Android Exploits 101

Maddie Stone
@maddiestone

whoami - Maddie Stone (she/her)

- Security researcher at Project Zero
  - Focused on 0-day exploits actively used in the wild

Goal

- Understand the “shape” of modern 0-day exploits for Android
- Foundation of current exploit threats to Android users to inform investments in defenses and detection
- Join the exploits fun! :)

In-the-wild 0-day exploits

- Goal: Learn from 0-days exploited in-the-wild to make 0-day hard.

- Tracking spreadsheet of known itw 0-days
- Root cause analyses: https://googleprojectzero.github.io/0days-in-the-wild/rca.html
- Year-in-review reports: 2020, 2019
- A World Where 0-day is Hard keynote [video]

0-day exploit:
an exploit defenders don’t yet know about

CVE-2021-1905: Qualcomm Adreno GPU memory mapping use-after-free

Ben Hawkes, Project Zero

The Basics

Disclosure or Patch Date: 1 May 2021

Product: Qualcomm Adreno GPU

Advisory: https://www.qualcomm.com/company/product-security/bulletins/may-2021-bulletin

Affected Versions: Prior to Android 2021-05-01 security patch level

Note: the Qualcomm Adreno GPU kernel driver may be used in other platforms aside from Android, but the following analysis was performed with Android in mind, since Android is a high priority area of interest for Project Zero.

First Patched Version: Android 2021-05-01 security patch level

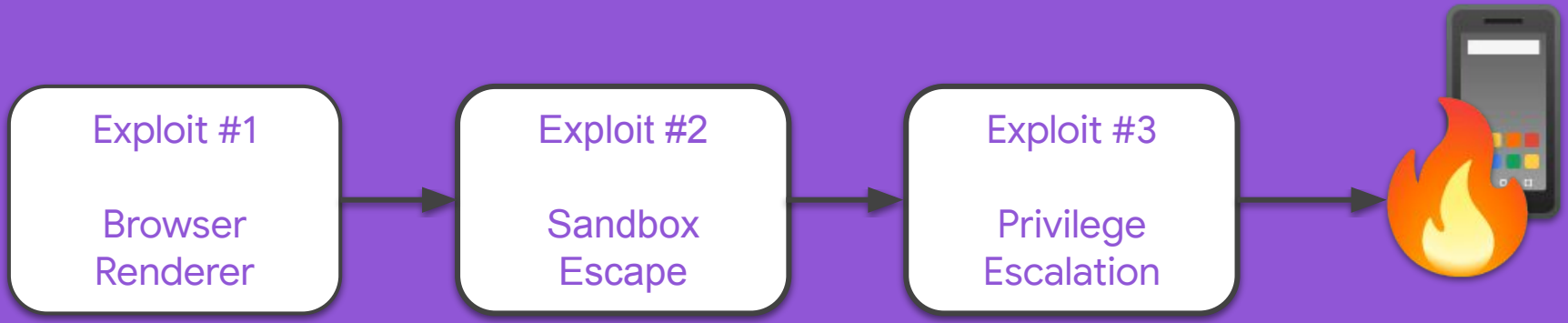
Issue/Bug Report: N/A

Patch CL:
https://source.codeaurora.org/quic/la/kernel/msm-4.9/commit/?id=d236d315145f8250523ce9e14897d62e5d6639fc
https://source.codeaurora.org/quic/la/kernel/msm-4.9/commit/?id=ec3c8cf016991818ca286c4fd92255393c211405

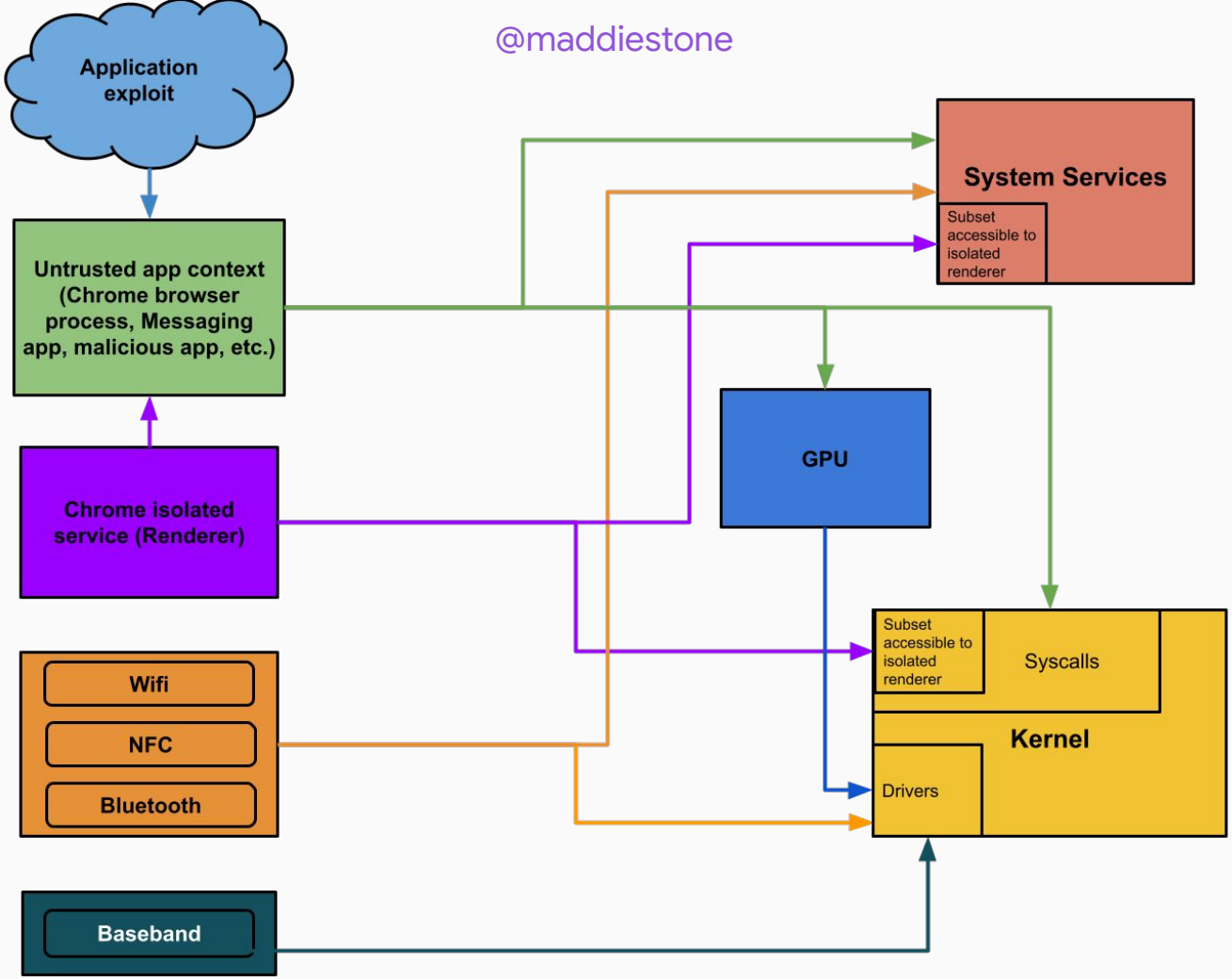
Bug-Introducing CL: N/A

Reporter(s): N/A

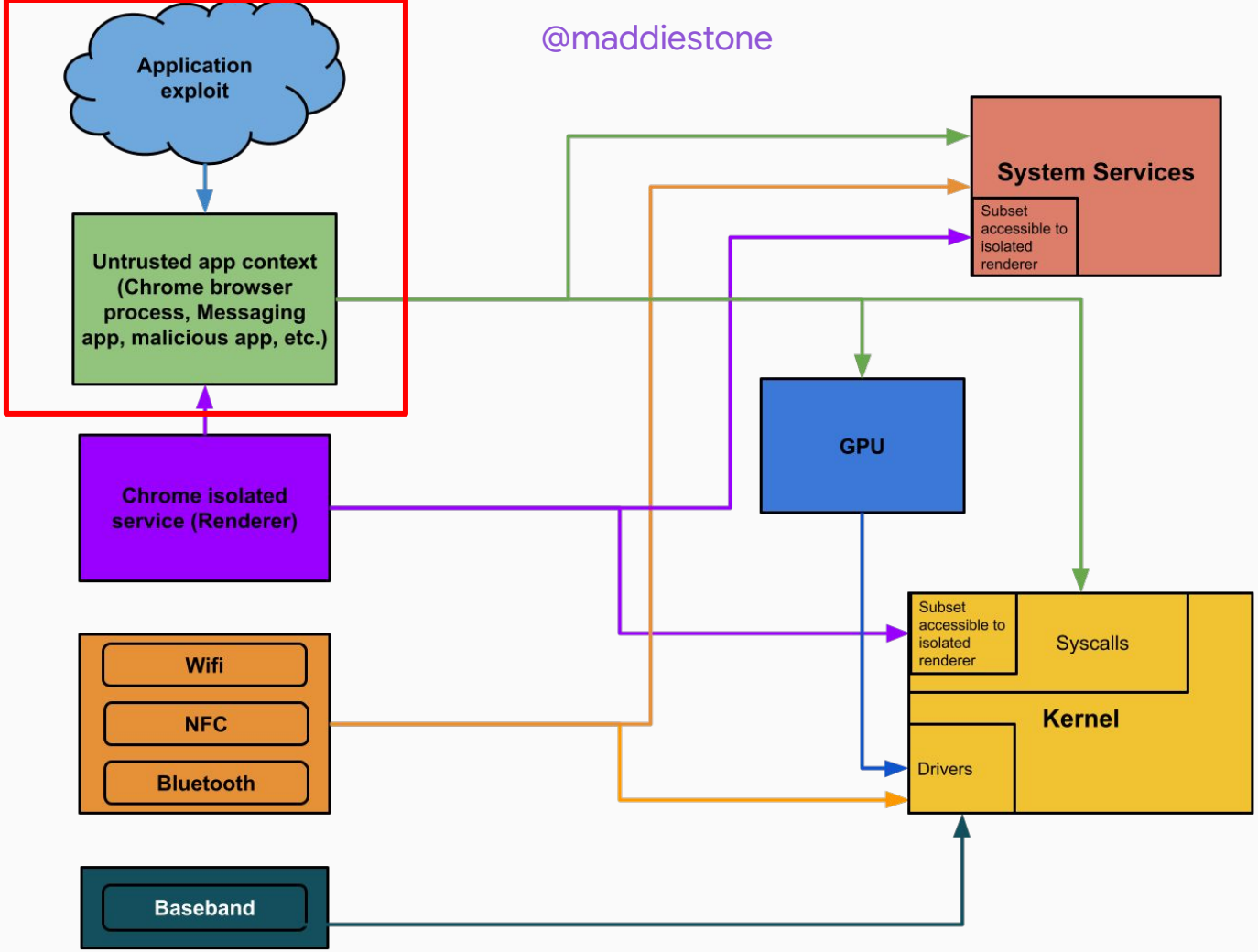
https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-1905.html


Exploit #1
Browser Renderer
Exploit #2
Sandbox Escape
Exploit #3
Privilege Escalation

@maddiestone



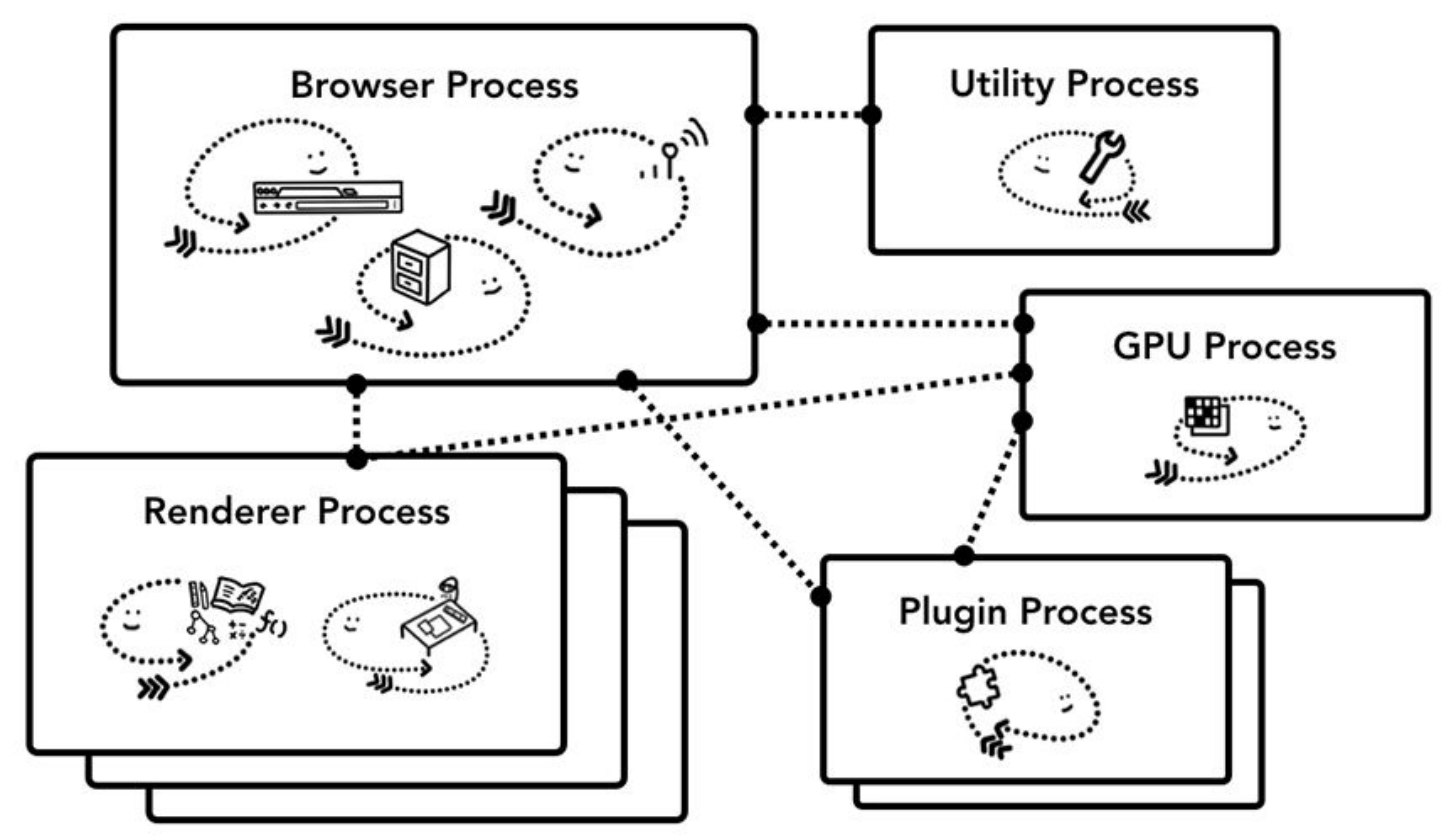
@maddiestone



Application Exploit - Messaging/Video Conferencing

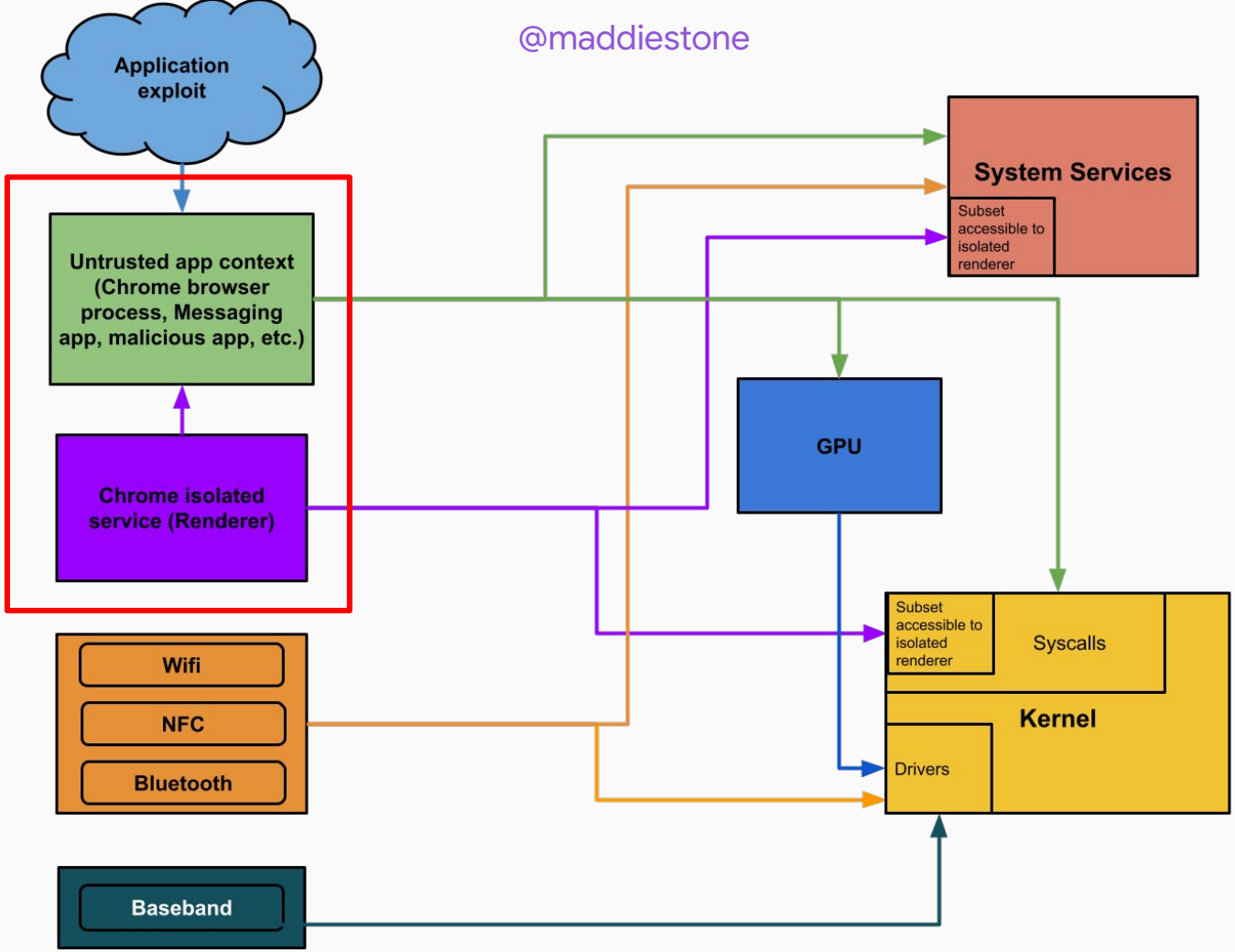
- Good 0-click attack surface
- For video/audio conferencing, WebRTC is a common library
  - Most android apps will have their conferencing & messaging code in a native library
- Lots of options:
  - Android messages, Duo, WhatsApp, Signal, Telegram, WeChat, and more...
- Examples:
  - WhatsApp 0-day CVE-2019-3568 [Checkpoint blog, P0 slides, P0 recording]
  - Exploiting Android Messengers
  - Detailed series on exploit chain for Samsung via MMS

Chrome (and other browsers)



https://developers.google.com/web/updates/2018/09/inside-browser-part1

@maddiestone



Chrome processes on Android

Browser process:

```
u:r:untrusted_app:s0:c216,c256,c512,c768  u0_a216    com.android.chrome
```

App zygote process (manages the isolated/renderer processes):

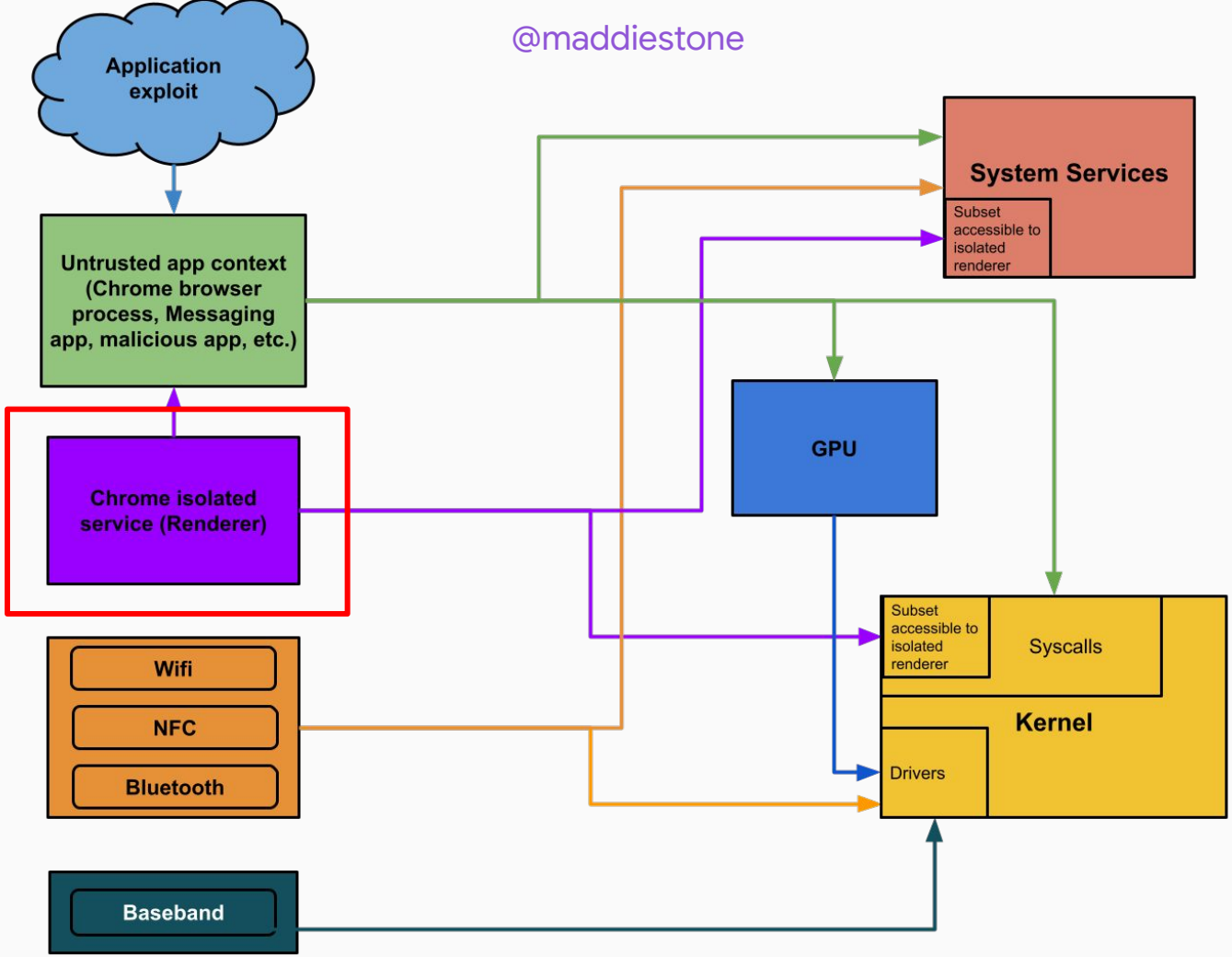
```
u:r:app_zygote:s0:c512,c768               u0_a216      com.android.chrome_zygote
```

GPU process:

```
u:r:untrusted_app:s0:c216,c256,c512,c768  u0_a216    com.android.chrome:privileged_process0
```

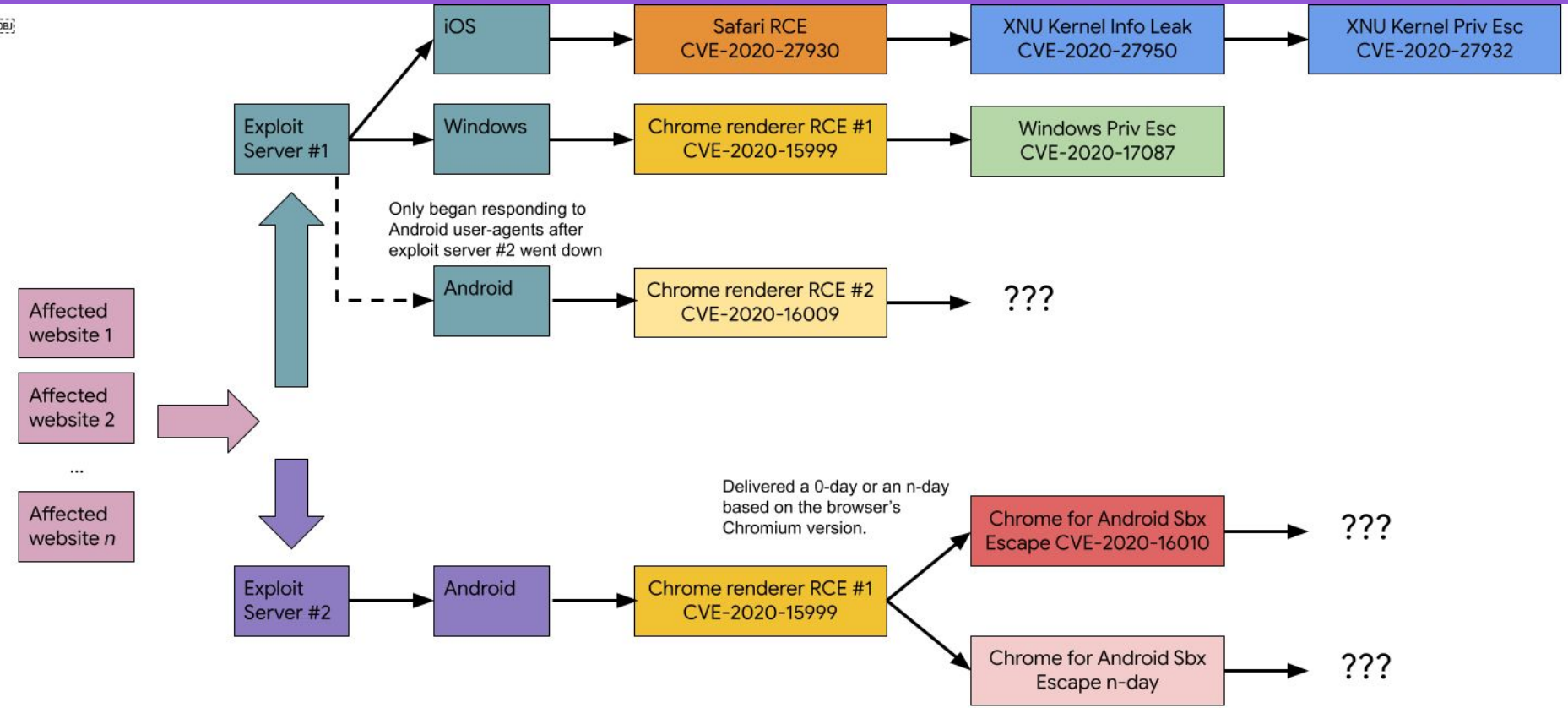
Isolated/renderer process[es]:

```
u:r:isolated_app:s0:c512,c768             u0_i9
com.android.chrome:sandboxed_process0:org.chromium.content.app.SandboxedProcessService0:9
```


@maddiestone
Application exploit
Untrusted app context (Chrome browser process, Messaging app, malicious app, etc.)
Chrome isolated service (Renderer)
System Services
Subset accessible to isolated renderer
GPU
Subset accessible to isolated renderer
Syscalls
Kernel
Drivers
Wifi
NFC
Bluetooth
Baseband

Browser Exploits - Renderer Remote Code Execution

- Chrome uses a multi-process architecture
  - Gaining remote code execution in the renderer process is usually the first step
  - Renderer exploits tend to be usable across
- Common renderer exploit targets
  - V8 (Javascript engine)
  - Blink / DOM
  - Fonts
- Still mostly memory corruption
- Examples:
  - CVE-2020-6418: Incorrect side-effect modelling issue in TurboFan leading to type confusions
  - CVE-2020-15999: FreeType Heap Buffer Overflow in Load_SBit_Png
  - CVE-2021-30551: Type confusion in v8



https://googleprojectzero.blogspot.com/2021/03/in-wild-series-october-2020-0-day.html

CVE-2020-15999

```
Load_SBit_Png( ... ) {
[...]
  png_get_IHDR( png, info,
                &imgWidth, &imgHeight,
                &bitdepth, &color_type, &interlace,
                NULL, NULL ); // *** 1 ***
[...]
  if ( populate_map_and_metrics )   {
    metrics->width  = (FT_UShort)imgWidth; // *** 2 ***
    metrics->height = (FT_UShort)imgHeight;
    map->width      = metrics->width;
    map->rows       = metrics->height;
    map->pixel_mode = FT_PIXEL_MODE_BGRA;
    map->pitch      = (int)( map->width * 4 );
[...]
  if ( populate_map_and_metrics ) {
    /* this doesn't overflow: 0x7FFF * 0x7FFF * 4 < 2^32 */
    FT_ULong  size = map->rows * (FT_ULong)map->pitch; // *** 3 ***
    error = ft_glyphslot_alloc_bitmap( slot, size ); // *** 4 ***
    if ( error )
      goto DestroyExit;  }
[...]
  png_read_image( png, rows ); // *** 5 ***
```

CVE-2020-15999

```
Load_SBit_Png( ... ) {
[...]
  png_get_IHDR( png, info,
                &imgWidth, &imgHeight,
                &bitdepth, &color_type, &interlace,
                NULL, NULL ); // *** 1 ***
[...]
  if ( populate_map_and_metrics )   {
    metrics->width  = (FT_UShort)imgWidth; // *** 2 ***
    metrics->height = (FT_UShort)imgHeight;
    map->width      = metrics->width;
    map->rows       = metrics->height;
    map->pixel_mode = FT_PIXEL_MODE_BGRA;
    map->pitch      = (int)( map->width * 4 );
[...]
  if ( populate_map_and_metrics ) {
    /* this doesn't overflow: 0x7FFF * 0x7FFF * 4 < 2^32 */
    FT_ULong  size = map->rows * (FT_ULong)map->pitch; // *** 3 ***
    error = ft_glyphslot_alloc_bitmap( slot, size ); // *** 4 ***
    if ( error )
      goto DestroyExit;  }
[...]
  png_read_image( png, rows ); // *** 5 ***
```

CVE-2020-15999

```
Load_SBit_Png( ... ) {
[...]
  png_get_IHDR( png, info,
                &imgWidth, &imgHeight,
                &bitdepth, &color_type, &interlace,
                NULL, NULL ); // *** 1 ***
[...]
  if ( populate_map_and_metrics )  {
    metrics->width  = (FT_UShort)imgWidth; // *** 2 ***
    metrics->height = (FT_UShort)imgHeight;
    map->width      = metrics->width;
    map->rows       = metrics->height;
    map->pixel_mode = FT_PIXEL_MODE_BGRA;
    map->pitch      = (int)( map->width * 4 );
[...]
  if ( populate_map_and_metrics ) {
    /* this doesn't overflow: 0x7FFF * 0x7FFF * 4 < 2^32 */
    FT_ULong  size = map->rows * (FT_ULong)map->pitch; // *** 3 ***
    error = ft_glyphslot_alloc_bitmap( slot, size ); // *** 4 ***
    if ( error )
      goto DestroyExit;  }
[...]
  png_read_image( png, rows ); // *** 5 ***
```

CVE-2020-15999

```
Load_SBit_Png( ... ) {
[...]
  png_get_IHDR( png, info,
                &imgWidth, &imgHeight,
                &bitdepth, &color_type, &interlace,
                NULL, NULL ); // *** 1 ***
[...]
  if ( populate_map_and_metrics )  {
    metrics->width  = (FT_UShort)imgWidth; // *** 2 ***
    metrics->height = (FT_UShort)imgHeight;
    map->width      = metrics->width;
    map->rows       = metrics->height;
    map->pixel_mode = FT_PIXEL_MODE_BGRA;
    map->pitch      = (int)( map->width * 4 );
[...]
  if ( populate_map_and_metrics ) {
    /* this doesn't overflow: 0x7FFF * 0x7FFF * 4 < 2^32 */
    FT_ULong  size = map->rows * (FT_ULong)map->pitch; // *** 3 ***
    error = ft_glyphslot_alloc_bitmap( slot, size ); // *** 4 ***
    if ( error )
      goto DestroyExit;  }
[...]
  png_read_image( png, rows ); // *** 5 ***
```

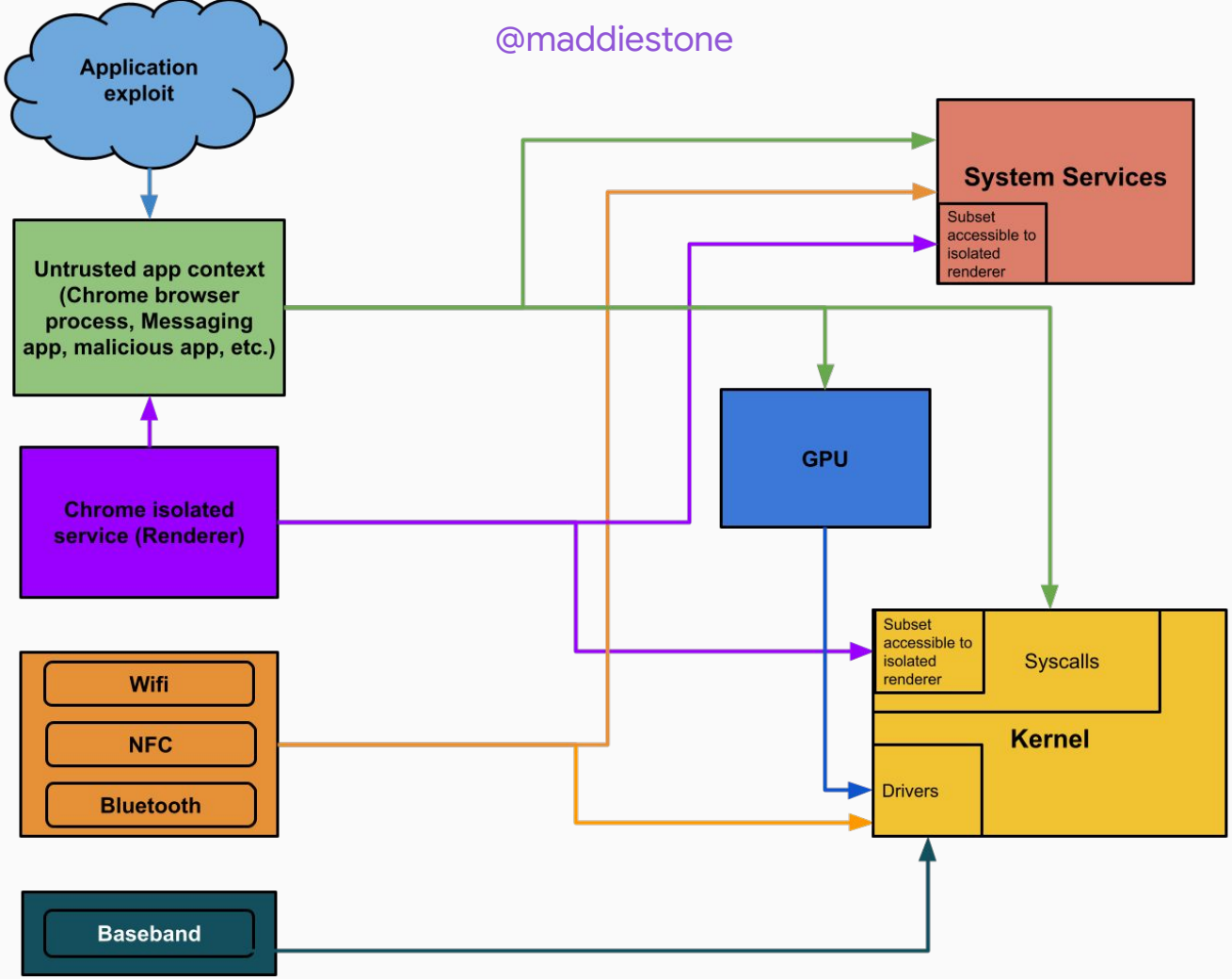
CVE-2020-15999

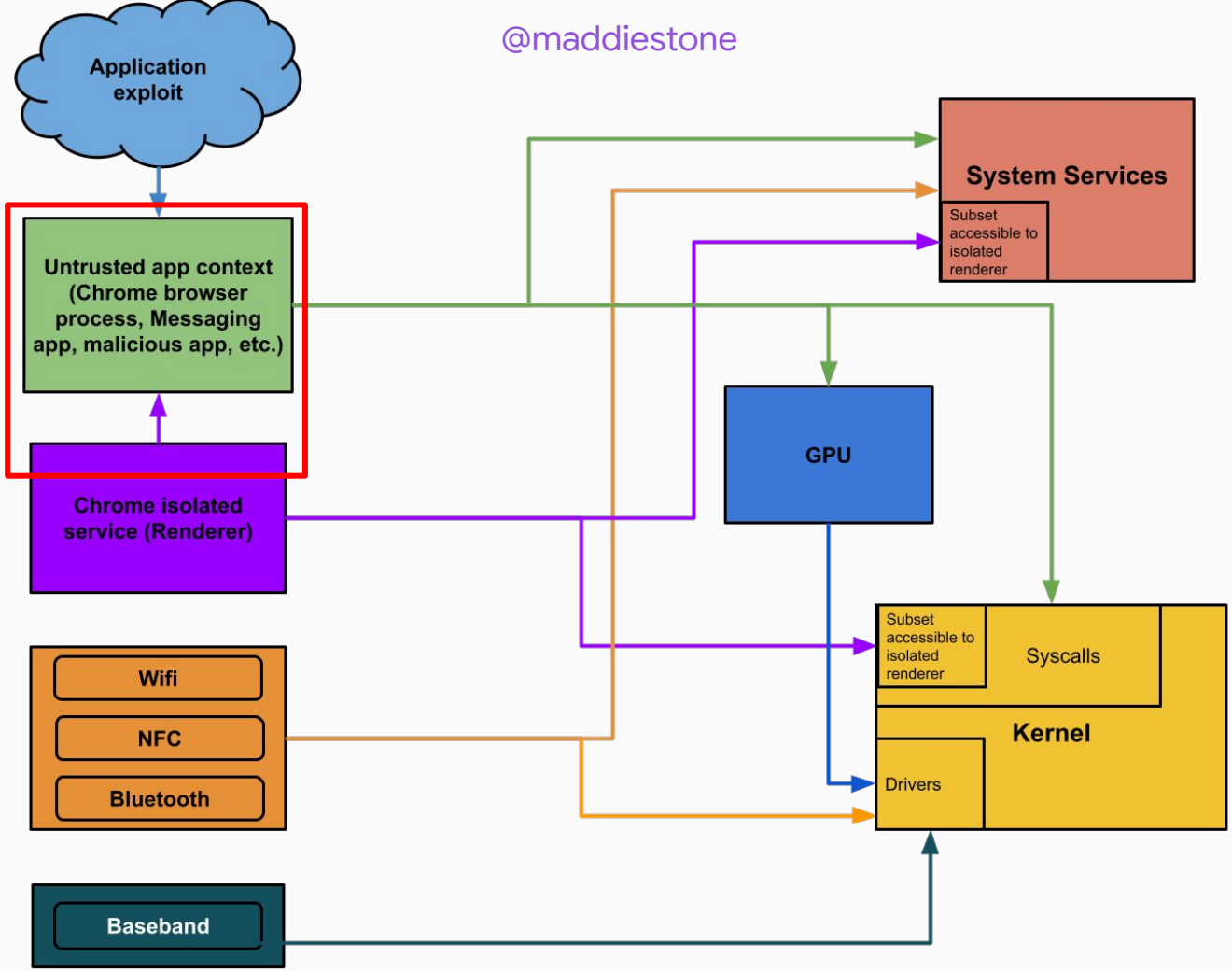
```
Load_SBit_Png( ... ) {
[...]
  png_get_IHDR( png, info,
                &imgWidth, &imgHeight,
                &bitdepth, &color_type, &interlace,
                NULL, NULL ); // *** 1 ***
[...]
  if ( populate_map_and_metrics )  {
    metrics->width  = (FT_UShort)imgWidth; // *** 2 ***
    metrics->height = (FT_UShort)imgHeight;
    map->width      = metrics->width;
    map->rows       = metrics->height;
    map->pixel_mode = FT_PIXEL_MODE_BGRA;
    map->pitch      = (int)( map->width * 4 );
[...]
  if ( populate_map_and_metrics ) {
    /* this doesn't overflow: 0x7FFF * 0x7FFF * 4 < 2^32 */
    FT_ULong  size = map->rows * (FT_ULong)map->pitch; // *** 3 ***
    error = ft_glyphslot_alloc_bitmap( slot, size ); // *** 4 ***
    if ( error )
      goto DestroyExit;  }
[...]
  png_read_image( png, rows ); // *** 5 ***
```

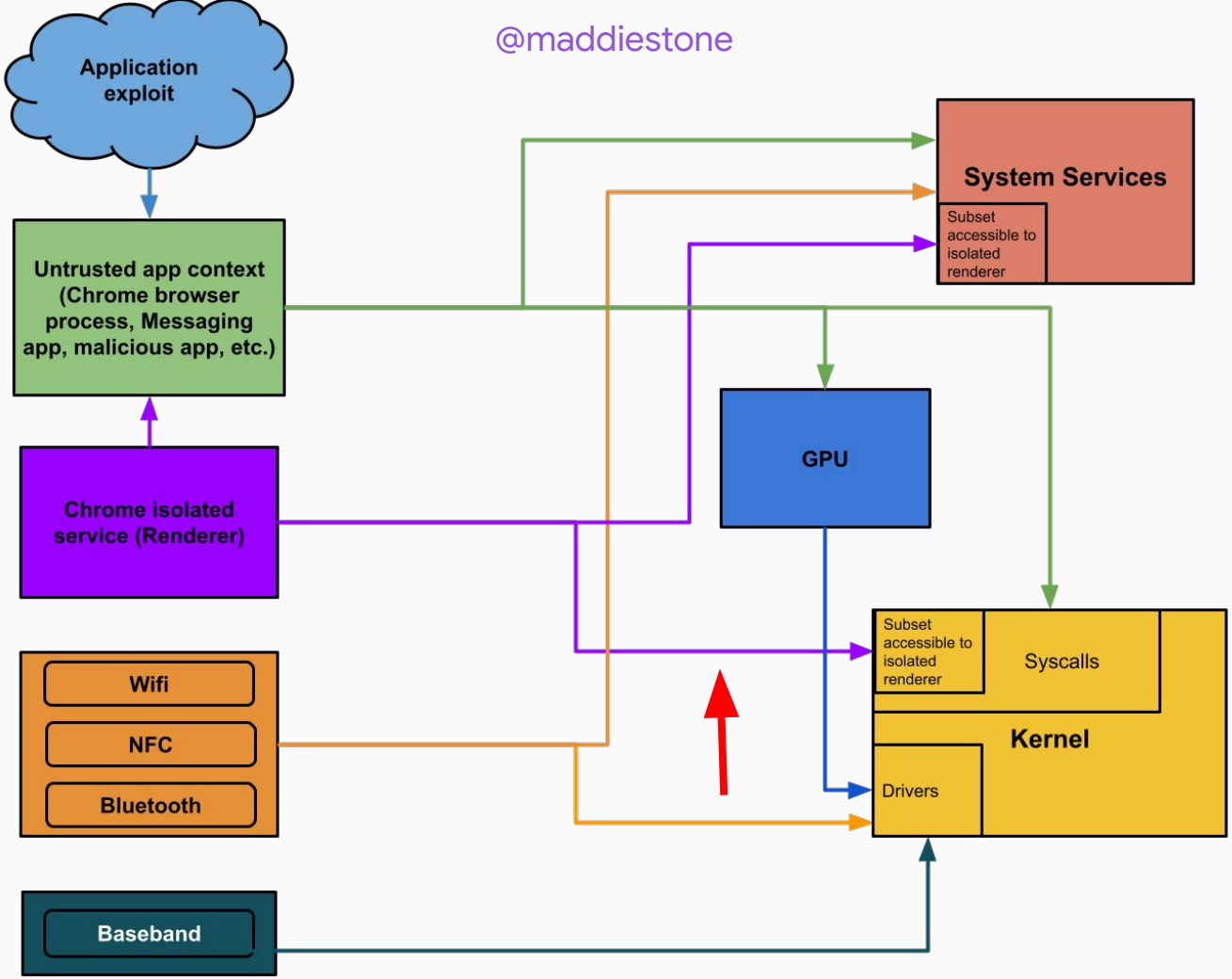
CVE-2020-15999

```
Load_SBit_Png( ... ) {
[...]
  png_get_IHDR( png, info,
                &imgWidth, &imgHeight,
                &bitdepth, &color_type, &interlace,
                NULL, NULL ); // *** 1 ***
[...]
  if ( populate_map_and_metrics )   {
    metrics->width  = (FT_UShort)imgWidth; // *** 2 ***
    metrics->height = (FT_UShort)imgHeight;
    map->width      = metrics->width;
    map->rows       = metrics->height;
    map->pixel_mode = FT_PIXEL_MODE_BGRA;
    map->pitch      = (int)( map->width * 4 );
[...]
  if ( populate_map_and_metrics ) {
    /* this doesn't overflow: 0x7FFF * 0x7FFF * 4 < 2^32 */
    FT_ULong  size = map->rows * (FT_ULong)map->pitch; // *** 3 ***
    error = ft_glyphslot_alloc_bitmap( slot, size ); // *** 4 ***
    if ( error )
      goto DestroyExit;  }
[...]
  png_read_image( png, rows ); // *** 5 ***
```

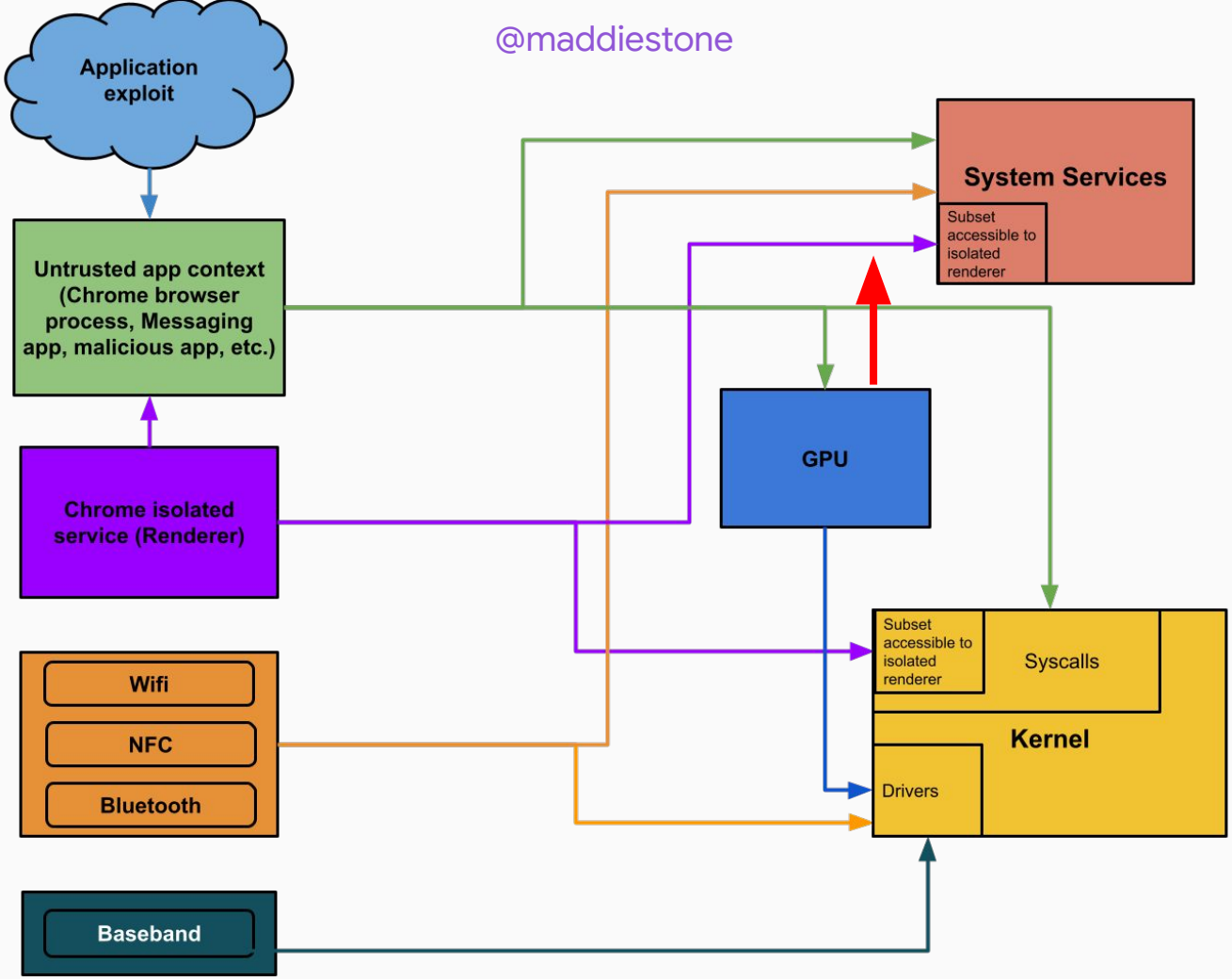
@maddiestone




@maddiestone
Application exploit
Untrusted app context (Chrome browser process, Messaging app, malicious app, etc.)
Chrome isolated service (Renderer)
System Services
Subset accessible to isolated renderer
GPU
Subset accessible to isolated renderer
Syscalls
Kernel
Drivers
Wifi
NFC
Bluetooth
Baseband


@maddiestone
Application exploit
Untrusted app context (Chrome browser process, Messaging app, malicious app, etc.)
Chrome isolated service (Renderer)
Wifi
NFC
Bluetooth
Baseband
System Services
Subset accessible to isolated renderer
GPU
Subset accessible to isolated renderer
Syscalls
Kernel
Drivers

@maddiestone



Browser Exploits - Sandbox Escape

- In this case, “sandbox escape” means escaping the Chrome sandbox, not the application sandbox
- With renderer RCE, the options to escape the sbx are:
  - Renderer (isolated_app) → Browser process (untrusted_app) [Purple to Green]
  - Renderer (isolated_app) → Chrome GPU process [Purple to Green]
  - Renderer (isolated_app) → Kernel (Binder) [Purple to Yellow]
  - Renderer (isolated_app) → System services [Purple to Red]
- Examples:
  - CVE-2020-6572: Chrome MediaCodecAudioDecoder Sandbox escape
    - (and CVE-2019-5870 and CVE-2019-13695)
  - CVE-2020-16010: Sandbox escape to Chrome GPU Process
  - CVE-2020-16045: Sandbox escape via Payment Processing Code

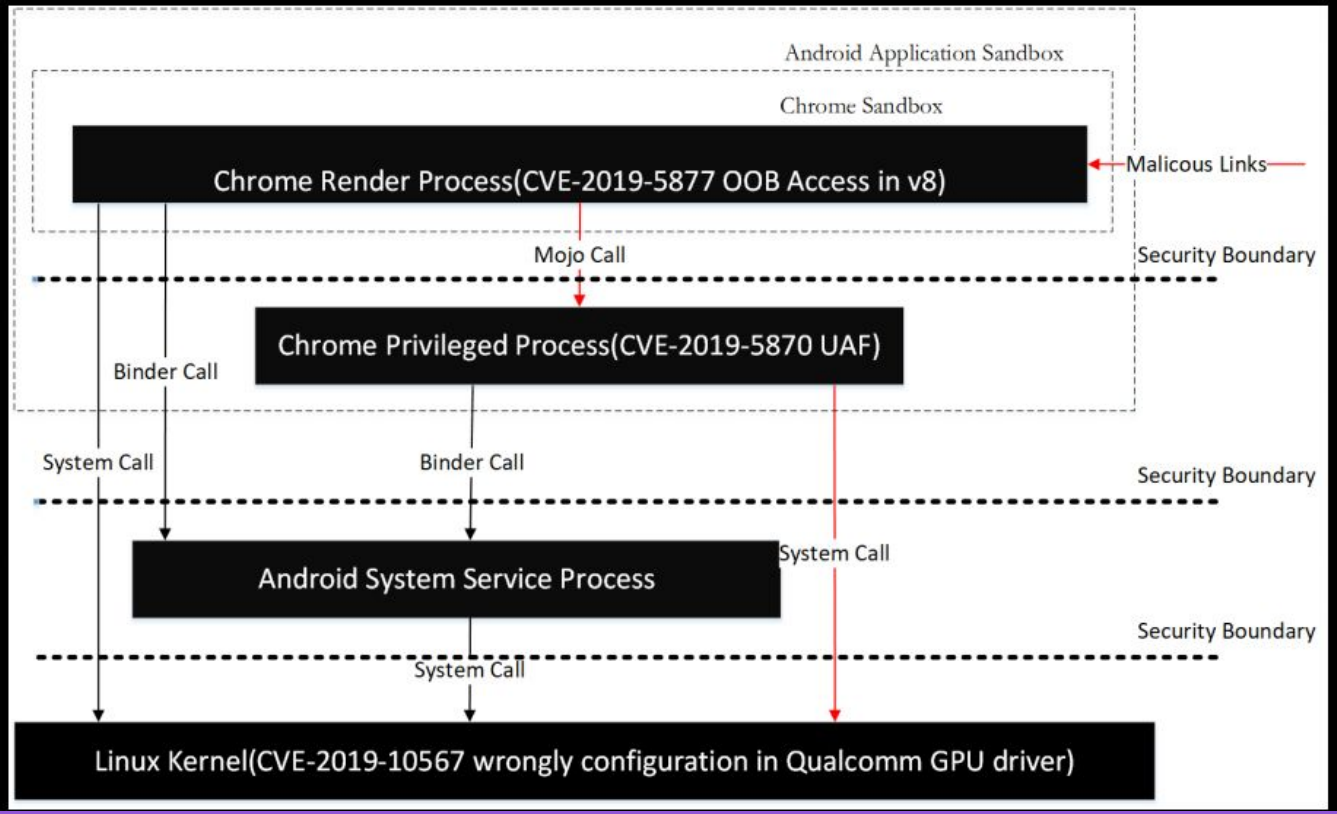
CVE-2020-6572

- Use-after-free in `MediaCodecAudioDecoder::~MediaCodecAudioDecoder()`
  - Android-specific code that uses Android's media decoding APIs to support playback of DRM-protected media on Android
  - Runs in the Chrome GPU process (`privilegedprocess`)
- A `unique_ptr` is assigned to another, going out of scope which means it can be deleted, while at the same time a raw pointer from the originally referenced object isn't updated

Example Chrome for Android Chains

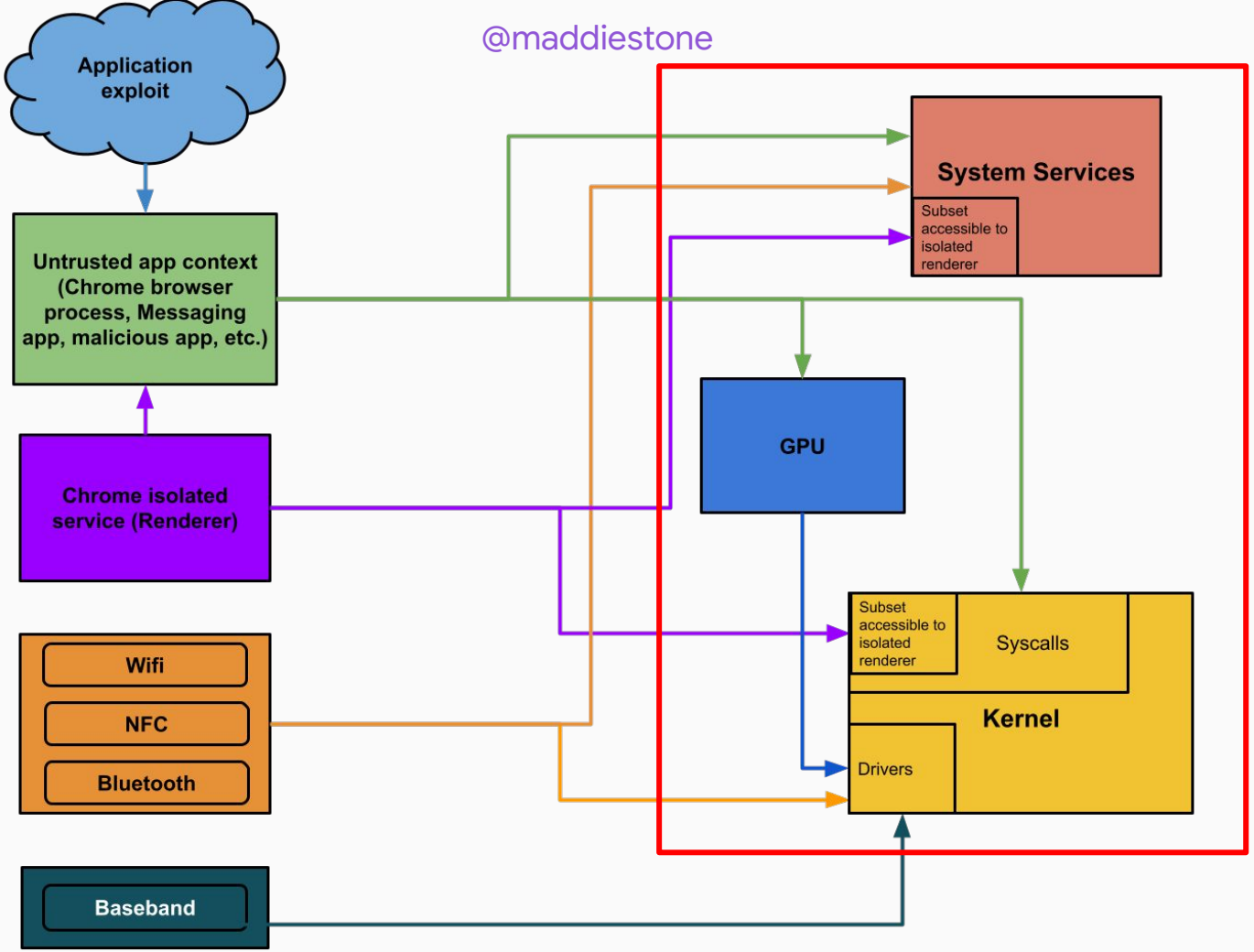
- 3-part blog series by Man Yue Mo
  - CVE-2020-15972 (Chrome webaudio) → CVE-2020-16045 (Chrome payment processing) → CVE-2020-11239 (Qualcomm GPU)
- Oct 2020 in-the-wild exploits
  - CVE-2020-15999 (Freetype) → CVE-2020-16010 (Chrome for Android) → ???
- TiYunZong

The Exploit Chain(TiYunZong)



Blackhat 2020: TiYunZong Pixel full chain exploit [slides, video]

Local Privilege Escalations


@maddiestone
Application exploit
Untrusted app context (Chrome browser process, Messaging app, malicious app, etc.)
Chrome isolated service (Renderer)
Wifi
NFC
Bluetooth
Baseband
System Services
Subset accessible to isolated renderer
GPU
Subset accessible to isolated renderer
Syscalls
Kernel
Drivers

Android LPE Attack Surfaces

From an attacker's perspective, maintaining an Android exploit capability is a question of covering the widest possible range of the Android ecosystem in the most cost-effective way possible.

- ***Tier*: Ubiquitous**

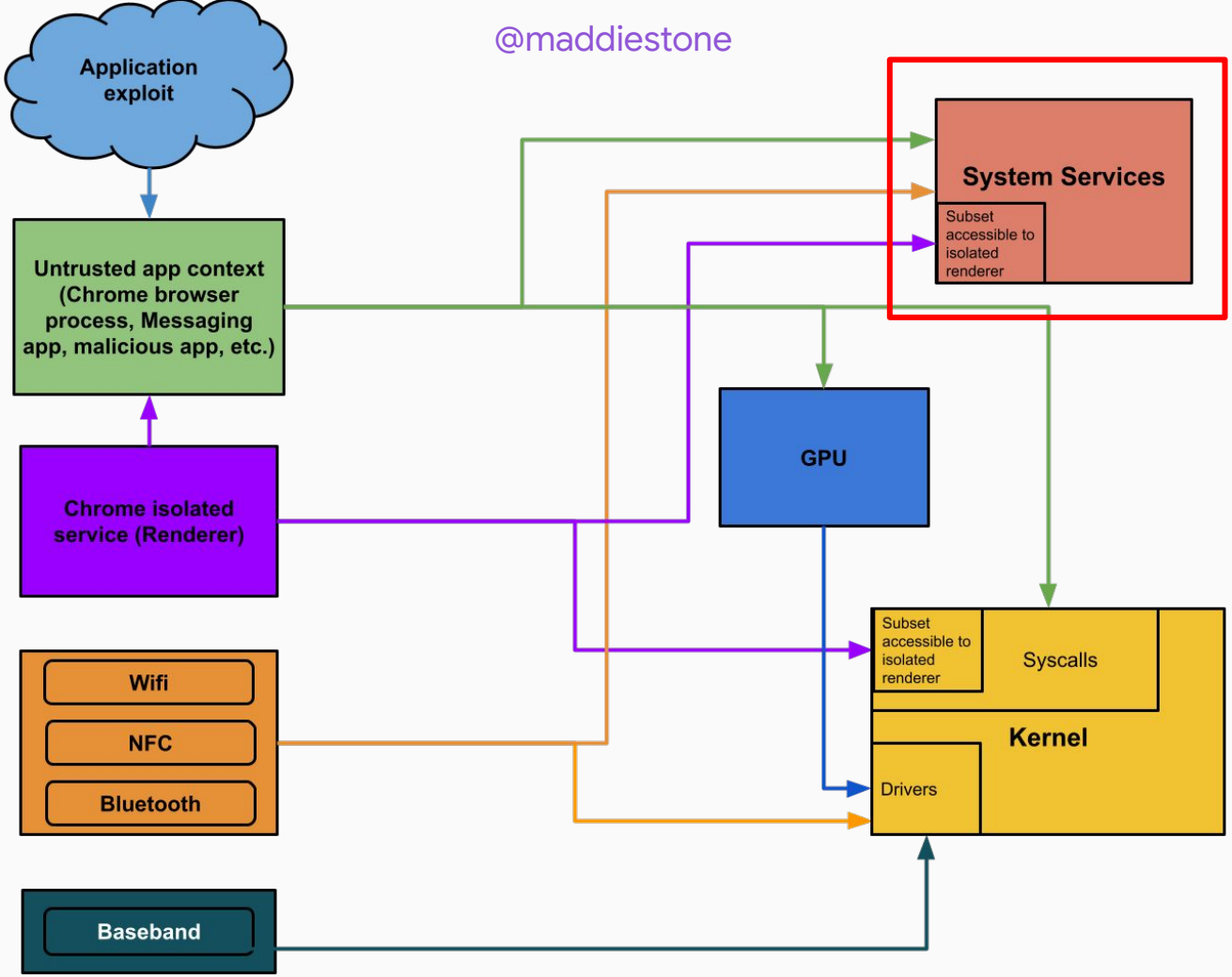
  Description: Issues that affect all devices in the Android ecosystem.
  Example: Core Linux kernel bugs like Dirty COW, or vulnerabilities in standard system services.
- ***Tier*: Chipset**

  Description: Issues that affect a substantial portion of the Android ecosystem, based on which type of hardware is used by various OEM vendors.
  Example: Snapdragon SoC perf counter vulnerability, or Broadcom WiFi firmware stack overflow.
- ***Tier*: Vendor**

  Description: Issues that affect most or all devices from a particular Android OEM vendor
  Example: Samsung kernel driver vulnerabilities
- ***Tier*: Device**

  Description: Issues that affect a particular device model from an Android OEM vendor
  Example: Pixel 4 face unlock "attention aware" vulnerability

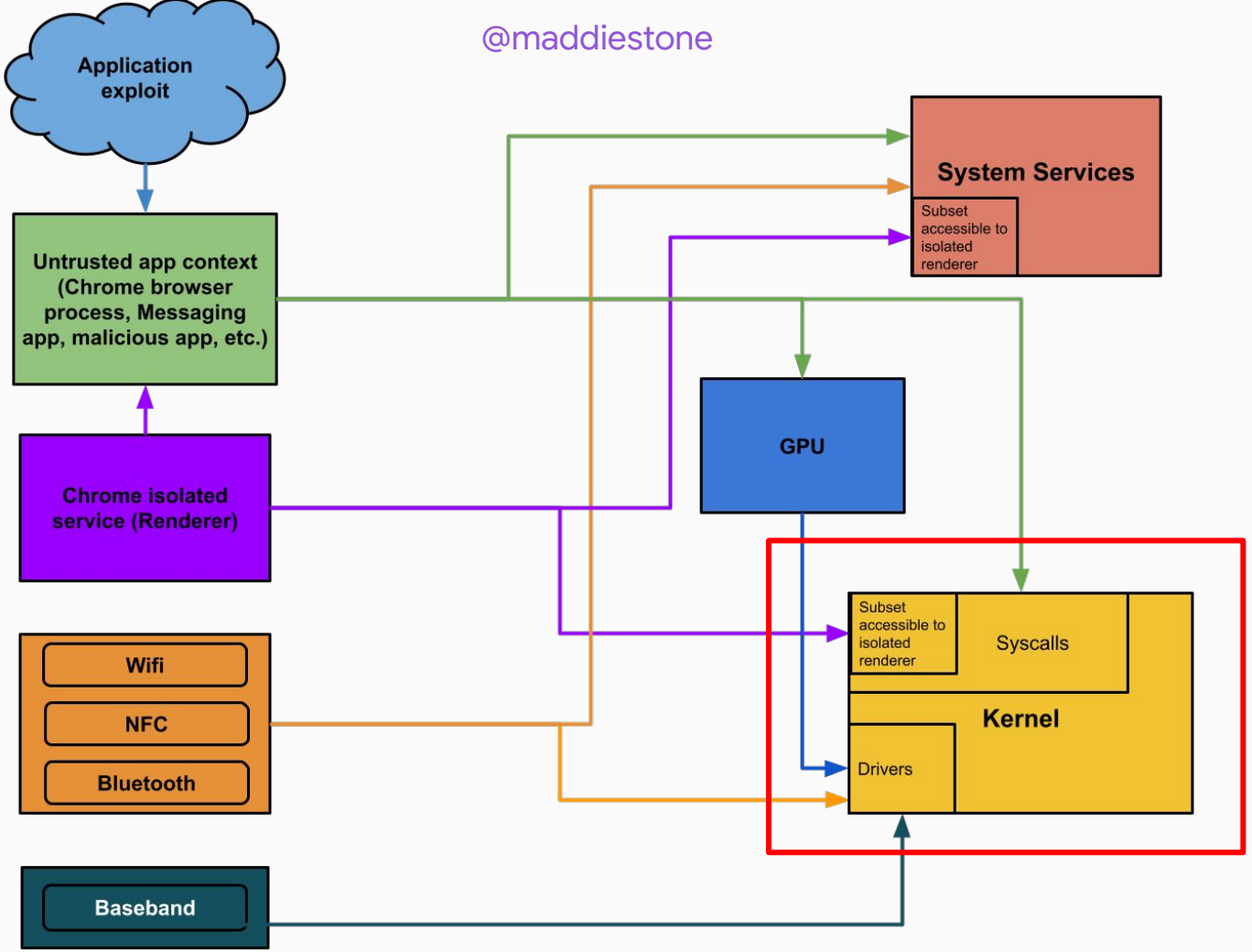
https://googleprojectzero.blogspot.com/2020/09/attacking-qualcomm-adreno-gpu.html


@maddiestone
Application exploit
Untrusted app context (Chrome browser process, Messaging app, malicious app, etc.)
Chrome isolated service (Renderer)
Wifi
NFC
Bluetooth
Baseband
System Services
Subset accessible to isolated renderer
GPU
Subset accessible to isolated renderer
Syscalls
Kernel
Drivers

System Services

- System services are services that run in the system_server process (UID: system)
  - These services are a part of the “Android Framework”: core services for the functionality of the phone written in Java.
    - Include services like Telephony, PackageManager, ActivityManager, PowerManager, etc.
  - https://android.googlesource.com/platform/frameworks/base/+/master/services/java/com/android/server/SystemServer.java
- Example
  - CVE-2018-9411 - https://blog.zimperium.com/cve-2018-9411-new-critical-vulnerability-multiple-high-privileged-android-services/
  - Deserialization Vulnerabilities

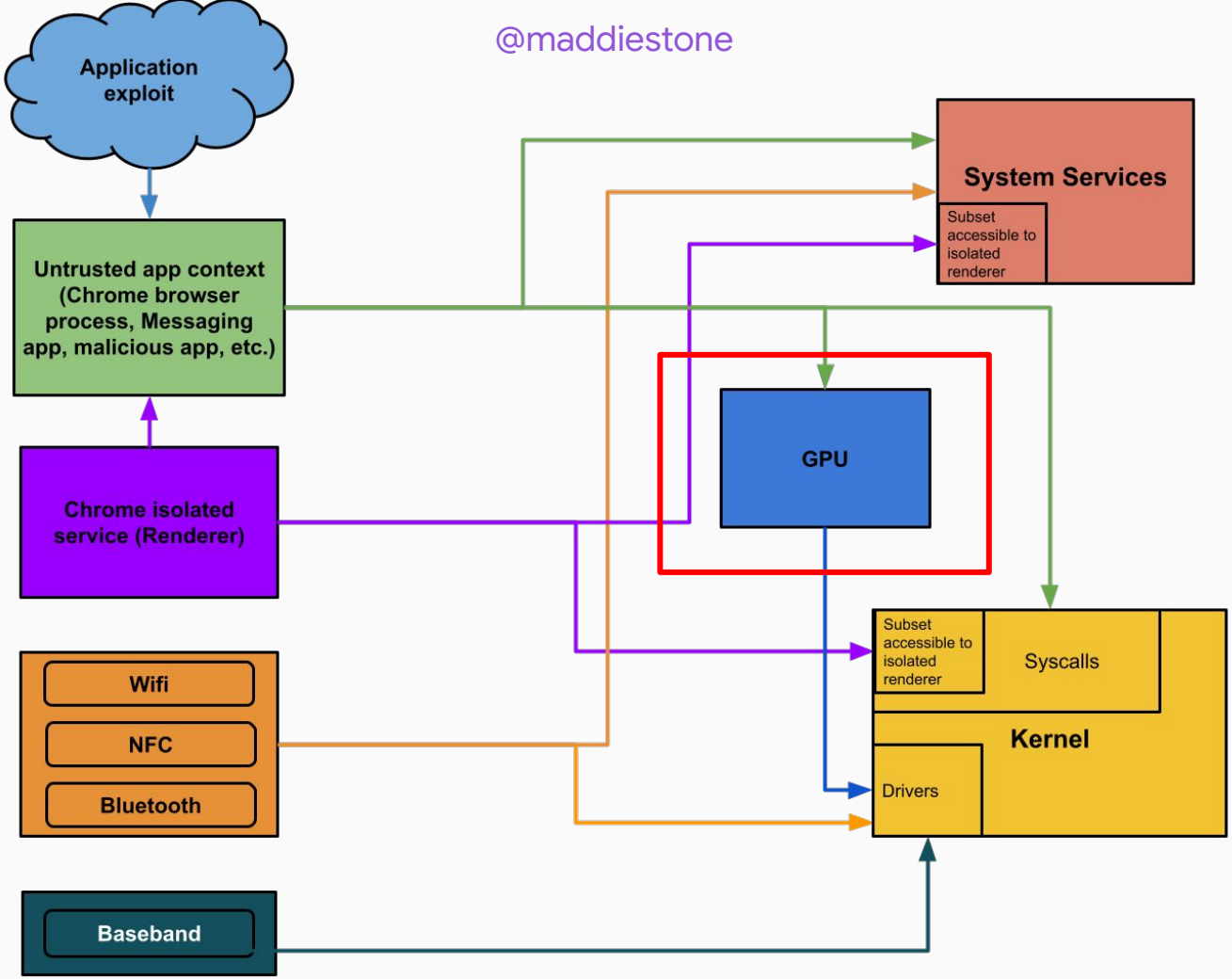
@maddiestone



Kernel Priv Escs

- Survey of n-days used in-the-wild
- Kernel drivers
- Syscalls
- Binder (Inter-Process Communication)
  - Accessible from Chrome renderer (isolated_app)
  - Examples:
    - CVE-2019-2215: “Bad Binder” [blog, video]
    - CVE-2020-0041: Using the same Binder bug to both escape the Chrome sandbox and escalate to root

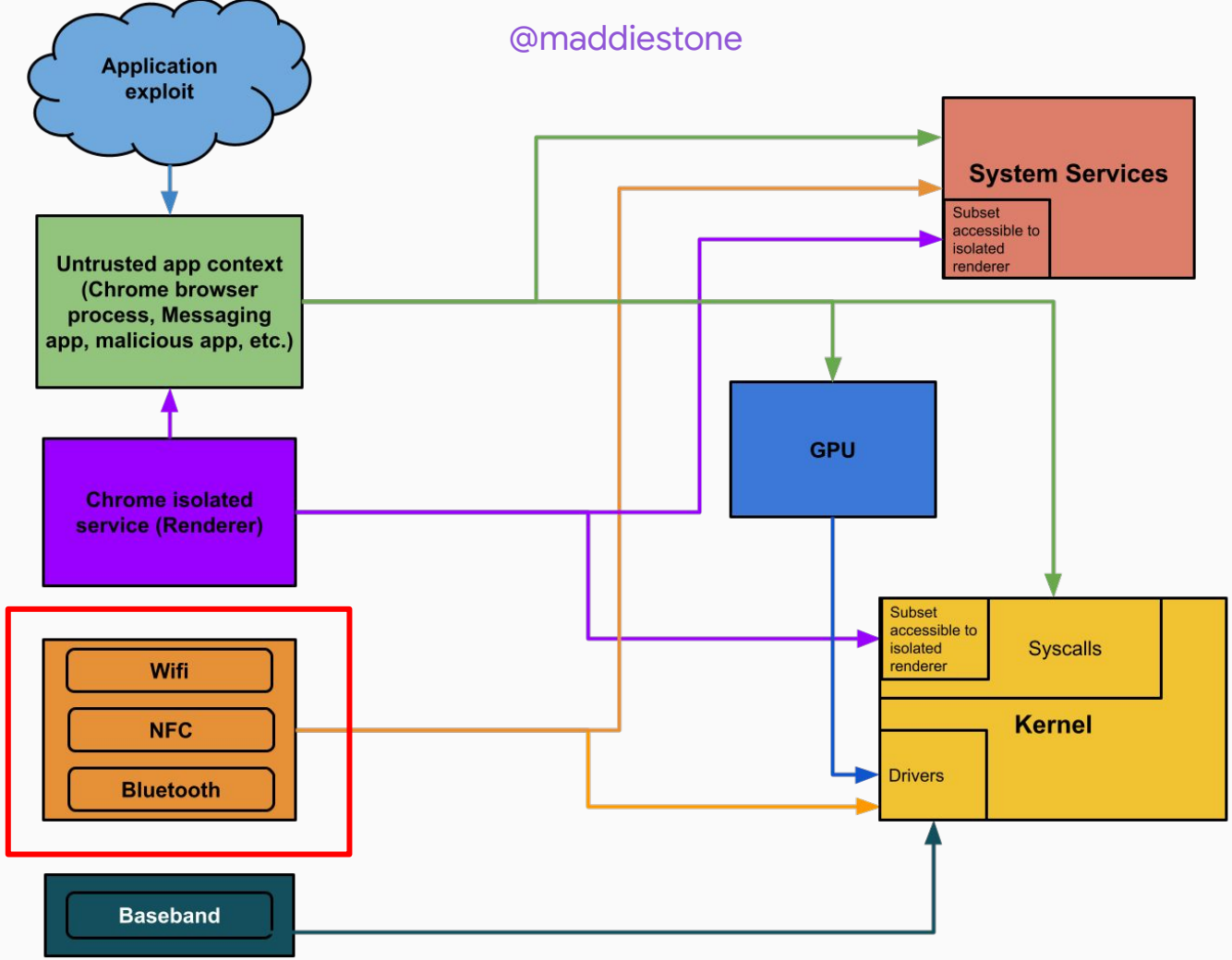
@maddiestone



GPU

- Large number of full-chains we're seeing in the last two years target the GPU for the LPE (5 itw bugs this year)

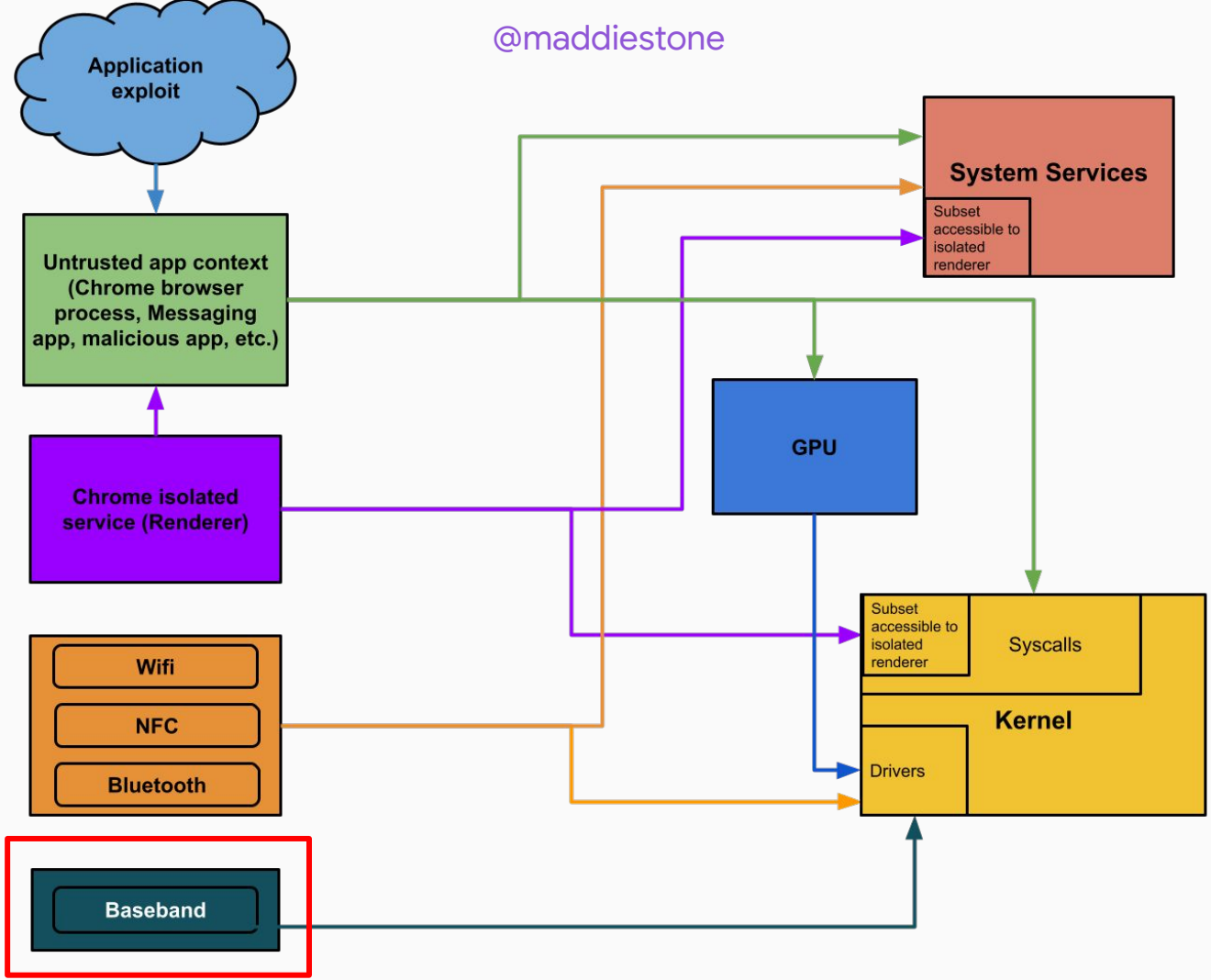
- Qualcomm Adreno & ARM Mali
- Examples:
  - CVE-2021-1905: Qualcomm Adreno GPU memory mapping use-after-free
  - PZ blog post by Ben Hawkes: Attacking the Qualcomm Adreno
  - Blackhat 2020: TiYunZong Pixel full chain exploit [slides, video, whitepaper]
  - Blackhat 2016: GPU-Security-Exposed


@maddiestone
Application exploit
Untrusted app context (Chrome browser process, Messaging app, malicious app, etc.)
Chrome isolated service (Renderer)
Wifi
NFC
Bluetooth
Baseband
System Services
Subset accessible to isolated renderer
GPU
Subset accessible to isolated renderer
Syscalls
Kernel
Drivers

WiFi, NFC, & Bluetooth

- Multiple options for exploitation routes
  - Usermode on component chip
  - Kernel mode on component chip
  - Usermode on application processor
  - Kernel on application processor
- Examples:
  - An-android-8-0-9-0-bluetooth-zero-click-rce

@maddiestone



Baseband

- Flow is usually get code execution on baseband processor and then escape to the application processor, usually via the kernel interface

- Examples:
  - Blackhat 2018: Exploitation of a Modern Smartphone Baseband [paper, video]
  - OffensiveCon 2020: Exploring the MediaTek Baseband
  - Walkthrough of a pwn2own baseband exploit

Common mitigations that have to be bypassed

- SELinux [both mitigation & general access control]
- Privileged Access Never (PAN) [1, 2]
- Knox (Samsung)
  - Real-time Kernel Protection (RKP) [1, 2, 3]
- User Access Override (UAO)
- Huawei Kernel Integrity Protection (HKIP)

Plenty of work to do! All are welcome :)