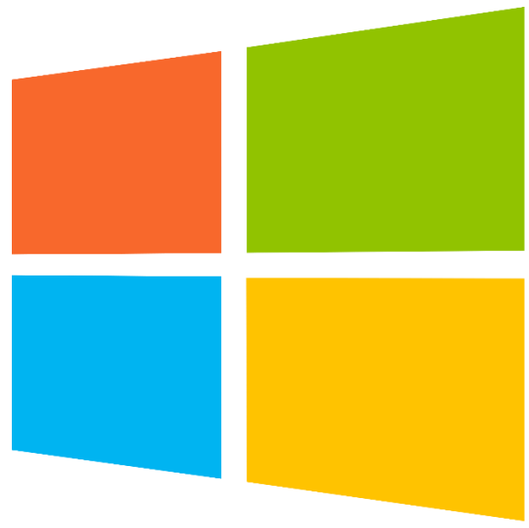
RINA on Windows

A brief intro to Win stack

Vladimír Veselý

2018-05-23



OS Market Share ❶

- All devices

2018	Win10	Win8	Win7	Vista	WinXP	Linux	Mac	Chrome OS	Mobile
April	44.6%	7.1%	23.8%	0.1%	0.4%	5.5%	10.6%	0.3%	7.8%
March	43.7%	7.4%	24.7%	0.1%	0.4%	5.5%	10.4%	0.3%	7.8%
February	42.3%	7.6%	25.6%	0.1%	0.5%	5.6%	10.3%	0.3%	7.9%
January	41.6%	7.9%	26.3%	0.1%	0.5%	5.7%	10.0%	1.3%	7.7%

- Mobile device

2018	Total	iOS*	Android	Windows	Others
April	7.75 %	1.32 %	6.21 %	0.16 %	0.04 %
March	7.77 %	1.33 %	6.18 %	0.22 %	0.04 %
February	7.85 %	1.30 %	6.29 %	0.20 %	0.06 %
January	7.74 %	1.21 %	6.29 %	0.18 %	0.06 %

- Data from https://www.w3schools.com/browsers/browsers_os.asp

OS Market Share ❷

- Desktop devices

- Mobile device

- Data from http://gs.statcounter.com/os-market-share

Message

- ...taken

- Let's focus on

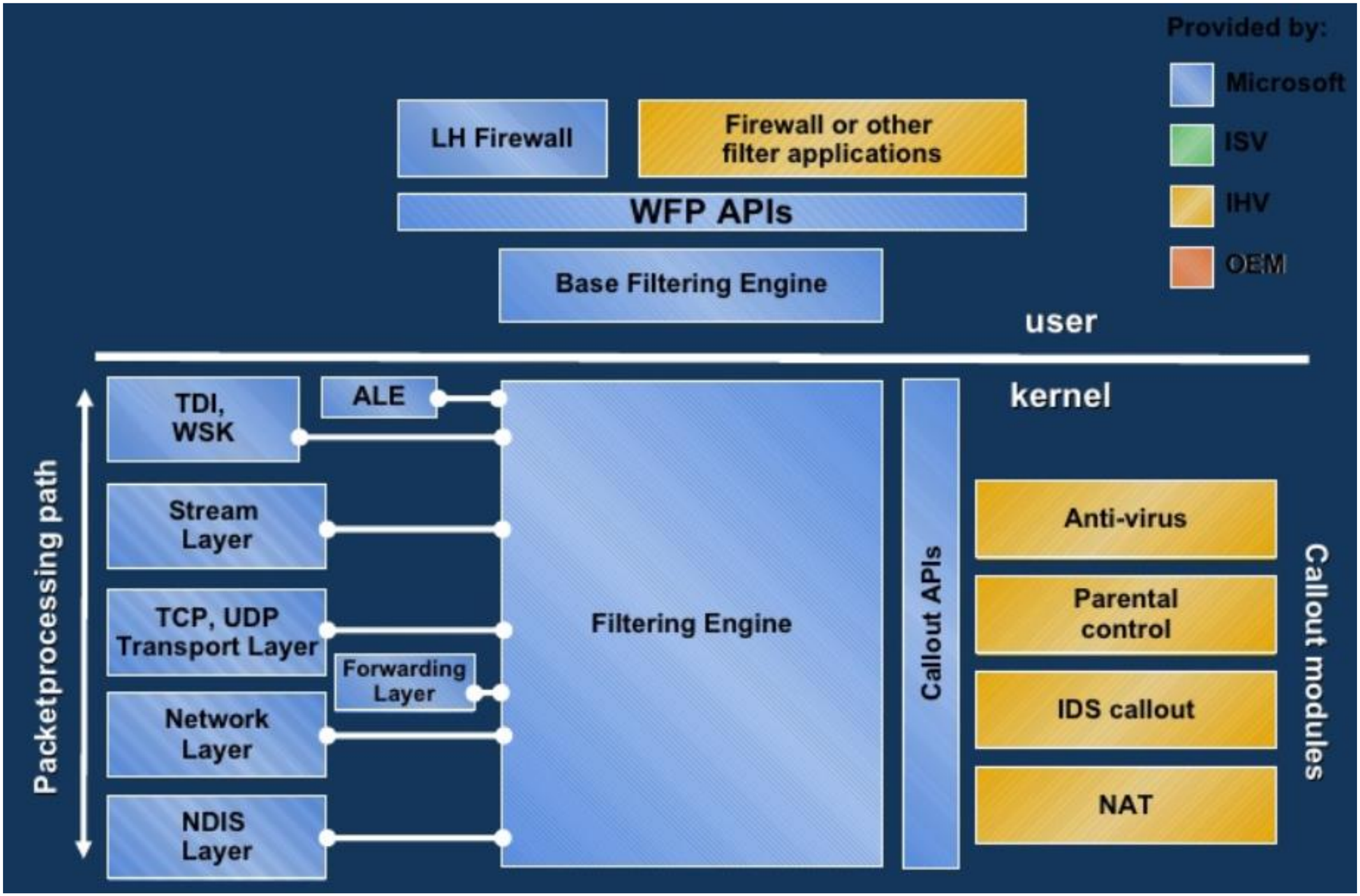
76% - 81% of usual web users

RINA on Windows

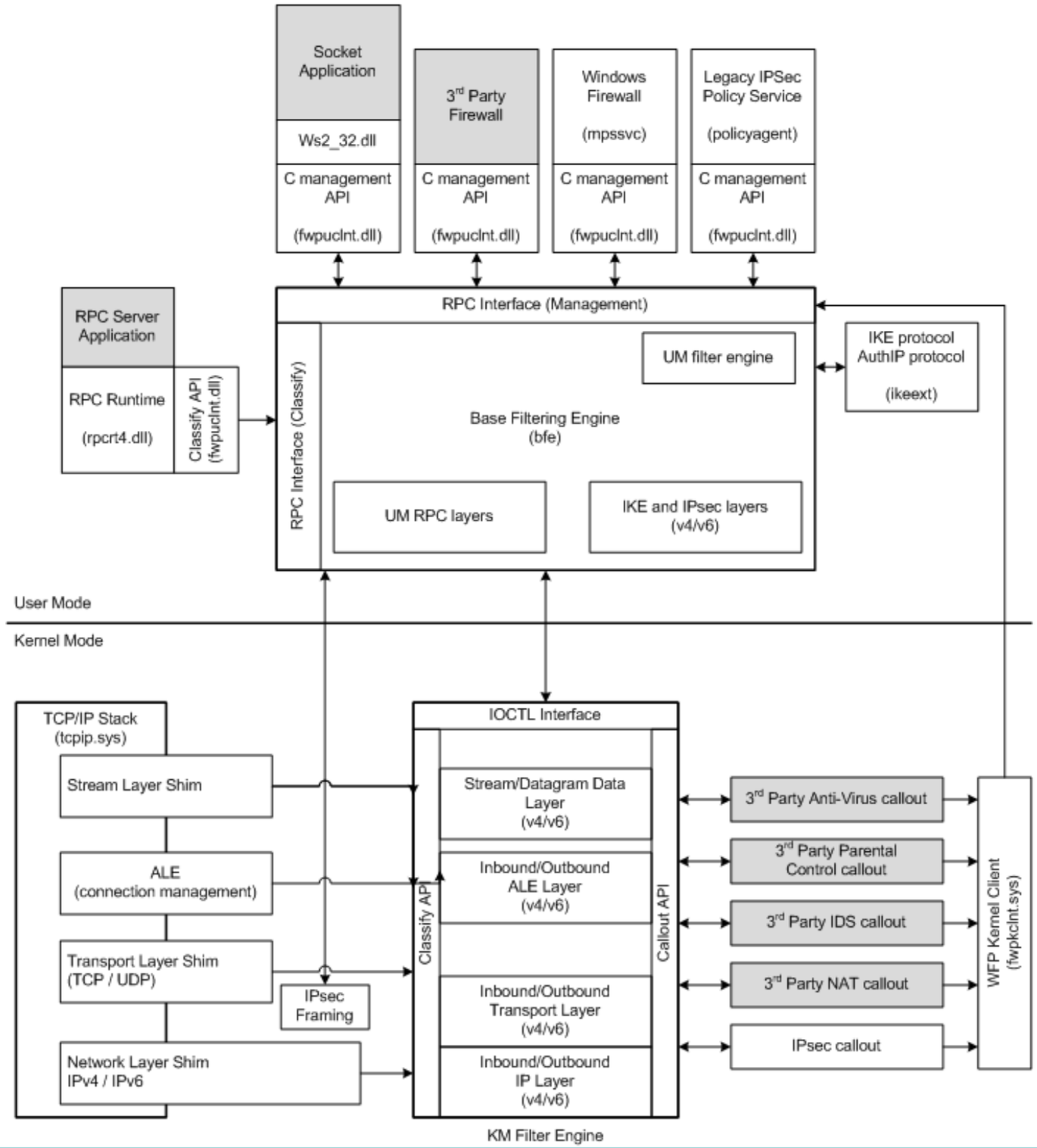
- Requirements
  1) Communicate directly with NIC
  2) Flexible shim-DIFs (RINA over L2, L3, L4 and L7)
  3) Useful debugging (via Wireshark)

- Implementing RINA stack has many functional similarities as

Windows Network Stack ❶



Windows Network Stack ❷



Key Components

- **Windows Filtering Platform (WFP)**
  - WFP exposes a set of APIs to permit, block, modify and/or secure inbound/outbount traffic
  - Both user-space (apps and services) and kernel (callbacks) APIs

- **Winsock Kernel (WSK)**
  - If needed then similar to “RAW sockets”
  - Able to leverage existing TCP/IP functionality

WFP - Callouts

- Callout drivers can perform the following tasks:
  - Deep inspection
  - Packet and Stream modification
  - Data logging
    - *It helps to avoid usual TCP/IP stack processing*
- Callout drivers can be hooked at different layers in stack
  - L2, L3, L4, L7 layers
  - *Here goes your Shim-DIF functionality*

Features

- C++
  - Visual Studio has templates for these kind of apps
  - Including Wireshark dissectors

- User-space stack implementation
  - *because there is no way how to write own kernel modules*
  - Ouroboros? (POSIX conformity, MinGW)

Hooking Service into WFP

```c
#include <windows.h>
#include <fwpmu.h>
#include <stdio.h>
#pragma comment(lib, "fwpuclnt.lib")
#define EXIT_ON_ERROR(fnName) \
   if (result != ERROR_SUCCESS) \
   { \
      printf(#fnName " = 0x%08X\n", result); \
      goto CLEANUP; \
   }
const GUID PROVIDER_KEY =
{
   0x5fb216a8,
   0xe2e8,
   0x4024,
   { 0xb8, 0x53, 0x39, 0x1a, 0x41, 0x68, 0x64, 0x1e }
};
#define SESSION_NAME L"SDK Examples"
DWORD Install(
         __in const GUID* providerKey,
         __in PCWSTR providerName,
         __in const GUID* subLayerKey,
         __in PCWSTR subLayerName
         )
{
   DWORD result = ERROR_SUCCESS;
   HANDLE engine = NULL;
   FWPM_SESSION0 session;
   FWPM_PROVIDER0 provider;
   FWPM_SUBLAYER0 subLayer;

   memset(&session, 0, sizeof(session));
   session.displayData.name = SESSION_NAME;
   session.txnWaitTimeoutInMSec = INFINITE;
```

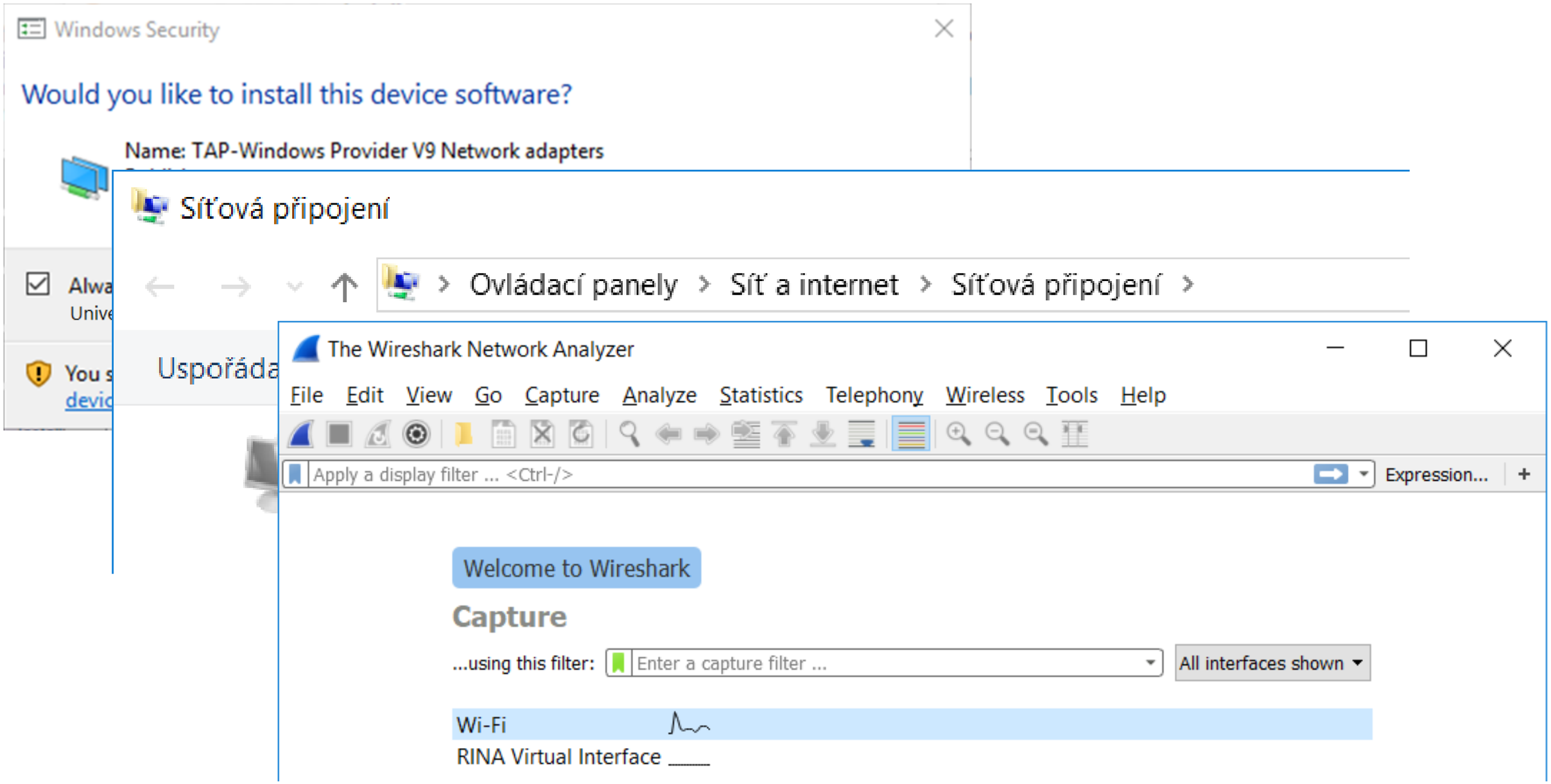
```c
   result = FwpmEngineOpen0(
               NULL,
               RPC_C_AUTHN_DEFAULT,
               NULL,
               &session,
               &engine
               );
   EXIT_ON_ERROR(FwpmEngineOpen0);

   result = FwpmTransactionBegin0(engine, 0);
   EXIT_ON_ERROR(FwpmTransactionBegin0);

   memset(&provider, 0, sizeof(provider));
   provider.providerKey = *providerKey;
   provider.displayData.name = (PWSTR)providerName;
   provider.flags = FWPM_PROVIDER_FLAG_PERSISTENT;

   result = FwpmProviderAdd0(engine, &provider, NULL);
   if (result != FWP_E_ALREADY_EXISTS)
   {
      EXIT_ON_ERROR(FwpmProviderAdd0);
   }

   memset(&subLayer, 0, sizeof(subLayer));
   subLayer.subLayerKey = *subLayerKey;
   subLayer.displayData.name = (PWSTR)subLayerName;
   subLayer.flags = FWPM_SUBLAYER_FLAG_PERSISTENT;
   subLayer.providerKey = (GUID*)providerKey;
   subLayer.weight = 0x8000;

   result = FwpmSubLayerAdd0(engine, &subLayer, NULL);
   if (result != FWP_E_ALREADY_EXISTS)
   {
      EXIT_ON_ERROR(FwpmSubLayerAdd0);
   }

   result = FwpmTransactionCommit0(engine);
   EXIT_ON_ERROR(FwpmTransactionCommit0);

CLEANUP:
   FwpmEngineClose0(engine);
   return result;
}
```

Intercepting App/User Traffic via WFP

```c
#include <windows.h>
#include <fwpmu.h>
#include <accctrl.h>
#include <aclapi.h>
#include <stdio.h>

#pragma comment (lib, "fwpuclnt.lib")
#pragma comment (lib, "advapi32.lib")

#define SESSION_NAME L"SDK Examples"

#define EXIT_ON_ERROR(fnName) \
   if (result != ERROR_SUCCESS) \
   { \
      printf(#fnName " = 0x%08X\n", result); \
      goto CLEANUP; \
   }

DWORD FilterByUserAndApp(
         __in HANDLE engine,
         __in PCWSTR filterName,
         __in_opt const GUID* providerKey,
         __in const GUID* layerKey,
         __in_opt const GUID* subLayerKey,
         __in_opt PCWSTR userName,
         __in_opt PCWSTR appPath,
         __in FWP_ACTION_TYPE actionType,
         __out_opt UINT64* filterId
         )
{
   DWORD result = ERROR_SUCCESS;
   FWPM_FILTER_CONDITION0 conds[2];
   UINT32 numConds = 0;
   EXPLICIT_ACCESS_W access;
   ULONG sdLen;
   PSECURITY_DESCRIPTOR sd = NULL;
   FWP_BYTE_BLOB sdBlob, *appBlob = NULL;
   FWPM_FILTER0 filter;

   if (userName != NULL)
   {
      BuildExplicitAccessWithNameW(
         &access,
         (PWSTR)userName,
         FWP_ACTRL_MATCH_FILTER,
         GRANT_ACCESS,
         0
         );

      result = BuildSecurityDescriptorW(
                  NULL,
                  NULL,
                  1,
                  &access,
                  0,
                  NULL,
                  NULL,
                  &sdLen,
                  &sd
                  );
      EXIT_ON_ERROR(BuildSecurityDescriptorW);

      sdBlob.size = sdLen;
      sdBlob.data = (UINT8*)sd;

      conds[numConds].fieldKey = FWPM_CONDITION_ALE_USER_ID;
      conds[numConds].matchType = FWP_MATCH_EQUAL;
      conds[numConds].conditionValue.type = FWP_SECURITY_DESCRIPTOR_TYPE;
      conds[numConds].conditionValue.sd = &sdBlob;
      ++numConds;
   }

   if (appPath != NULL)
   {
      result = FwpmGetAppIdFromFileName0(appPath, &appBlob);
      EXIT_ON_ERROR(FwpmGetAppIdFromFileName0);

      conds[numConds].fieldKey = FWPM_CONDITION_ALE_APP_ID;
      conds[numConds].matchType = FWP_MATCH_EQUAL;
      conds[numConds].conditionValue.type = FWP_BYTE_BLOB_TYPE;
      conds[numConds].conditionValue.byteBlob = appBlob;
      ++numConds;
   }

   memset(&filter, 0, sizeof(filter));
   filter.displayData.name = (PWSTR)filterName;
   filter.providerKey = (GUID*)providerKey;
   filter.layerKey = *layerKey;
   if (subLayerKey != NULL)
   {
      filter.subLayerKey = *subLayerKey;
   }
   filter.numFilterConditions = numConds;
   if (numConds > 0)
   {
      filter.filterCondition = conds;
   }
   filter.action.type = actionType;

   result = FwpmFilterAdd0(engine, &filter, NULL, filterId);
   EXIT_ON_ERROR(FwpmFilterAdd0);

CLEANUP:
   FwpmFreeMemory0((void**)&appBlob);
   LocalFree(sd);
   return result;
}
```

Results

- Abomination of OpenVPN source-codes

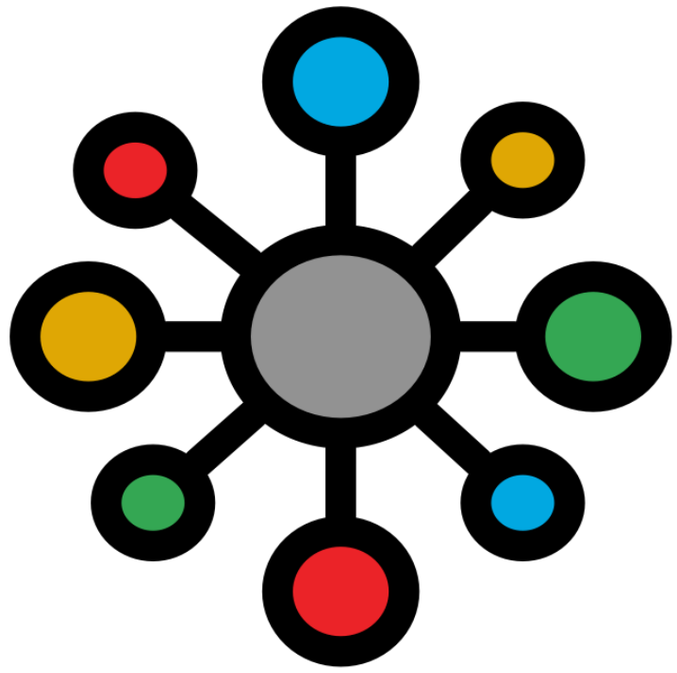


Team Up

- For now, write me email veselyv@fit.vutbr.cz

- Later, be ready for
  - GitHub repo
  - Mailing-list
  - MatterMost/Slack channel
  - *Grant proposal ???*

Thank you!

NES  FIT

http://www.fit.vutbr.cz/research/groups/nes@fit/