Digital Security

black hat
MOBILE SECURITY SUMMIT
LONDON 2015

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Ivan ‘Steph’ Yushkevich, Alexander ‘dark_k3y’ Bolshev

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

; cat /dev/user

Digital Security

- Ivan ‘Steph’ Yushkevich:
  - Security Auditor @ Digital Security
  - Role: Mobile security guy
- Alexander ‘dark_k3y’ Bolshev
  - Security Researcher @ Digital Security, Ph.D., Assistant Professor @ SPb ETU.
  - Role: Fuzzing && SCADA security guy

Agenda

- Very quick ICS 101
- Types of mobile ICS applications
- Example vulnerabilities
- Testing methodology
- Attack examples
- Conclusion

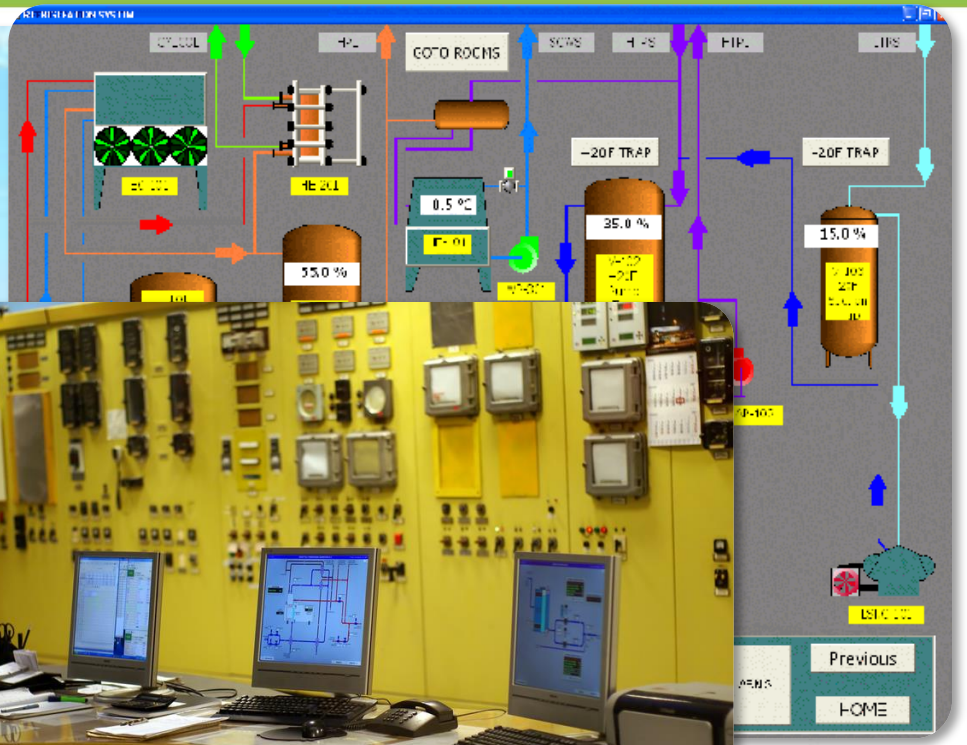
ICS 101

What is ICS

- ICS stands for Industrial Control System
- Today, ICS infrastructures are commonly used in every factory and even in your house, too!
- ICS collects data from remote stations (also called field devices), processes them and uses automated algorithms or operator-driven supervisory to create commands to be sent back

ICS

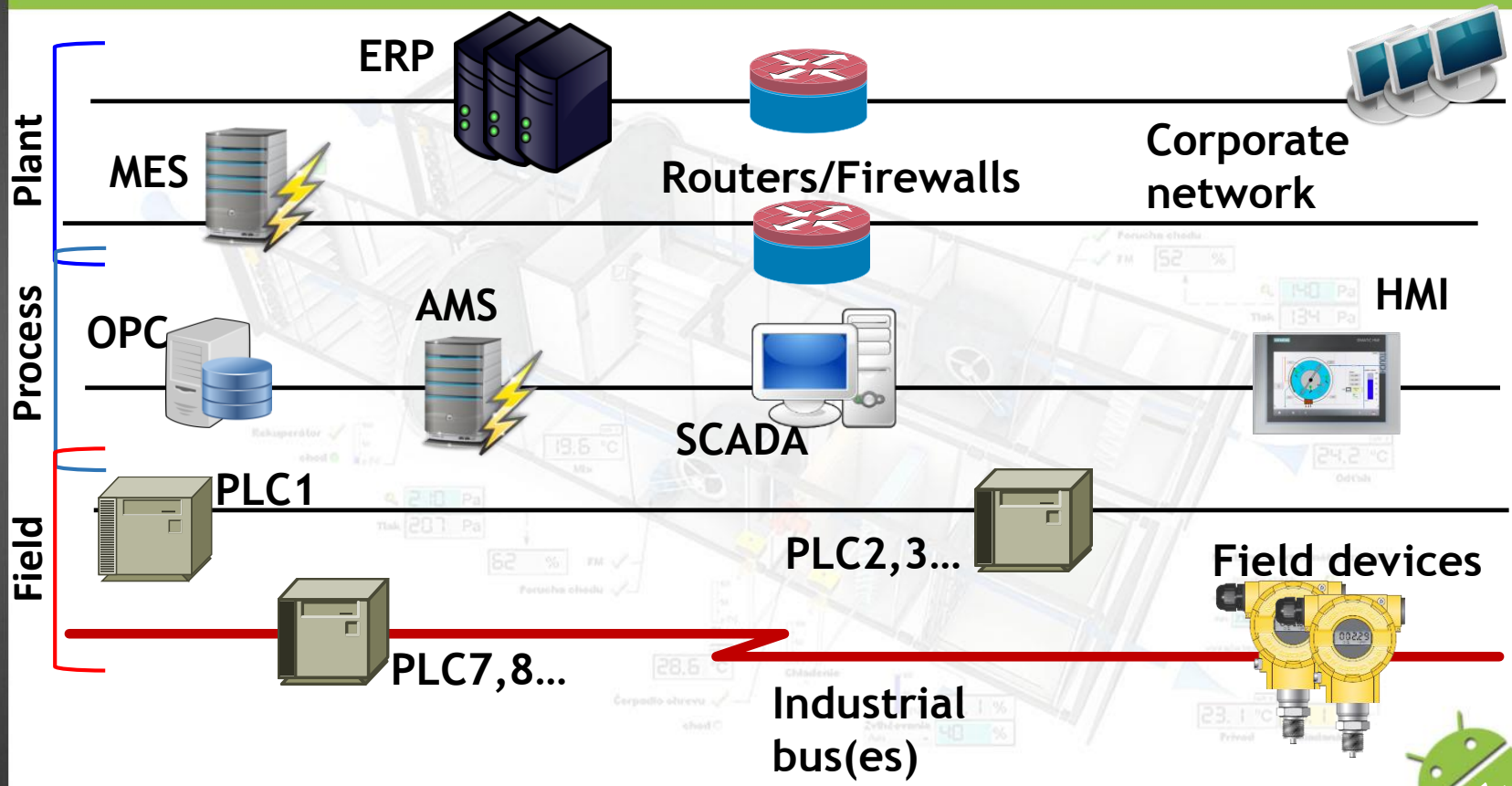
SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM



http://sub0day.com/wp-content/uploads/2015/01/asdC2121.jpg

http://www.paceindustrial.com/uploads/images/Controls/Industrial_System_Page.gif

Typical ICS Infrastructure



ICS 101 terms

- Transmitters/RTUs – works with real world objects and parameters
- PLC (Programmable Logic Controllers) -- digital system used for automation of typically industrial electromechanical processes
- SCADA – systems operating with coded signals over communication channels so as to provide control of remote equipment
- OPC – Open Platform Communications
- HMI – Human-machine interfaces
- MES – Manufacturing executioning system

Mobile ICS applications

Mobile ICS Apps classification

Control-room applications

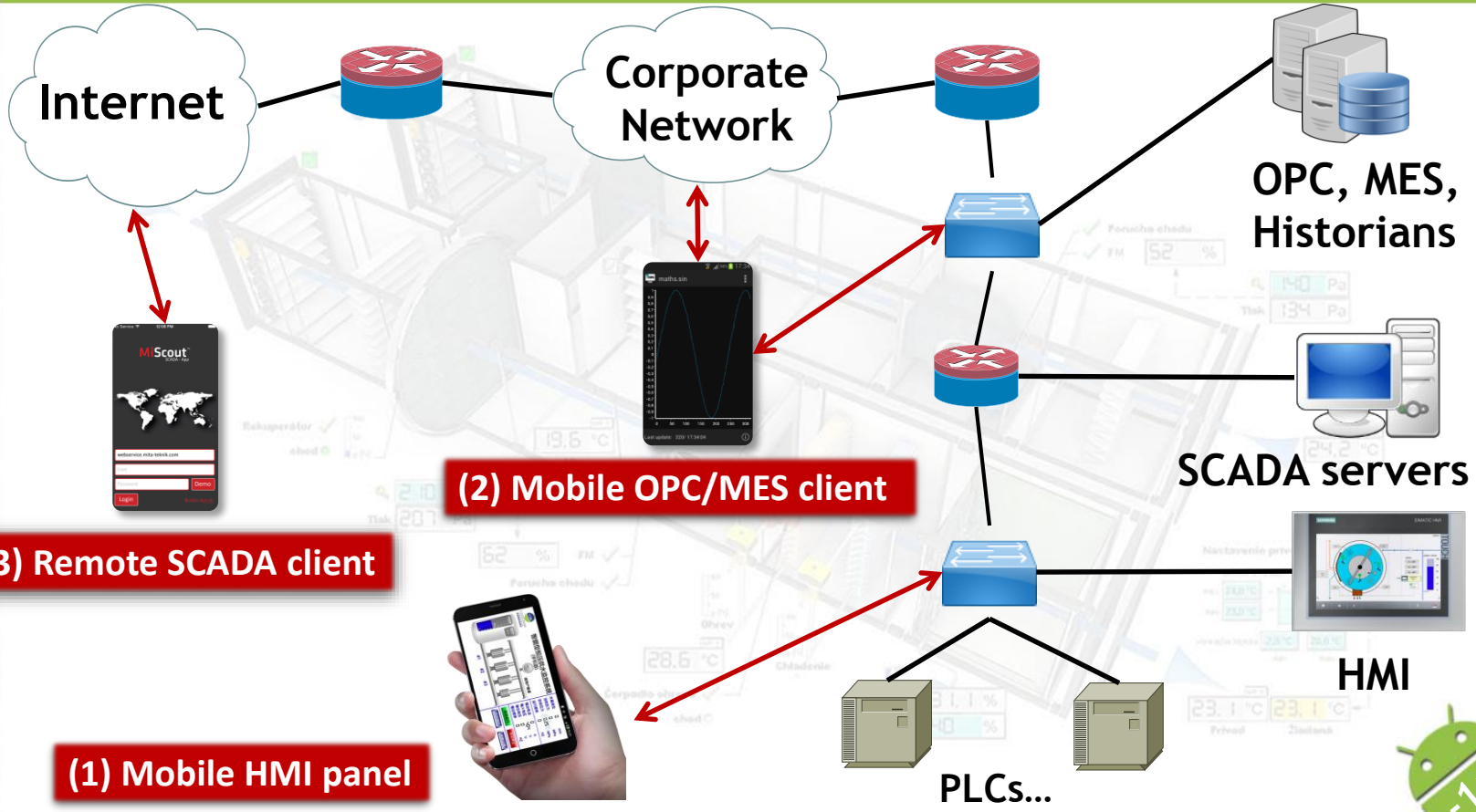
- PLC configuration/interaction app
- SCADA client
- Mobile HMI panel

MES-/OPC-/Historian- clients

Remote SCADA clients

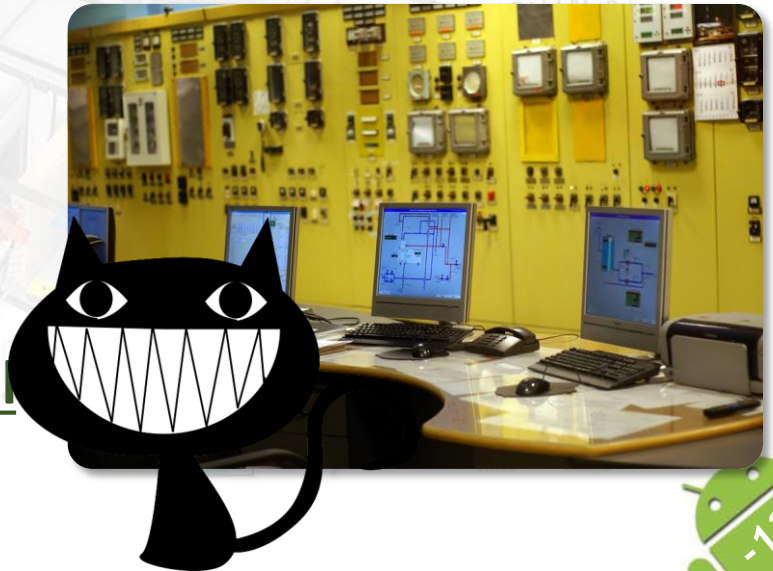
Mobile Apps place in the ICS

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM



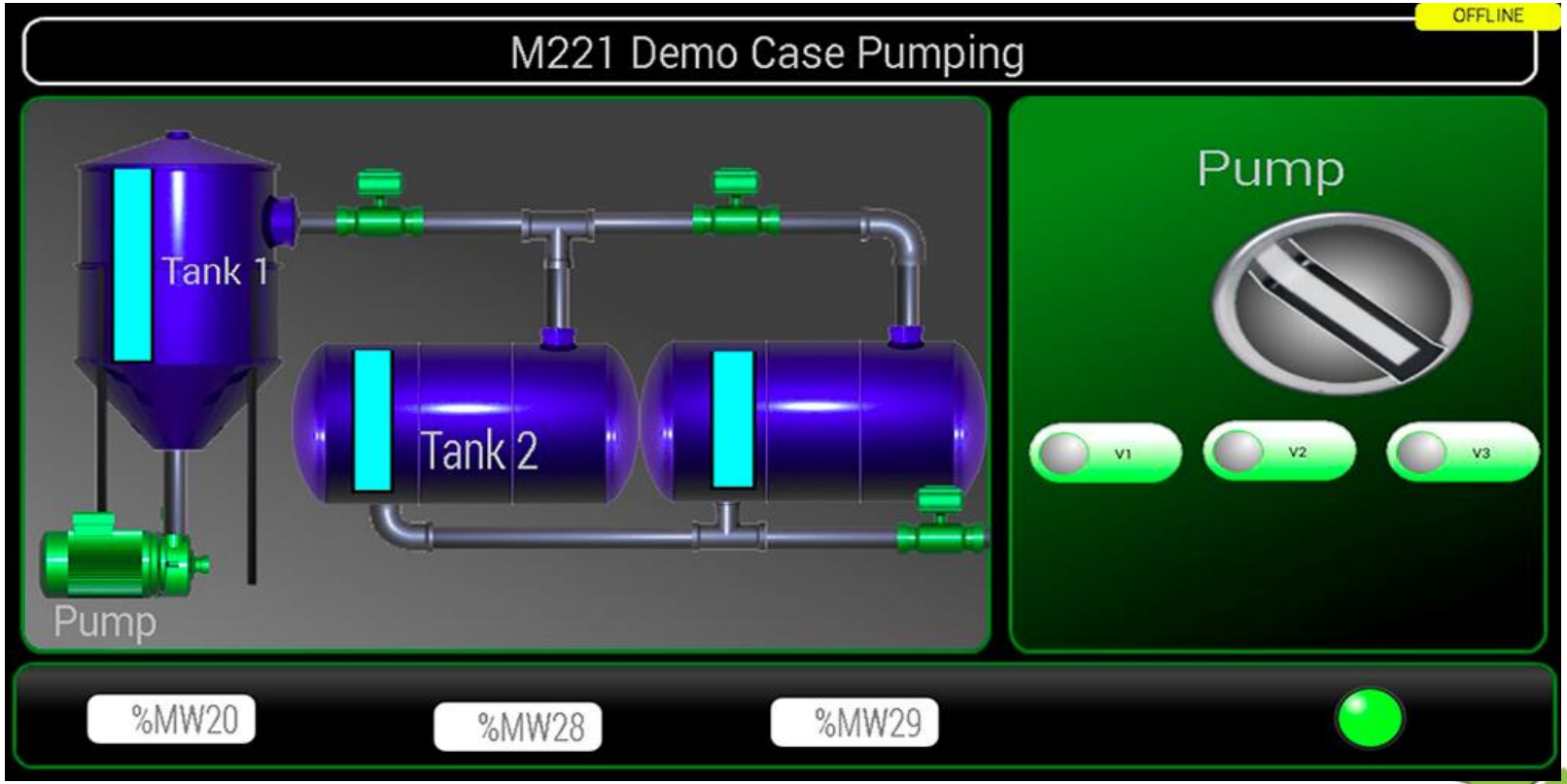
Control-room applications

- Direct configuring/monitoring/supervising industrial process and/or its components
- Several types:
  - PLC configuration/interaction app
  - SCADA client
  - Mobile HMI panel
- These applications reside inside "safe" (at least firewalled and separated) control room network
- **Full or partial "local" control of industrial process**



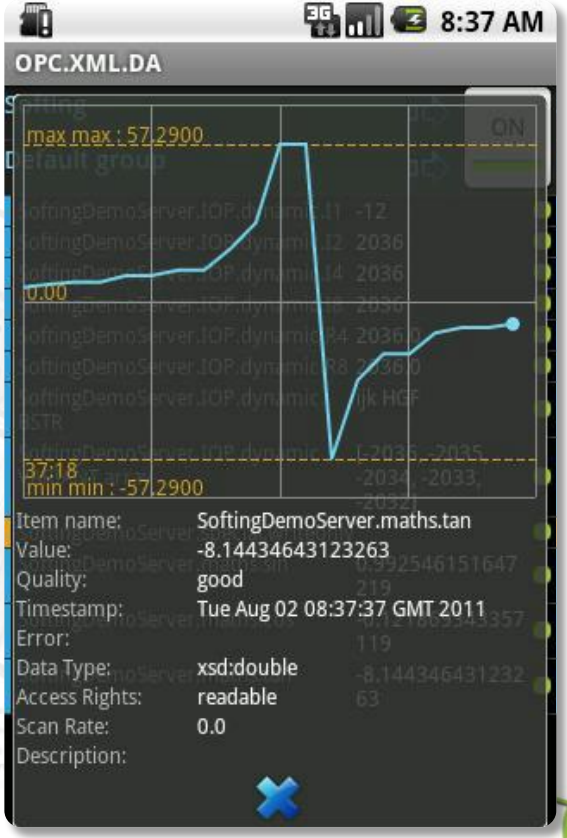
SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Mobile HMI system



OPC-/MES-/Historian- clients

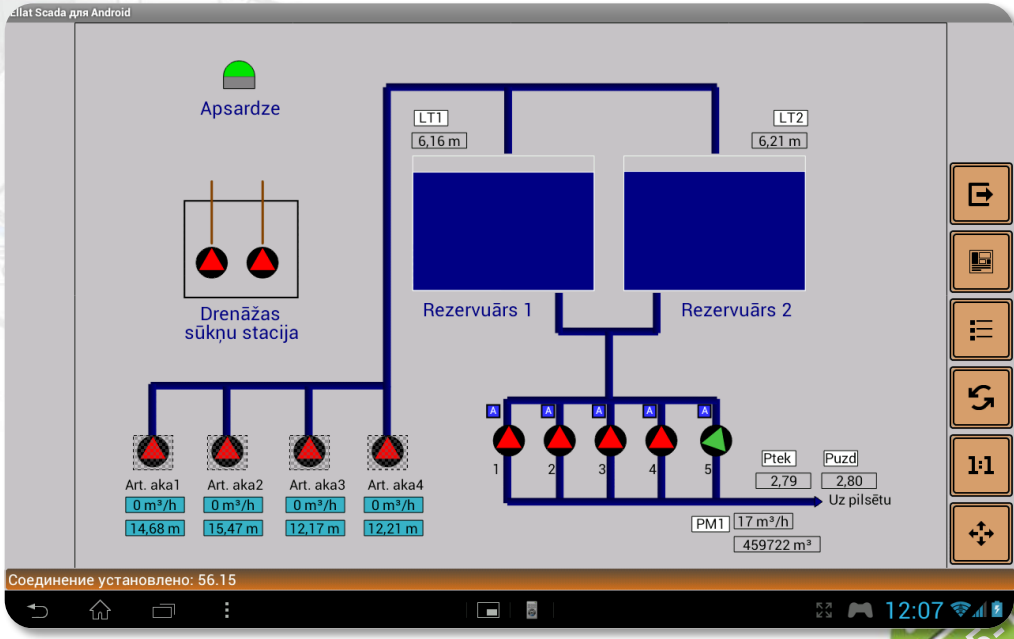
- Allow engineer and process owner to read and interpret some data from middle-high level components of ICS.
- Data is read-only; moreover, you don't have direct access to the PLCs, HMIs or SCADA servers -- your only ability is to read some variables or aggregated values.



SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

SCADA remote control apps

- Applications that allows remote (outside of safe perimeter or even plant network) monitoring/controlling of the industrial process
- For **ALL** applications in this group, we find pictures/schemes/architecture sketches/documents from vendor where mobile app is shown as a remote control client ***outside*** of the plant network (high-low levels)



Typical threats

Typical threats

Unauthorized physical access to device

Communication channel compromise

Application compromise

- Server-side
- Client-side

Physical device access

- Smartphone/tablet loss
- "Unlocked phone on the table"
- SDCARD data compromise

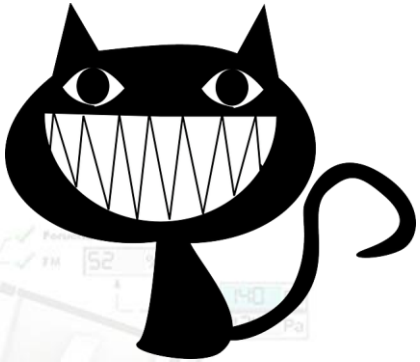
Communication channel compromise

Man-in-the-middle attack:

- Wi-Fi
  - Public AP/network
  - Private network compromise
  - Attacks against WPA2-Ent.
  - Faking AP with Karma, …
- GSM/GPRS
  - Fake AP

http://www.linuxx.eu/2014/09/osmocombb-hardware-and-software-setup.html

Application compromise



- Server-side:
  - SQL injection
  - Remote code execution
- Client-side:
  - RCE in client
  - Threat from other vulnerable or malicious (trojan?) application (in case of incorrect data storage)

Main threats summary

Control-room app

- Server DoS or compromise
- Lack of server-side data validation in terms of industrial process
- Compromise of stored data that could lead interface/feature modification (HMI apps)
- Client-side DoS

MES/OPC/Historian client

- Process information leak through protocol vulnerability
- Server DoS or compromise
- Client-side DoS or compromise
- Deceiving the operator by hiding alarms

SCADA remote control app

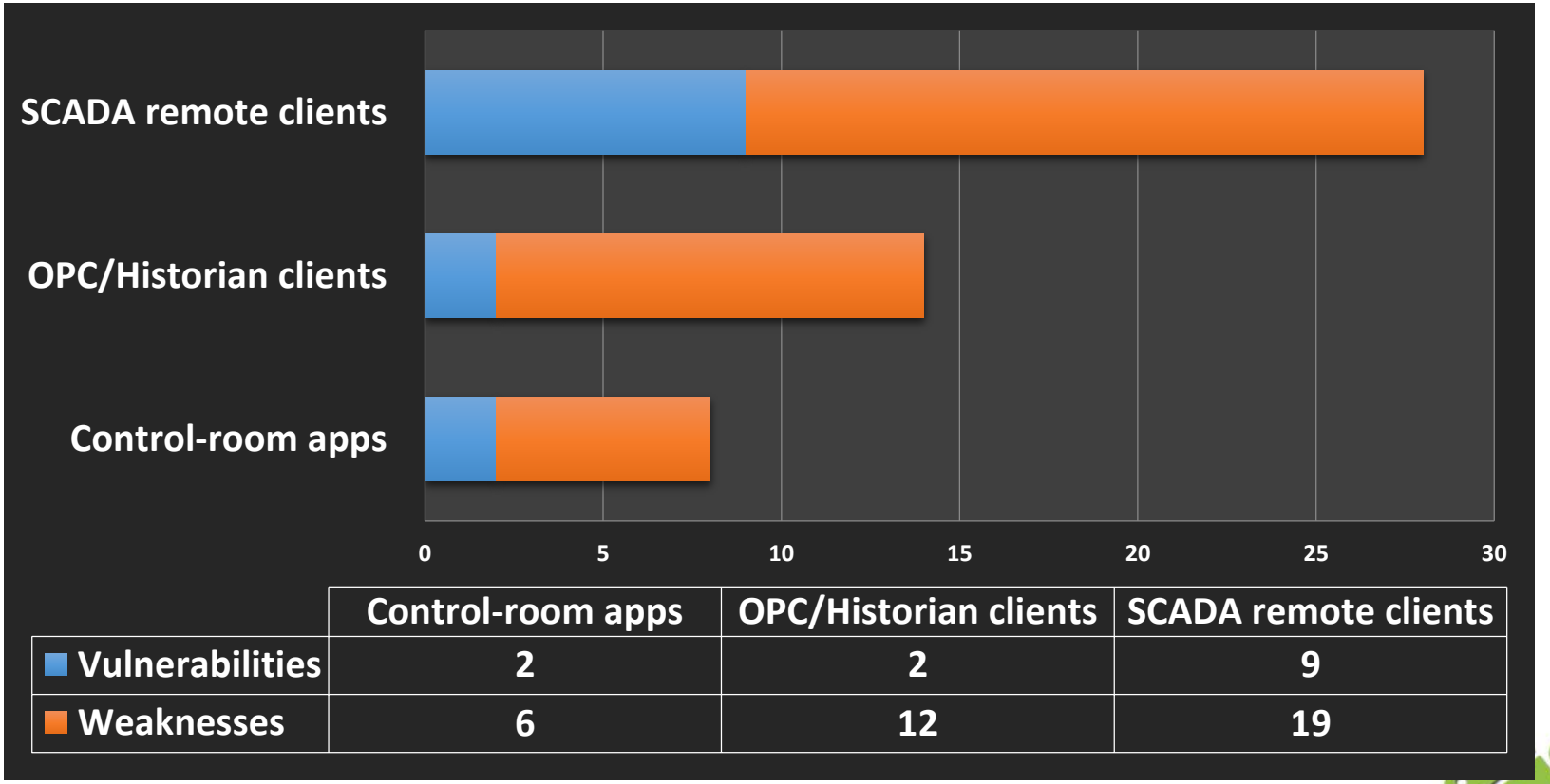
- Compromise of process through protocol or application vulnerability
- Lack of server-side data validation in terms of industrial process
- Compromise of process through vulnerability in server
- Client-side DoS or compromise

Analyzed applications list

Afcon Pulse	Autobase SCADA	CyBroDroid	Ellat SCADA	HMI MASTER
HMI OBA7	MiScout	OPC XML DA client	OPC XML DA Explorer	PLC-5 HMI
Pro-face Remote HMI	ProficySCADA	Prosys OPC UA client lite	S7 Android	Movicon Progea Web Client
ScadaTouch	Siemens LOGO! App	Wagoid	Watch*IT	WHS Live Light

Legend:	Control-room app	OPC-/MES-/Historian client	Remote SCADA client

Test results summary



	Control-room apps	OPC/Historian clients	SCADA remote clients
Vulnerabilities	2	2	9
Weaknesses	6	12	19

Test results summary

Vulnerability/weakness	Control-room app	OPC-/MES- client	Remote SCADA client
M1: Weak server-side controls	0	**1**	0
M2: Insecure Data Storage	**2**	0	**3**
M3: Insufficient transport layer protection	0	**3**	**6**
M5: Poor authorization and authentication	*n/a*	**3**	**4**
M6: Broken crypto	0	**2**	**7**
M7: Client side injection	**1**	0	0
M8: Security decisions via untrusted inputs	*n/a*	0	**1**
No password protection	**4**	**5**	**4**
Denial of Service	**1**	0	**3**

Authentication problems

Example vulnerabilities

But… responsible disclosure

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

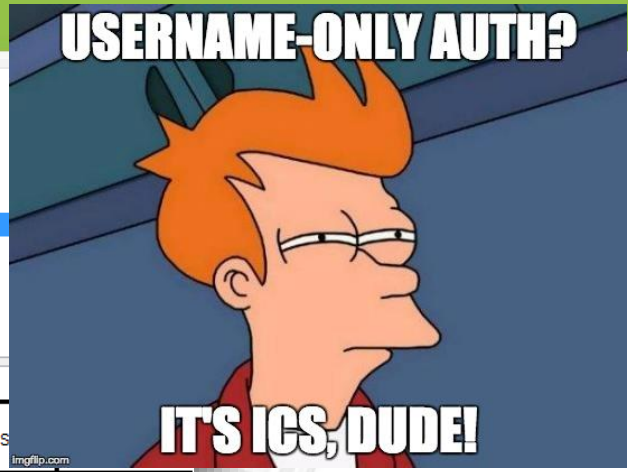
Plain-text authentication

```
Internet Protocol Version 4, Src: 192.168.100.193 (192.168.100.193), Dst: [redacted]
Transmission Control Protocol, Src Port: 52700 (52700), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 200
Hypertext Transfer Protocol
⊞ POST /service/check_username_password/ HTTP/1.1\r\n
⊞ Content-Length: 29\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
  Host: [redacted]\r\n
  Connection: Keep-Alive\r\n
  \r\n
  [Full request URI: http://[redacted]/service/check_username_password/]
  [HTTP request 1/1]
  [Response in frame: 24]
HTML Form URL Encoded: application/x-www-form-urlencoded
⊞ Form item: "username" = "guest"
⊞ Form item: "password" = "guest"
```

```
⊞ [Expert Info (Chat/Sequence): GET /webservice/api/associate.json/login?UserName=demo&Password=demo&IsMobi
  Request Method: GET
  Request URI: /webservice/api/associate.json/login?UserName=demo&Password=demo&IsMobile=true
  Request Version: HTTP/1.1
Host: [redacted]\r\n
Connection: Keep-Alive\r\n
Accept-Encoding: gzip\r\n
\r\n
[Full request URI: http://[redacted]/webservice/api/associate.json/login?UserName=demo&Pass
[HTTP request 3/16]
[Prev request in frame: 73]
[Response in frame: 110]
[Next request in frame: 144]
```

Username-only authentication

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM




```
<DefaultUserCheckWithVersionResponse
  xmlns="http://tempuri.org/">
 <DefaultUserCheckWithVersionResult>
  true
 </DefaultUserCheckWithVersionResult>
 <username>
  guest
 </username>
 <err_msg/>
</DefaultUserCheckWithVersionResponse>
</soap:Body>
</soap:Envelope>
```

CHECK DEFAULT USERNAME

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <DownLoadWithData2Response
   xmlns="http://tempuri.org/">
   <DownLoadWithData2Result>
    [truncated]c1Bhc3NDb2RlLDUyNDMyNzINCnNEZXNjcmlwdGlvbizsm7kgC
   </DownLoadWithData2Result>
   <result>
    2
   </result>
  </DownLoadWithData2Response>
 </soap:Body>
</soap:Envelope>
```

LOAD CONFIG IN BAS

```
sPassCode,5243272
sDescription,
nWebValueUpdateTime,0
sStartPage,StartUp.modX
bAutoOpenStartPage,True
bRightProgrammEnd,True
bRightTagChange,True
bRightConfigAlarm,True
bRightConfigEtc,True
bRightTagMemberAlarmLevel,Tru
e
bRightTagMemberDataSave,True
bRightAlarmConfirm,True
bRightAlarmEventDelete,False
bRightTagMemberViewRange,Tru
e
```

```
220  75 6c 74 3e 63 31 42 68  63 33 4e 44 62 32 52 6c   ult>c1Bh c3NDb2Rl
230  4c 44 55 79 4e 44 4d 79  4e 7a 49 4e 43 6e 4e 45   LDUyNDMy NzINCnNE
240  5a 58 4e 6a 63 6d 6c 77  64 47 6c 76 62 69 7a 73   ZXNjcmlw dGlvbizs
250  6d 37 6b 67 36 34 32 77  36 36 71 6f 49 4f 71 34   m7kg642w 66qoIOq4
260  73 4f 75 7a 75 43 44 73  67 71 7a 73 6d 71 6e 73   sOuzuCDs gqzsmqns
270  6e 70 41 4e 43 6d 35 58  5a 57 4a 57 59 57 78 31   npANCm5X ZWJWYWx1
```

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Built-in cryptokeys

```
public boolean a(String paramString, Context paramContext)
{
  try
  {
    SharedPreferences.Editor localEditor = PreferenceManager.getDefaultSharedPreferences(paramContext).edit();
    localEditor.putString("ApplicationPassword", a(paramString, "[illegible]NM"));
```

```
public byte[] c(String paramString1, String paramString2)
{
  try
  {
    SecretKeySpec localSecretKeySpec = new SecretKeySpec(paramString2.getBytes(), "Blowfish");
    Cipher localCipher = Cipher.getInstance("Blowfish");
```

```
private static final String CredBaseSeed = new String("420[illegible]#jms");
String str1 = new String(localHostData1.m_displayName + CredBaseSeed);
String str2 = localHostData1.m_userName;
Object localObject1 = localHostData1.m_password;
if (str2 != null) {}
try
{
  str2 = CredStore.encrypt(str1, str2);
  if (localObject1 != null)
  {
    String str3 = CredStore.encrypt(str1, (String)localObject1);
    localObject1 = str3;
```

```
private static String cle = "[illegible]";
private static byte[] cleInBytes;
private static SecretKeySpec clef;
private static byte[] ivBytes;
private static byte[] ivBytesIOs;

static
{
  byte[] arrayOfByte = new byte[16];
  arrayOfByte[0] = 103;
  ivBytes = arrayOfByte;
  ivBytesIOs = new byte[16];
}

public Cryptage()
{
  try
  {
    cleInBytes = cle.getBytes("UTF8");
    clef = new SecretKeySpec(cleInBytes, "AES");
    return;
  }
```

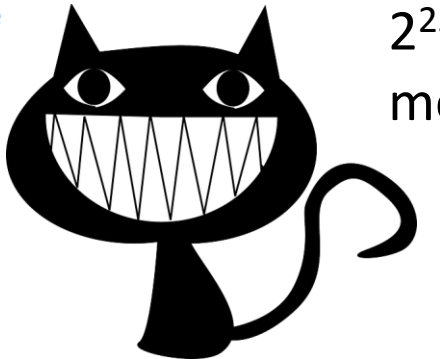
Weak hashes

```
String str2 = ((EditText)findViewById(2131230733)).getText().toString();
if (str1.length() == 0)
{
  Toast.makeText(this, "Input the Username", 1).show();
  return false;
}
String str3 = Long.toString(ZipPassword(str2));
Log.i("passcode", str3);
localNetWebServiceCall.addProperty("username", str1);
localNetWebServiceCall.addProperty("password", str3);
localNetWebServiceCall.Call();
```

```
long ZipPassword(String paramString)
{
  int i = 0;
  int j = 0;
  String str = paramString.toUpperCase();
  for (int k = 0;; k++)
  {
    if (k >= str.length()) {
      return j + 65536 * i;
    }
    i = (short)(i ^ str.charAt(k));
    j = (short)(j + str.charAt(k));
  }
}
```

Let p_i – password symbols, then:

$$\sum i\ p_i + 65535 \oplus_i p_i < 2^{25}$$

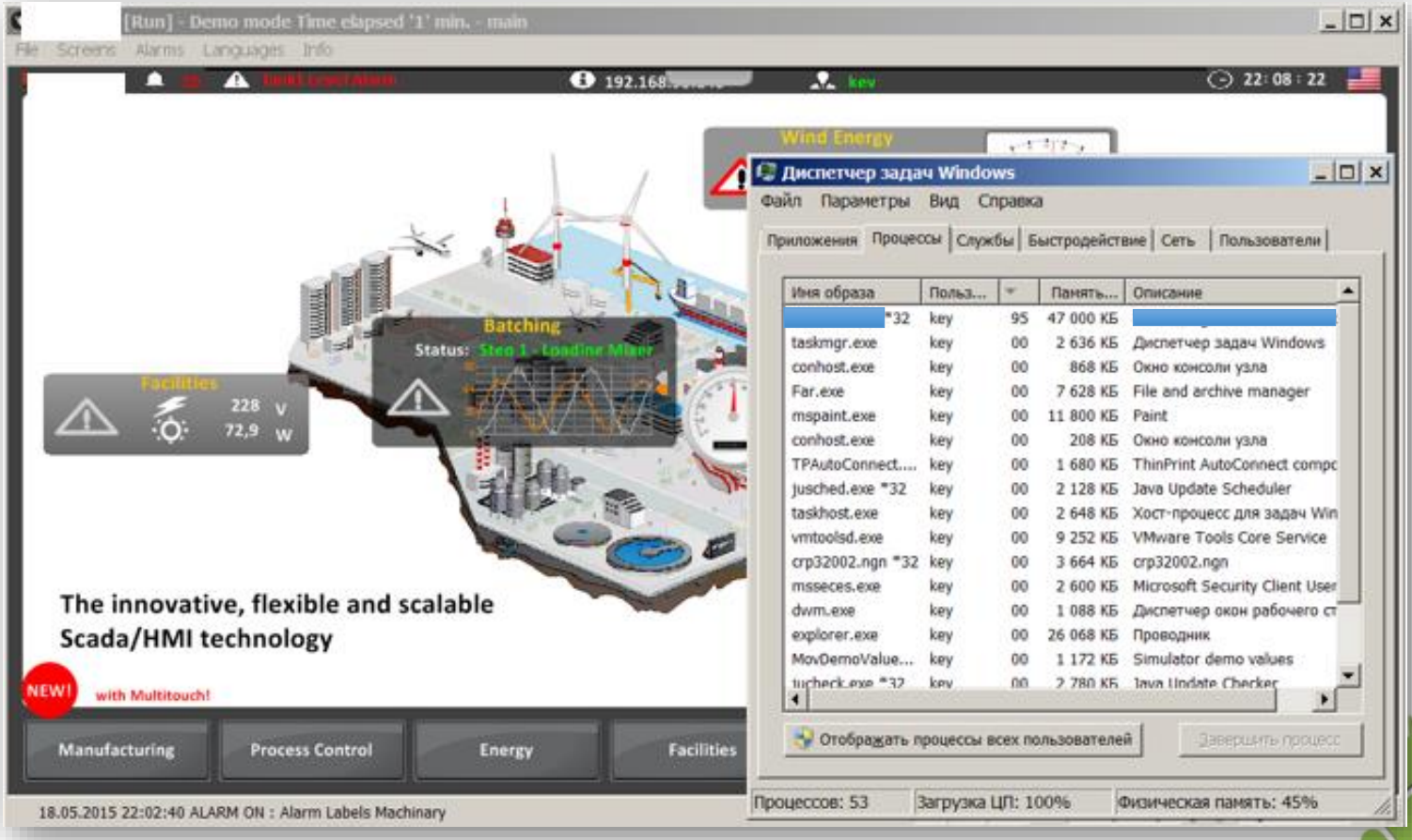


2^{25} is easily crackable on most modern computers

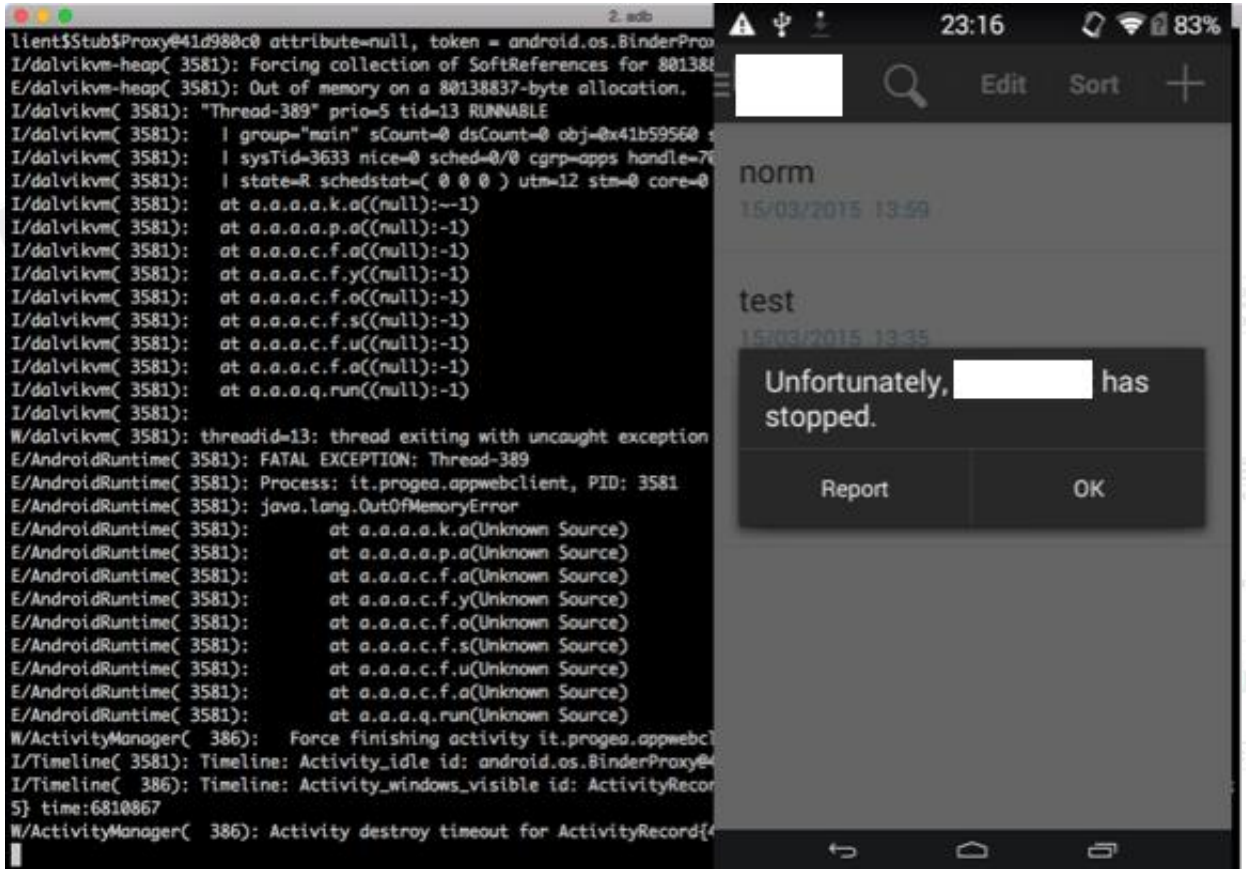
Crypto in mobile ICS apps…

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Server-side Denial of service



Client-side Denial of Service



Double free(..)?

```
I/DEBUG   (   77): backtrace:
E/NativeCrashListener(  386): Exception dealing with report
E/NativeCrashListener(  386): libcore.io.ErrnoException: read failed: EAGAIN (Try again)
E/NativeCrashListener(  386):   at libcore.io.Posix.readBytes(Native Method)
E/NativeCrashListener(  386):   at libcore.io.Posix.read(Posix.java:128)
E/NativeCrashListener(  386):   at libcore.io.BlockGuardOs.read(BlockGuardOs.java:149)
E/NativeCrashListener(  386):   at com.android.server.am.NativeCrashListener.consumeNativeCrashData(NativeCrashListener.jav
a:240)
E/NativeCrashListener(  386):   at com.android.server.am.NativeCrashListener.run(NativeCrashListener.java:138)
I/DEBUG   (   77):     #00  pc 003aabe4  /data/app-lib/                                  .so (QIOStatusListModel::getIOData
(int)+159)
I/DEBUG   (   77): AM write failure (32 / Broken pipe)
I/DEBUG   (   77):     #01  pc 0000dcd7  /system/lib/libc.so (free+10)
I/DEBUG   (   77):
I/DEBUG   (   77): stack:
I/DEBUG   (   77):          54c4d770  ffffffff
I/DEBUG   (   77):          54c4d774  2bb5c1a8
I/DEBUG   (   77):          54c4d778  54c4d808
I/DEBUG   (   77):          54c4d77c  400382fb  /system/lib/libc.so (dlfree+50)
I/DEBUG   (   77):          54c4d780  40071000  /system/lib/libc.so
I/DEBUG   (   77):          54c4d784  54c4d7b4
I/DEBUG   (   77):          54c4d788  54c4d7b4
I/DEBUG   (   77):          54c4d78c  2bb5c1a8
I/DEBUG   (   77):          54c4d790  ffffffff
I/DEBUG   (   77):          54c4d794  40034cd9  /system/lib/libc.so (free+12)
I/DEBUG   (   77):          54c4d798  537a6a61  /data/app-lib/                    /libQt5Core.so (QObjectPrivate::~QObje
ctPrivate())
I/DEBUG   (   77):          54c4d79c  54938c85  /data/app-lib/                    /libLOGOAPP.so (QDebug::~QDebug()+86)
I/DEBUG   (   77):          54c4d7a0  53834eac  /data/app-lib/                    /libQt5Core.so
I/DEBUG   (   77):          54c4d7a4  54c4d7b4
I/DEBUG   (   77):          54c4d7a8  54c4d7c8
I/DEBUG   (   77):          54c4d7ac  54956bc7  /data/app-lib/c                    libLOGOAPP.so (QIOStatusListModel::ge
tIOData(int)+130)
I/DEBUG   (   77):     #00  54c4d7b0  00000001
I/DEBUG   (   77):          54c4d7b4  53836088  /data/app-lib/                    /libQt5Core.so (QListData::shared_null
)
```

Testing methodology

Test steps

Analysis and filling 'Test checklist'

Client and server fuzzing

Deep analysis with reverse-engineering

Checklist

Application

- Purpose of app: SCADA/HMI/PLC/OPC/etc..
- Permissions
- Password protection
- Native code
- Web-based components

Protocol

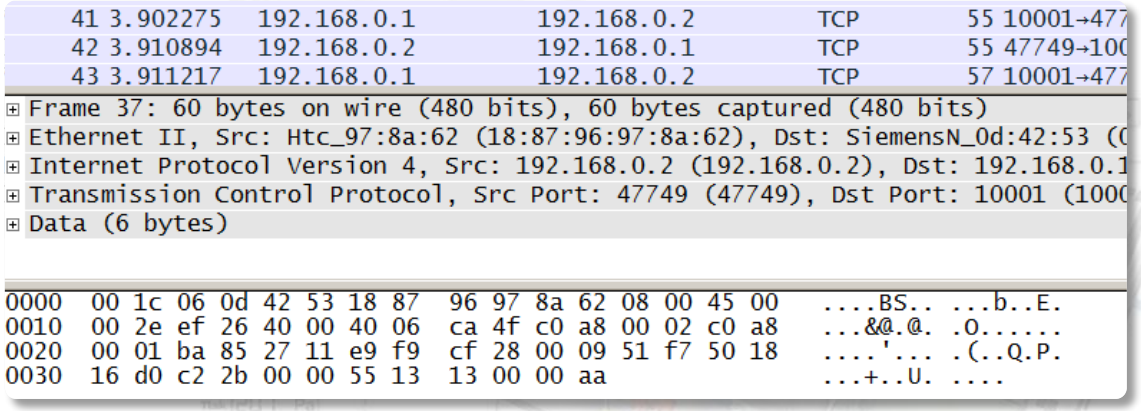
- Authentication.
- Tokens/cookies/sessions
- SSL
- XML
- Server API

Storage

- Connection strings/passwords
- Data/Projects/HMI interfaces etc..

Fuzzing

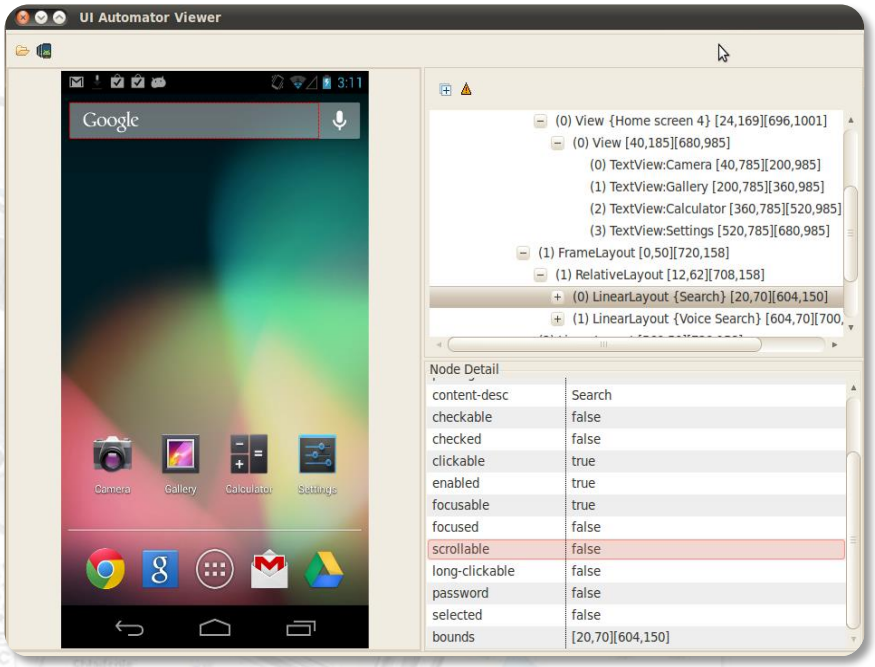
- Some applications used vendor-specific protocols to interact between client and server



- Full reverse-engineering of such protocols could take infinite amounts of time
- To simplify security testing in such cases we used fuzzing approach

Automating fuzzing with Ulautomator

- To do fuzzing we need periodic communications between mobile app and server
- Problem: mobile app is a GUI app. In most cases it won't do any interaction with server without user command
- Solution: use Android UIAutomator (GUI testing tool) to emulate user taps in mobile app



Native UIs

- **Problem:** many of mobile ICS apps uses native code (C++ or Delphi, arm7eabi). GUIs of such applications also use native elements. Uiautomator standard methods has no support of interacting with them
- **Solution:** simple custom extension of uiautomator
- **Method:**
  - Do first round of fuzzing with no mutations (with *correct* data)
  - Capture series of screenshots during first round and put them in cache
  - On next round, when crashes/disconnects could occur. We will detect them by comparing current state screen with appropriate screen in cache

Fuzzing proxy – erlamsa

- Written in Erlang.
- Multithreaded
- Supports TCP, UDP and HTTP GET/POST data fuzzing
- Implements almost all radamsa mutations in the same way (excl. XML).
- Modified/new mutations:
  - Some new raw binary mutations
  - Length-field prediction
  - Binary-structure prediction
  - Modified ascii-bad mutator (to increase the probability of XML injections)
  - Some other minor corrections and new things
- Not magic or completely original tool, but just fitting “old good” radamsa in a fuzzing proxy task

Fuzzing architecture

GuiTester Lib

Gui test scenario (erlang)

UDP, erlang objects

Uiautomator app

Fuzzed application

Fuzzing proxy (erlamsa)

Server (SCADA, demo application, PLC, e.t.c.)

Test scenario example

```
show_status_window(S) ->
        {ok} = nuitestclient:send_cmd(S, {click, 200, 600}, 3000),
        {ok} = nuitestclient:send_cmd(S, {click, 200, 350}, 3000),
…
prepare(IP, Port) ->
        S = nuitestclient:connect(IP, Port),
        run_logoapp(S), timer:sleep(5000),
        {ok} = nuitestclient:send_cmd(S, {putscreenoncache, "working"}, 5000), S.
…
fuzz(S, false, FailCnt) ->
        io:format("Relauncing...~n"),
        {ok} = nuitestclient:send_cmd(S, {click, 358, 463}, 3000),
        timer:sleep(1000), show_status_window(S), timer:sleep(1000),
        {ok, Res} =
                nuitestclient:send_cmd(S, {comparescreenoncache, "working"},
5000),
        fuzz(S, Res, FailCnt + 1).
```

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Fuzzing in progress ☺

Attack examples

Attack 1: insufficient validation of input parameters from mobile app

- **To conduct this vector, attacker should:**
  - Either be connected to the network with the target client/server (in case of control room application)
  - Or knew remote control SCADA endpoint (server address) and have valid login credentials

- E.g. potential victim uses his smartphone with public Wi-Fi AP (e.g. in restaurant or trade center). If there are no encryption (or weak/vulnerable) in place, attacker could MitM the connection with server (using, for example, ARP spoofing). When the legitimate request will go to the system, they could be changed

Attack 1: query

```
GET /scgi/?c8785.<removed>=26999& HTTP/1.1
Cookie:
sessionid=98aec9bc19d2bed32d3cc9a1140920e2
Host: <removed>
Proxy-Connection: close
Connection: close
```

Attack 1: results

Attack 2: compromising HMI storage

- Several applications stores HMI databases (circuits, interfaces, connection parameters and HMI projects) insecurely (e.g. on SDCARD)
- If HMI project data will be somehow compromised, attacker slightly reconfigure or change its interface. He could replace component events and logic, or, for example, sensors data sources
- Unsuspecting operator is “compromised” now and could make a wrong decision.

Ways of HMI project compromise

Mobile malware

SDCARD access

HMI project

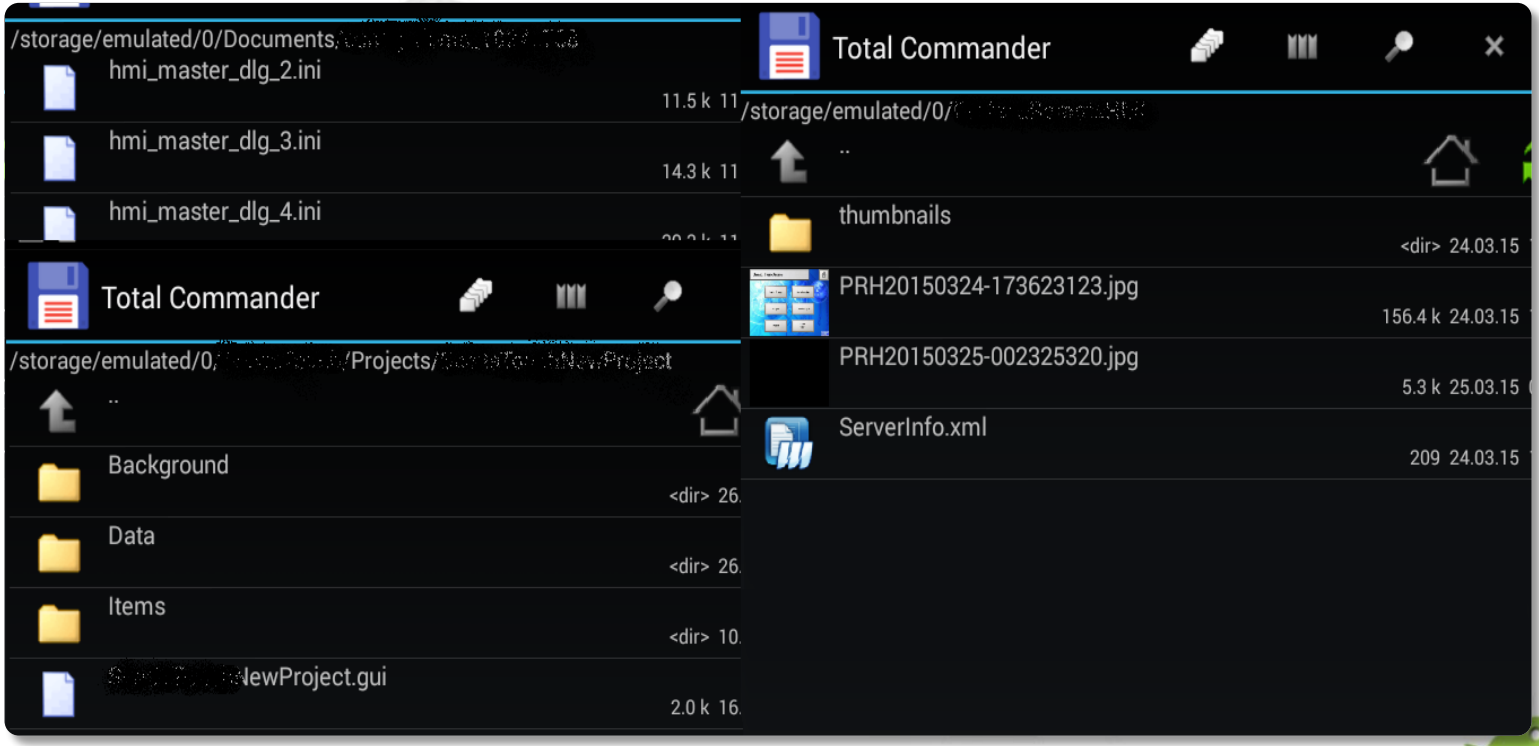
Other application vulnerability

ABUSING ANDROID APPS AND GAINING EXECUTION

Attack 2: Original HMI interface

SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Attack 2: HMI project storage



SCADA AND MOBILE: SECURITY ASSESSMENT OF THE APPLICATIONS THAT TURNS YOUR SMARTPHONE INTO A FACTORY CONTROL ROOM

Attack 2: HMI project file format

Clear text

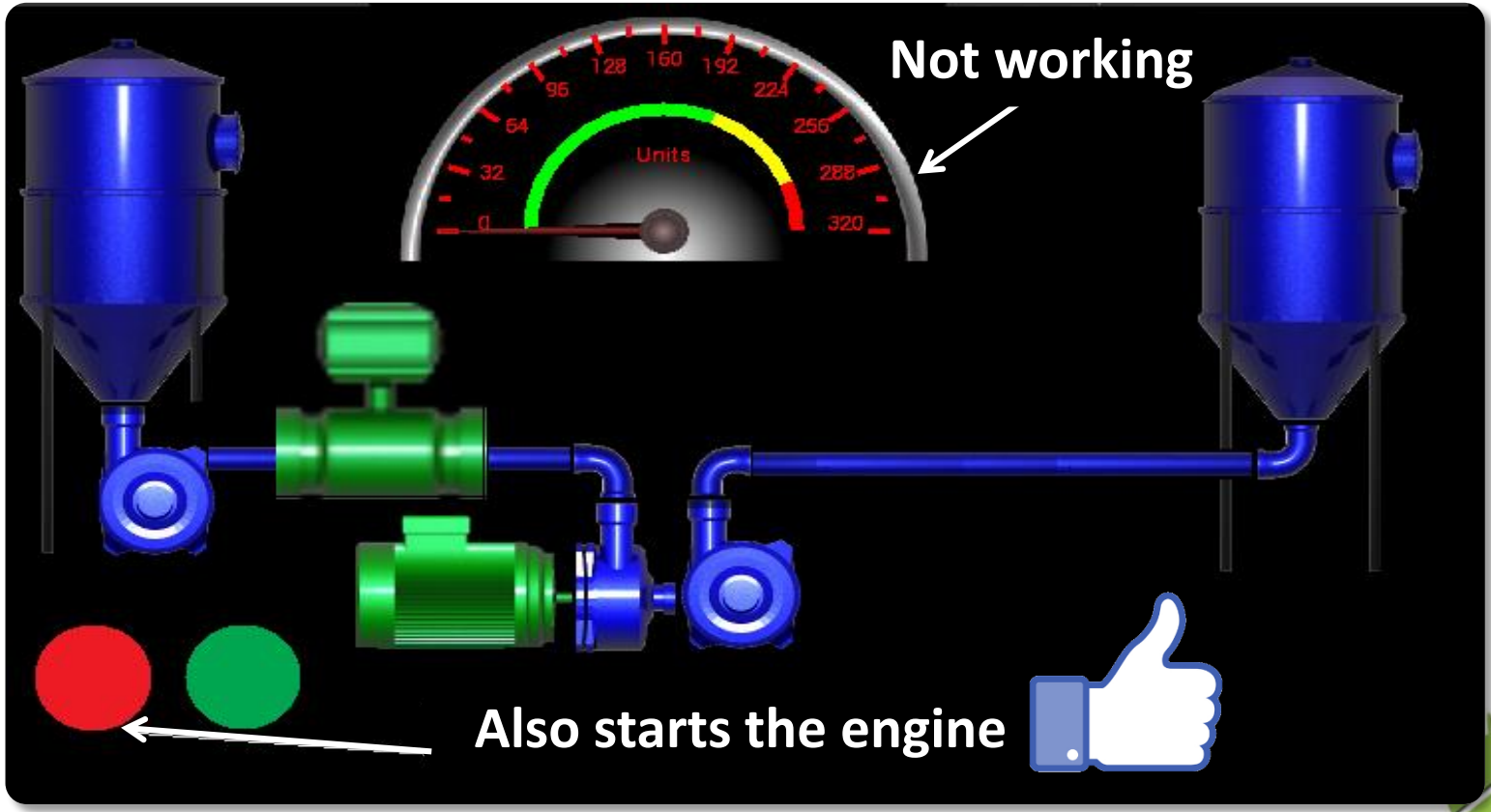
```
<object>
  <type>bargraph</type>
  <var>c8785.industry_boiler_level</var>
  <min>0</min>
  <max>50</max>
  <barcolor>#408FE3</barcolor>
  <background></background>
  <direction>vertical</direction>
  <reverse>0</reverse>
  <width>225</width>
  <height>220</height>
  <x>432</x>
  <y>438</y>
</object>
<object>
  <type>bargraph</type>
  <var>c8785.industry_boiler_level</var>
  <min>50</min>
  <max>75</max>
  <barcolor>#E3E140</barcolor>
  <background></background>
  <direction>vertical</direction>
  <reverse>0</reverse>
  <width>225</width>
  <height>110</height>
  <x>432</x>
  <y>328</y>
</object>
<object>
  <type>bargraph</type>
  <var>c8785.industry_boiler_level</var>
  <min>75</min>
  <max>100</max>
  <barcolor>#EB2324</barcolor>
  <background></background>
  <direction>vertical</direction>
  <reverse>0</reverse>
  <width>225</width>
  <height>110</height>
  <x>432</x>
  <y>218</y>
```

Encrypted

```
f5pKwKbp9jX4aHyVQhPi2NIfAMvNIiIjPdlk71tku21ELYYgP7RMVADp
xokhCqj0yZcmwiF066B3bLSDhlXJxapq9K5M69yuHFldSNwd5GA/1
dcK8Giz+g2KmRAPDmCs3MBfSL6caJ09exSJdJJMDOLOvEwlulz68
MH70YqeOKz5H7d2dPM2BPUr1IvuMxmKEMtbFxj21VtMGBeMYBX
FMouodEUnCsOiXvLOKR2lMbCKwXVDZQ3LvKk0rJ/UpkLpp/yjltR3G
M+W7pcDegqkPibCnH0Nme4NbDiZeAkNS95YnlShwtzwryXYjsREYh
8jUjq0UuHpeudxISF6pzaokMHlfVHdwcNoLAcLfzXWDSgpkBrHReLD
bKc1+RhPPPMF43C+XeZvyeCG82pQgtE0YsIV4EmCygU5THTtPxEZ
z3V2go0Ry8UMHfmkHKEcQQ7Q61pV3t+zmwzLPwuhunBTDlvoAU+
9PIPw2O2VrAKYkniN5YsOVn6Nrh6pBvt1qb6ybBjZEktbFi66KVcyel5
kLk5ADP9kXsGPyyk0gXo7eKivdHwk1kqu8XnbsQB/pIzWHmAHzqKE
BK9w8EzlDch7U6kuAlUG3XYs4zC496oOvKq43xa3sMNid8r9IADYYZ
X5acD3AqTs1K5juyzrC49+UGfV1N6A9ovCqMPoxqHJZzCnohdq+nX
mr1tklk3XybGzIzNp4Xor/CT5680FBk+JljEt9ctoDz30okH+ZA9mh/jos
JcbSnJP0odA2Gnjj3xP7mAexBrOldfcFAKY0Oa6ZvXI2fukjZkhFGXCE
yk63BWxdvSB3FBR507mwscwadgkfrjQJxW68ZZB4sqNBZc/TPv17
lFziSxCogf61ubYN830+PZPS516++QkRwYk/CrhtltfOSBYQwK30j4W0
axLnq+00q6gNqVJvzklnQL5DAadjteAADaZKUs8edPIKhaO3mmll9wk
vsbrh50ldtd+OVhlQKH1f0qSCMdFddRo+BNdrNzDnL5PYnHlUaUNvH
vgNY8QEbRiwMKXC/ZnRIPdnDvyyQD2UNiC2hgm+IqDofL7w2h+8Xl
VfWTA6SS3tF2BclerzZOKiEScreyKUImpgxPTN0GtltutDdX1xkVM7h
1kDLSz3csxlFCPqIWSHjJcEU8oXAXH519itxASY7fkVZ3DIKrLzN/b6
4YpLJ6zomMv3ZfuwxemTm672zkGSlyXzvBhrTRFdDu1o9PzhX+NC
+gJMFmUur0nT6rZhG8WWbv3l0aNDXIeFO8H4epWiu5gVuoygWHJ
sof9Me8+5z01u+7JyBjql9gO3G4Zu9CA0sggHuQwQujcGNwXrnEem
3PepVSRHwUAF835jW9TbXzD4qfxycSjEBok+fd1W8dKmOEfLJLNC
f0KuaFXtJADRutT7EQRZND2llVwA7WAV+Ohn3R4BFxu0lYKeP/K42
```

- Different formats
- May be either encrypted or not
- No difference at last

Attack 2: Changed HMI panel



Attack 2: Changed HMI panel

Attack 2: …

original pic source: http://www.smartcityexpo.com/new_products/-/newness/955751/Afcon-s-Pulse-Mobile-2-0-for-cities?return=microsite

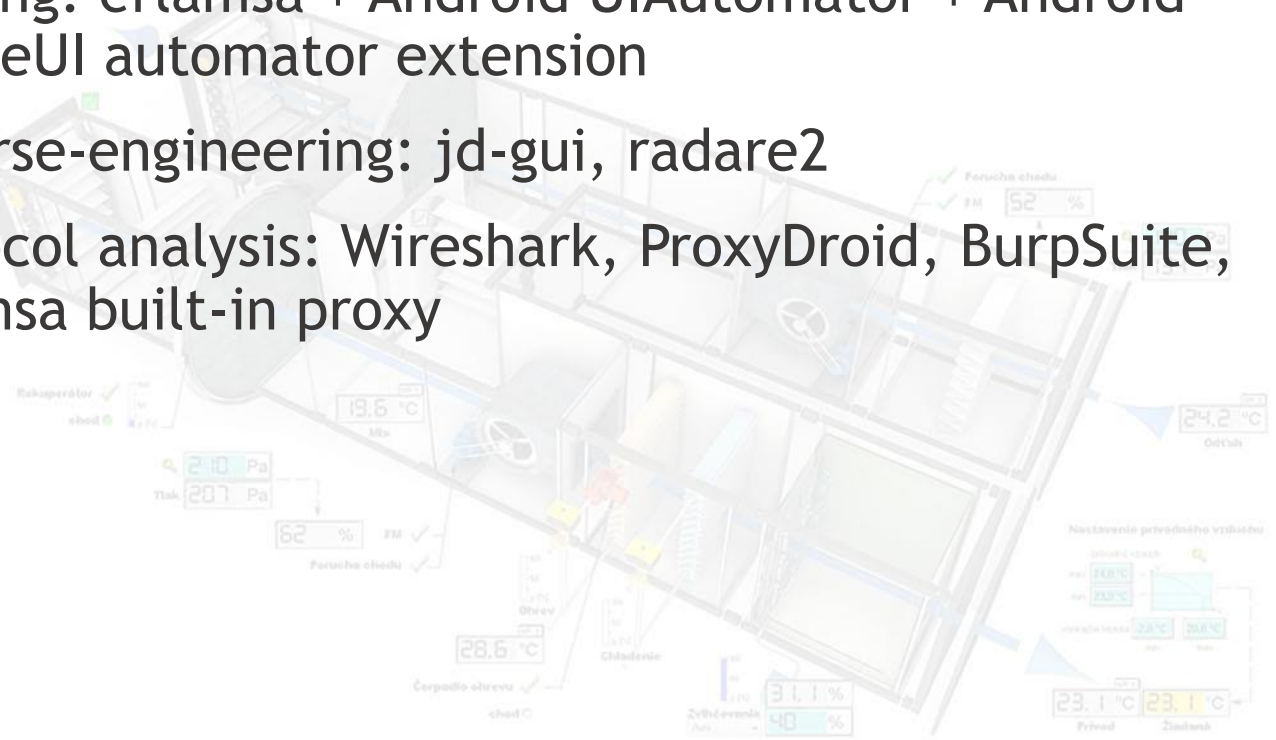
Conclusions

- We've reviewed 20 android applications that works with ICS infrastructure in different ways
- Didn't find at least one application without weaknesses and/or vulnerabilities
- Discovered about 50 vulns/weaknesses
- The most dangerous consequence of these vulnerabilities is «compromising» the operator himself, i.e. inspire him to have false understanding of current industrial process state
- SCADA and ICS comes to the mobile world recently, but brings old approaches and weaknesses. Hopefully, due to the rapidly developing nature of mobile software, all this problems will soon be gone

Message to *some* developers

Used tools

- Fuzzing: erlamsa + Android UIAutomator + Android NativeUI automator extension
- Reverse-engineering: jd-gui, radare2
- Protocol analysis: Wireshark, ProxyDroid, BurpSuite, Erlamsa built-in proxy

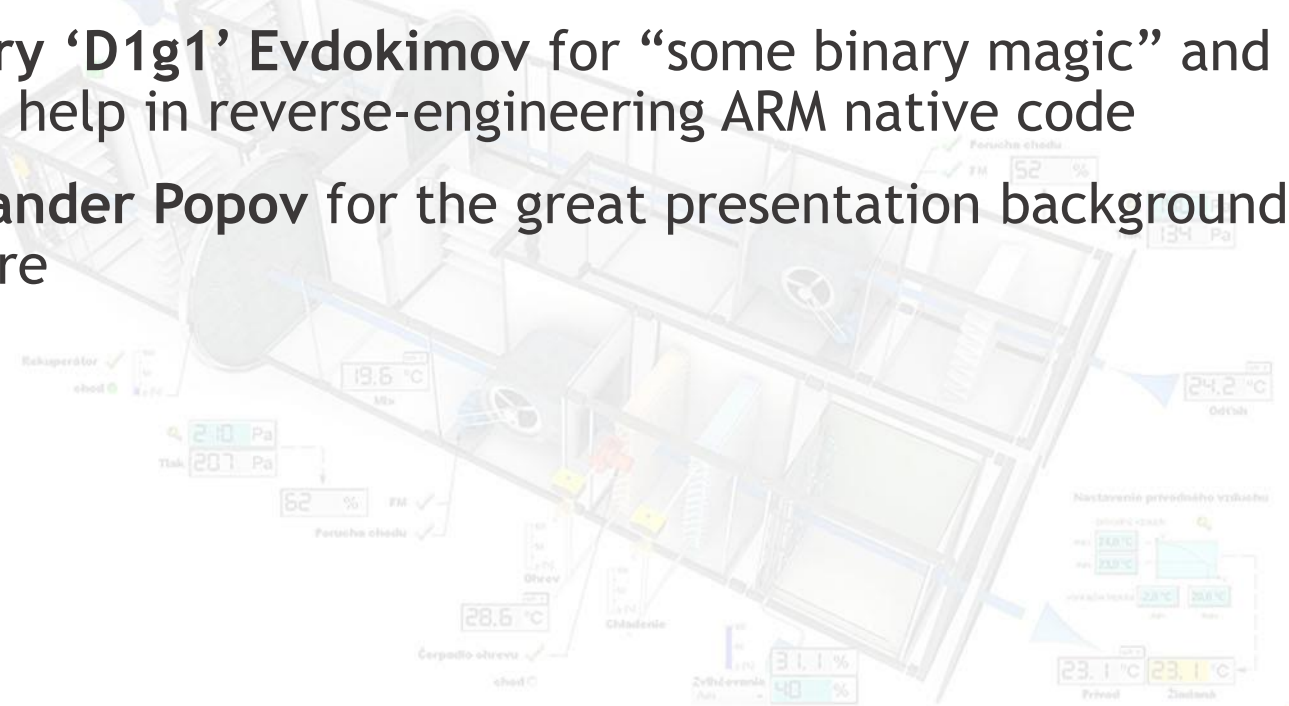


Links

- http://github.com/Darkkey/AndroidHMISecurity -- actual whitepaper, sources & binaries of Android NativeUI, will be filled upon disclosure process…
- http://github.com/Darkkey/erlamsa -- erlamsa mutational-based fuzzer proxy (fork of radamsa)

Thanksgiving service

- **Marina Krotofil** for help and many useful ideas
- **Dmitry 'D1g1' Evdokimov** for "some binary magic" and great help in reverse-engineering ARM native code
- **Alexander Popov** for the great presentation background picture



Q&A

http://dsec.ru

@dsecru

@dark_k3y