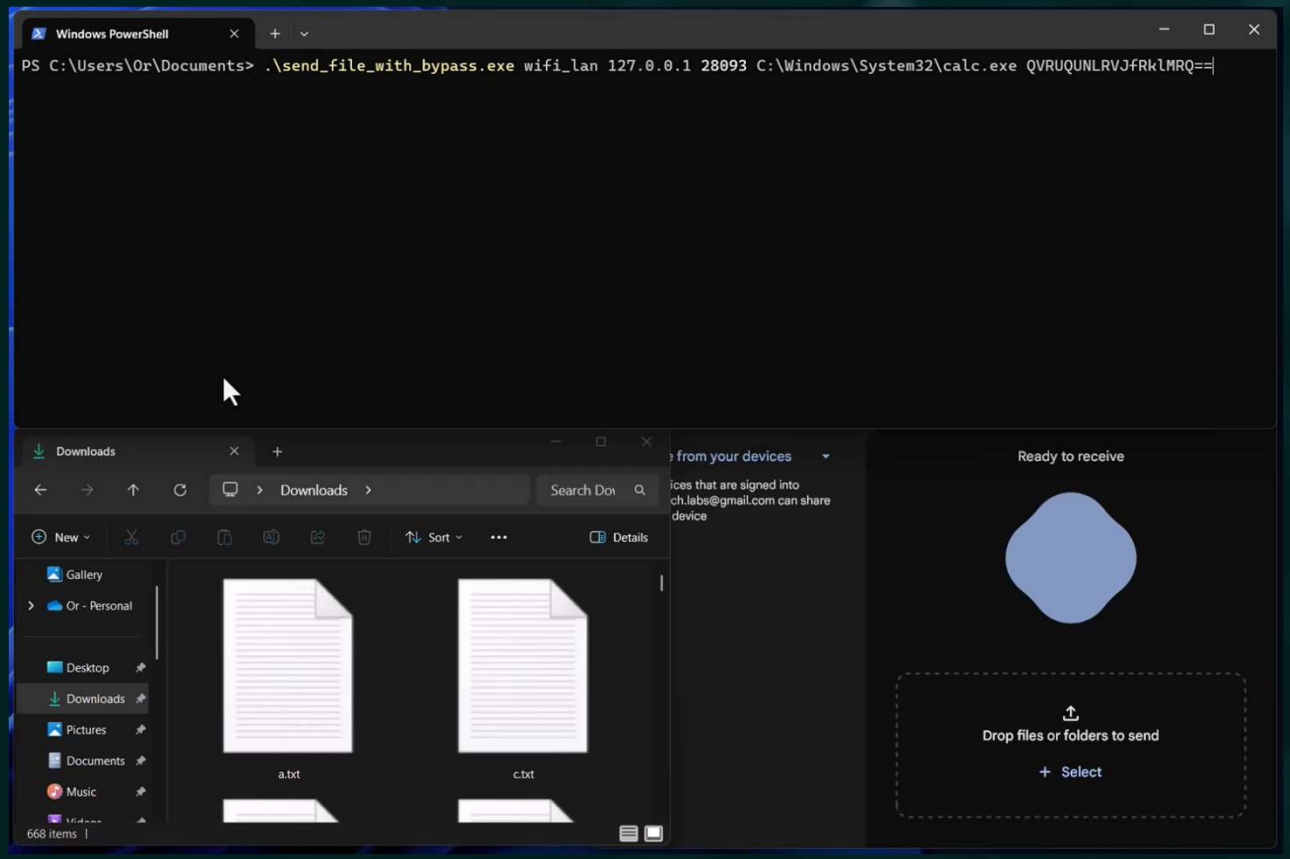
Quick Share crashes again

Patch - Remote Unauthorized File Write

Files are still written to disk on Windows but are later deleted.

Google calls them:

“Unknown Files”



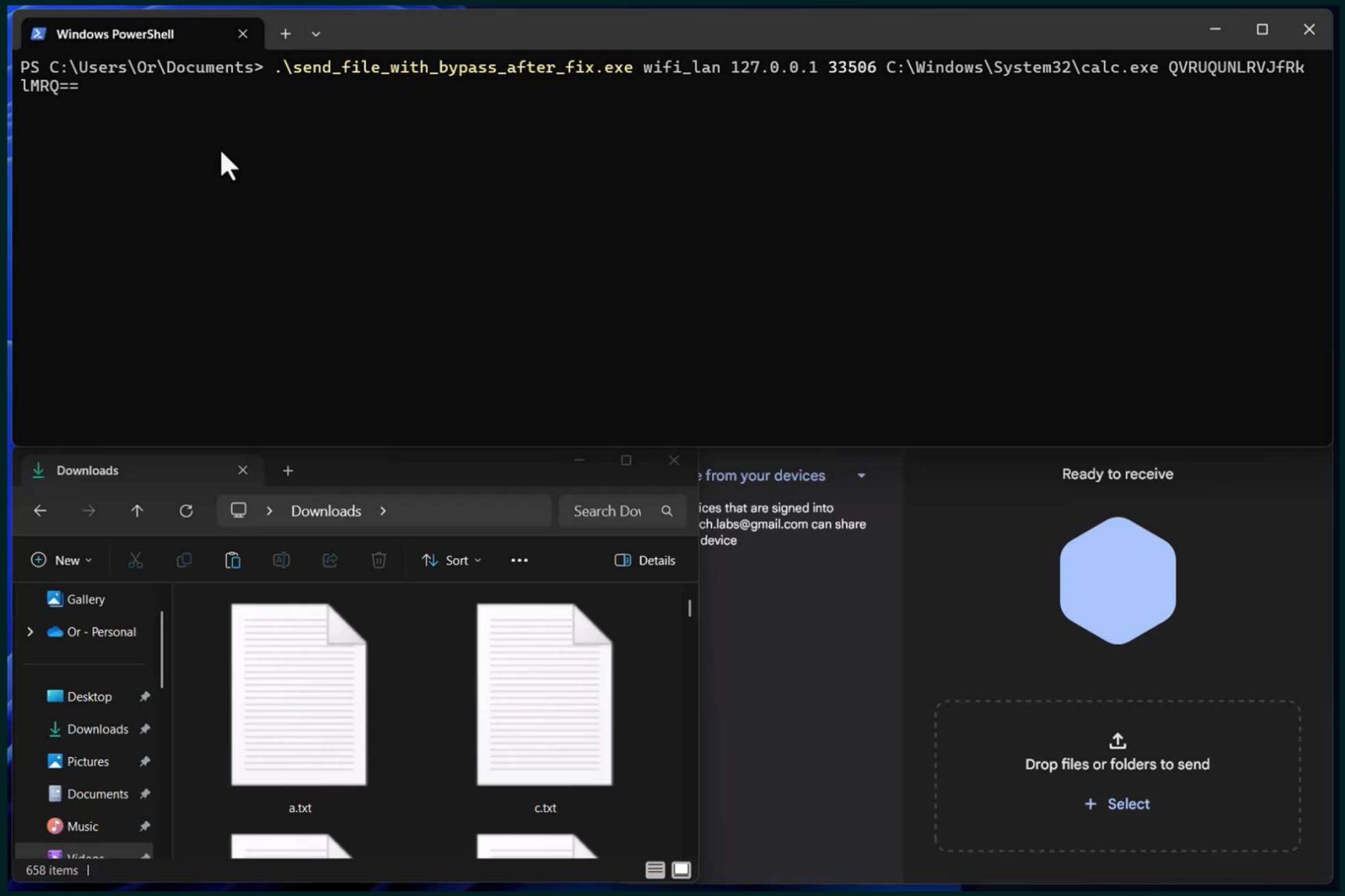
```
void NearbySharingServiceImpl::RemoveIncomingPayloads(
    const IncomingShareSession& session) {
  LOG(INFO) << __func__ << ": Cleaning up payloads due to transfer failure";
  nearby_connections_manager_->ClearIncomingPayloads();
  std::vector<std::filesystem::path> files_for_deletion;
  auto file_paths_to_delete =
      nearby_connections_manager_->GetAndClearUnknownFilePathsToDelete();
  for (auto it = file_paths_to_delete.begin(); it != file_paths_to_delete.end();
       ++it) {
    VLOG(1) << __func__ << ": Has unknown file path to delete.";
    files_for_deletion.push_back(*it);
  }
  std::vector<std::filesystem::path> payload_file_path =
      session.GetPayloadFilePaths();
  files_for_deletion.insert(files_for_deletion.end(), payload_file_path.begin(),
                            payload_file_path.end());
  file_handler_.DeleteFilesFromDisk(std::move(files_for_deletion), []() {});
}
```

Bypass - Remote Unauthorized File Write

Send two FILE Payload Transfer Frame with the same Payload ID

Result:

Only the first file is deleted


Windows PowerShell
PS C:\Users\Or\Documents> .\send_file_with_bypass_after_fix.exe wifi_lan 127.0.0.1 33506 C:\Windows\System32\calc.exe QVRUQUNLRVJfRk
lMRQ==
Downloads
Downloads
Search Do
New
Sort
Details
Gallery
Or - Personal
Desktop
Downloads
Pictures
Documents
Music
a.txt
c.txt
658 items
from your devices
ices that are signed into
ch.labs@gmail.com can share
device
Ready to receive
Drop files or folders to send
Select

Google’s Response

"We greatly appreciate research from the security community that helps keep our users safe. We have deployed fixes for all of the reported vulnerabilities. To our knowledge, these vulnerabilities have not been exploited in the wild. No action is required by Quick Share users. The fixes will be automatically applied.

Developers using the open source repository can refer to the CVEs for further information on how to apply the fixes:

CVE-2024-38271
CVE-2024-38272

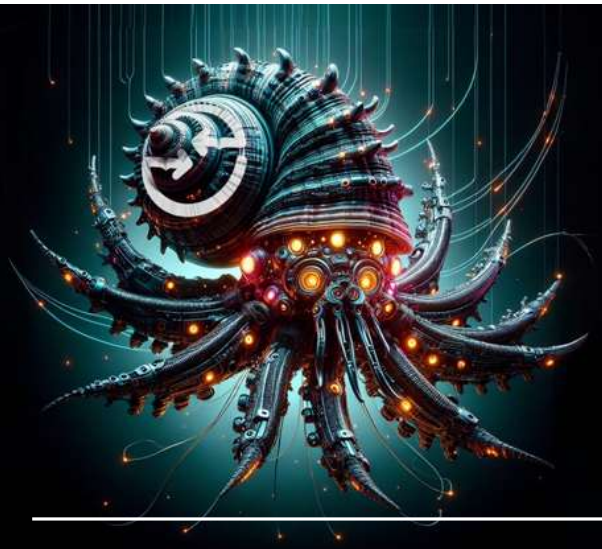
July 23rd, 2024

CVEs

CVE-2024-38271 – Forcing a lasting WiFi connection

CVE-2024-38272 – File approval dialog bypass

CVE-2024-10668 - Fix Bypass for CVE-2024-38272



Takeaways

Takeaways

Standard stones may sometimes be forged into deadly drones

Takeaways

Standard stones may sometimes be forged into deadly drones

It's crucial for vendors and organizations not to underestimate seemingly simple bugs or known issues

Takeaways

Standard stones may sometimes be forged into deadly drones

It's crucial for vendors and organizations not to underestimate seemingly simple bugs or known issues

It’s crucial to not fixate solely on memory corruption and fuzzing techniques when examining a program's security

GitHub + Q&A

QuickShell

@oryair1999

https://www.linkedin.com/in/or-yair/

@_BinWalker_

https://www.linkedin.com/in/the-shmuel-cohen/