Shattered Trust: When Replacement Smartphone Components Attack



Omer Shwartz (omershv@post.bgu.ac.il)
Amir Cohen (amir3@post.bgu.ac.il)
Dr. Asaf Shabtai (shabtaia@bgu.ac.il)
Dr. Yossi Oren (yos@bgu.ac.il)

Ben-Gurion University of the Negev, Israel

Overview



- Motivation
- Attack surface
- Vulnerability discovery and demo
- Discussion, conclusions and questions

Motivation



Motivation



Image from: Wikimedia

- Smartphone components often break.
- Many replacements are counterfeit.
- “Street corner phone repair shops” available everywhere.

Research Question

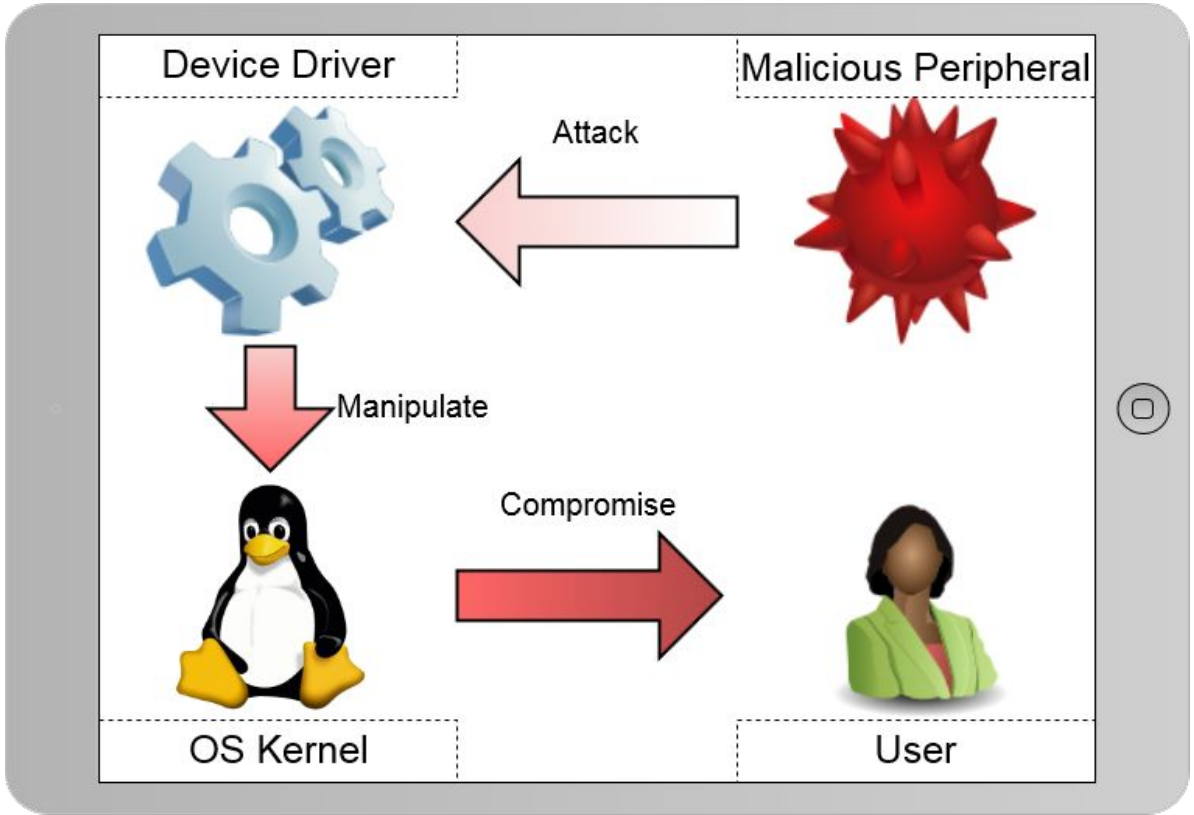


What if a smartphone peripheral was malicious?

Could it attack the stock driver?

Would it affect the user's privacy?

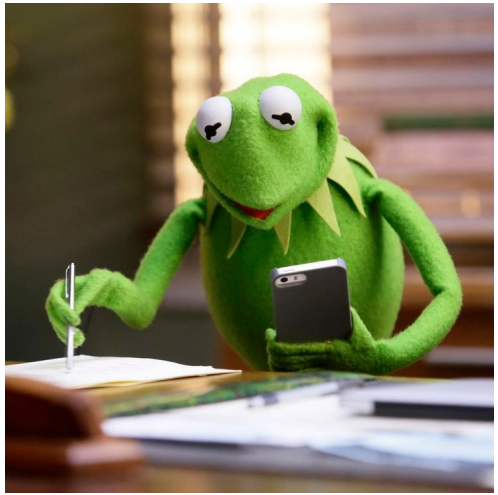
Attack model



Attack surface survey

Shwartz, O., Shitrit, G., Shabtai, A., Oren, Y. (2017) “From smashed screens to smashed stacks: Attacking mobile phones using malicious aftermarket parts”, Workshop on Security for Embedded and Mobile Systems (SEMS 2017).

Attack surface survey



- We started by doing a semi-automated analysis of the source code of 26 android smartphones.
- Drivers were catalogued to vendor and version.
- 89 different driver versions were evaluated.

SONY htc lenovo SAMSUNG LG MOTOROLA HUAWEI

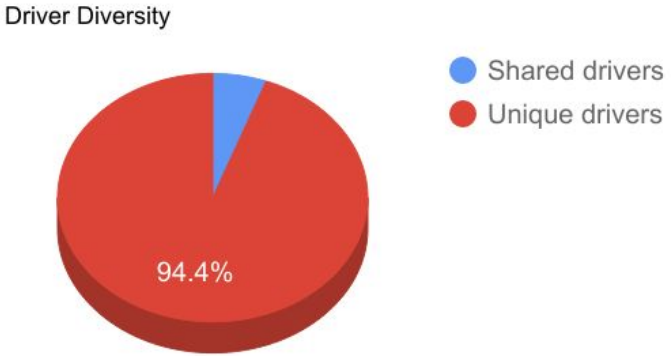
Peripheral diversity

Phone model	Touchscreen	NFC	Charger IC	Battery	Wireless Charger
HTC Desire 626	Syn_t1,Atm_t1	NXP_n1	Sum_c1	Max_b1	---
HTC One A9	Syn_t2	NXP_n2	Sum_c2	HTC_b1	---
HTC One M9+	$Syn_t3$$Max_t1$	NXP_n3	Sum_c3	HTC_b2	---
Huawei Honor 5X	Syn_t4	NXP_n4	Hua_c1	TI_b1	---
Huawei Nexus 6P	Syn_t5	*NXP_n5	Qua_c1	Qua_b1	---
Lenovo K3 Note	Med_t1	---	Med_c1	Med_b1	---
Lenovo K5 Note	Med_t2	Qua_n1	Sum_c4	Med_b2	---
Lenovo Vibe K4 Note	Med_t3	Med_n1	Tex_c1	Med_b3	---
LG Nexus 4 E960	Syn_t6	Bro_n1	Int_c1	TI_b2	TI_w1
LG Nexus 5	Syn_t7	Bro_n2	Sum_c5	TI_b3	TI_w2
LG G3	Atm_t2	NXP_n6	Sum_c6	*TI_b4,Max_b2	TI_w3
LG G4	Syn_t8	NXP_n7	Sum_c7	Max_b3	---
LG Nexus 5X	Syn_t9	*NXP_n5	Qua_c2	Qua_b2	---
Motorola Nexus 6	Atm_t3	Bro_n3	Sum_c8	TI_b5,Max_b4	Mot_w1
Samsung Galaxy S5	Atm_t4	NXP_n8,Bro_n4	Sum_c9	TI_b6,Max_b5	---
Samsung Galaxy S6	*STM_t1	Sam_n1	*Max_c1	*Max_b6	Tex_w4
Samsung Galaxy S6 edge	*STM_t1	Sam_n2	*Max_c1	*Max_b6	Tex_w5
Samsung Galaxy S6 edge+	STM_t2	Sam_n3	*Max_c1	*Max_b6	Tex_w6
Samsung Galaxy S7	Sam_t1	NXP_n9,Sam_n4	Max_c2	Max_b7	IDT_w1
Samsung Galaxy S7 edge	Sam_t2	NXP_n10,Sam_n5	Max_c3	Max_b8	IDT_w2
Sony Xperia Z2	Max_t2	NXP_n11	Qua_c3	*TI_b4,Max_b9	---
Sony Xperia M4 Aqua	Syn_t10	NXP_n12	Qua_c4	Sum_b1	---
Sony Xperia M5	Syn_t11	NXP_n13	Med_c2	Med_b4	---
Sony Xperia Z5	Syn_t12,Atm_t5	NXP_n14	Sum_c10	TI_b7	---
Sony Xperia E5	Med_t4	Med_n2	Med_c3	Med_b5	---
Sony Xperia XA	Med_t5	Med_n3	Qno_c1	Med_b6	---

Attack surface survey results

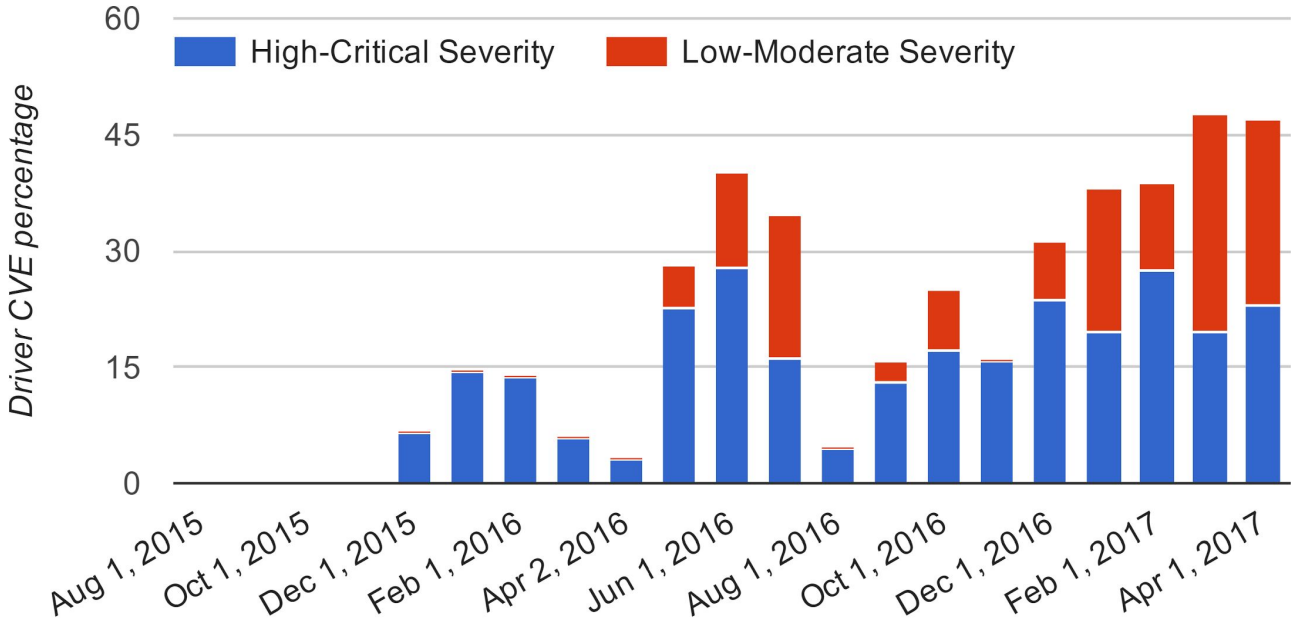


- Only three drivers were used in two phone models.
- Only two drivers were used on three or more phone models.
- Most of the drivers were unique to their respective device.



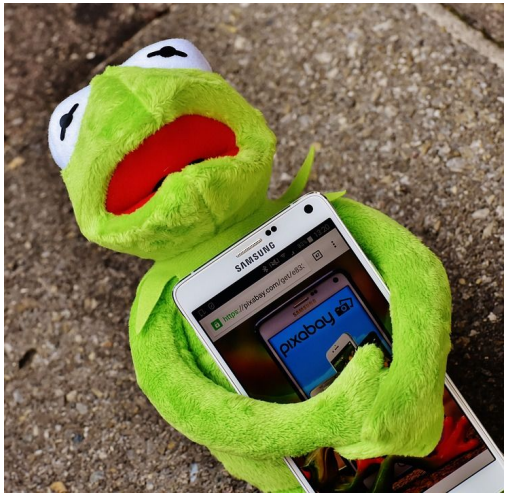
Attack surface survey results

Percentage of Driver related android CVEs is on the rise.

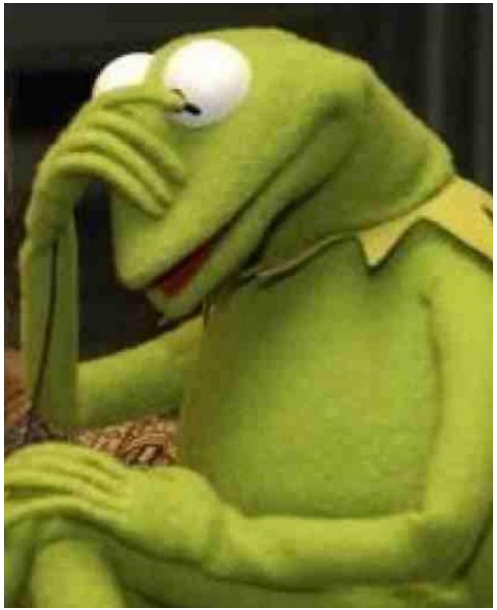


Attack surface survey insights

- A codebase that diverse is ought to contain bugs and vulnerabilities.
- Bugs are easy to find (more ahead…)



Bug Hunting



- We started looking for device driver bugs that may be exploited by the component.
- In less than two hours (!), we found exploitable buffer and heap overflows in a touchscreen module.
- Further analysis revealed dozens more potential issues in multiple drivers.
- The issues found were reported, acknowledged (CVE-2017-0650) and patched by Google.

Actual Exploitation

Remote attacker

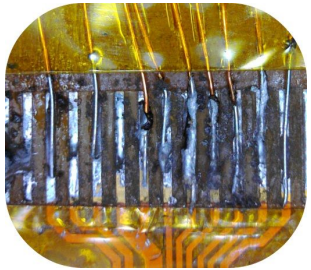
Malicious chip debug output

Phone UART output



Workbench







CVE-2017-0650

```
static void synaptics_rmi4_sensor_report(struct synaptics_rmi4_data *rmi4_data,
        bool report)
{
    int retval;
    unsigned char data[MAX_INTR_REGISTERS + 1];
    unsigned char *intr = &data[1];
    struct synaptics_rmi4_f01_device_status status;
    struct synaptics_rmi4_fn *fhandler;
    struct synaptics_rmi4_exp_fhandler *exp_fhandler;
    struct synaptics_rmi4_device_info *rmi;

    rmi = &(rmi4_data->rmi4_mod_info);

    /*
     * Get interrupt status information from F01 Data1 register to
     * determine the source(s) that are flagging the interrupt.
     */
    retval = synaptics_rmi4_reg_read(rmi4_data,
            rmi4_data->f01_data_base_addr,
            data,
            rmi4_data->num_of_intr_regs + 1);
```

CVE-2017-0650

```
static int synaptics_rmi4_query_device(struct synaptics_rmi4_data *rmi4_data)
{
```

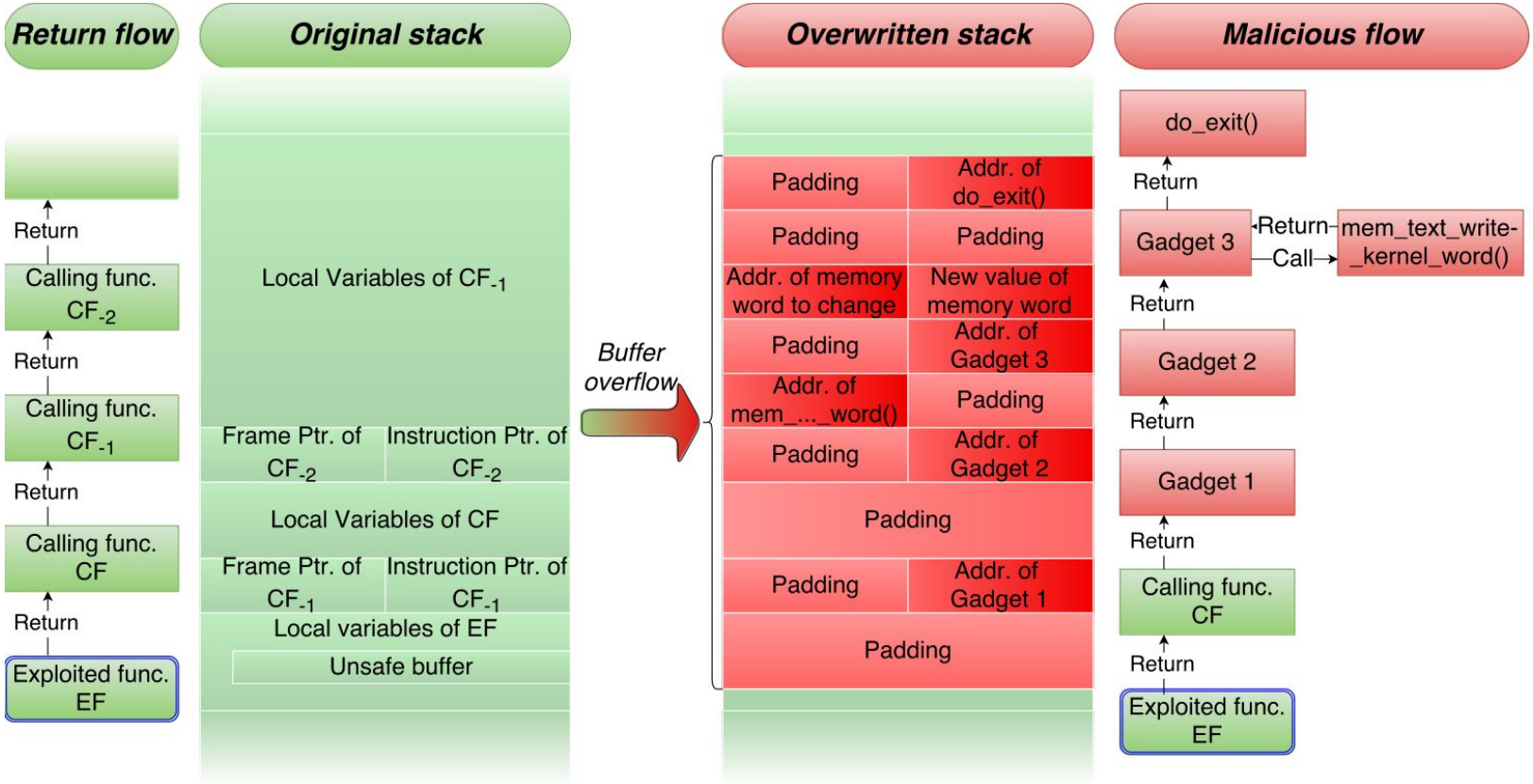
Get data from device...

....

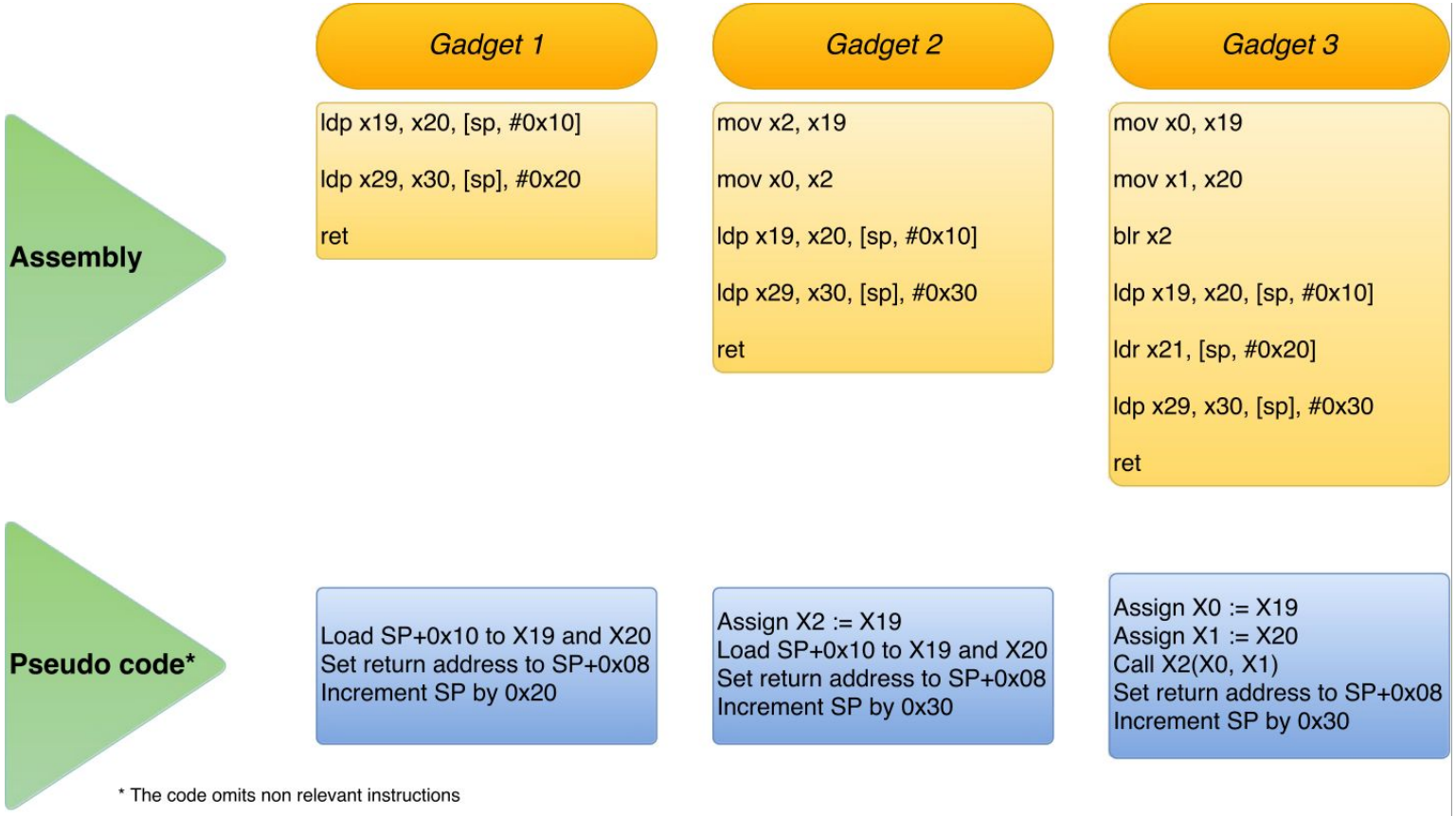
```
  if (!list_empty(&rmi->support_fn_list)) {
      list_for_each_entry(fhandler, &rmi->support_fn_list, link) {
          if (fhandler->num_of_data_sources) {
             rmi4_data->intr_mask[fhandler->intr_reg_num] |=
                     fhandler->intr_mask;
          }
```

```
unsigned char no_sleep_setting;
unsigned char intr_mask[MAX_INTR_REGISTERS];
unsigned char *button_txrx_mapping;
unsigned short num_of_intr_regs;
unsigned short f01_query_base_addr;
unsigned short f01_cmd_base_addr;
unsigned short f01_ctrl_base_addr;
unsigned short f01_data_base_addr;
unsigned int firmware_id;
unsigned int config_id;
int irq;
```

ARM64 ROP Attack



ARM64 ROP Attack



Vulnerabilities

- Vulnerabilities such as CVE-2017-0650 are easy to find!
- Another vulnerability was found in a different touchscreen driver by another manufacturer. This vulnerability was proved in a POC level. (Pending Responsible Disclosure)

Touch Logging & Touch Injection

- In addition to exploiting the kernel, the touchscreen can abuse its known capabilities.
- The touchscreen or a component on the touchscreen bus can record user touches.
- Injection of touch events can also be done without any user interaction.

Proofs of Concept



- Malicious Software Installation
  https://youtu.be/83VMVrcEOCM
- Take Picture and Send Via Email
  https://youtu.be/WS4NChPjaaY
- Replace URL with Phishing URL
  https://youtu.be/XZujd42eYek
- Log and Exfiltrate Screen Unlock Pattern
  https://youtu.be/fY58zoadqMA
- Complete Phone Compromise
  https://youtu.be/sDfD5fJfiNc



Discussion



- In smartphones, device drivers are very diverse, leading to many potential vulnerabilities.
- The device drivers are trusted by the kernel, and the peripherals are trusted by the drivers.
- We showed how a peripheral may exploit that trust.
- Detection of attacks may be impossible with current tools and design.

Conclusions

- Attacks of this sort are practical and may be highly effective.
- Consumers have no tools to help them detect such attacks.
- Countermeasures, such as a physical interface firewall may prove effective.
- There is much more left to explore in the field of malicious hardware components.





Thank you

- Omer Shwartz (omershv@post.bgu.ac.il)
- Amir Cohen (amir3@post.bgu.ac.il)
- Dr. Asaf Shabtai (shabtaia@bgu.ac.il)
- Dr. Yossi Oren (yos@bgu.ac.il)

