Debugging Tool - x64dbg

Azlan Mukhtar
CYSECA Solutions Sdn Bhd

About Debugging

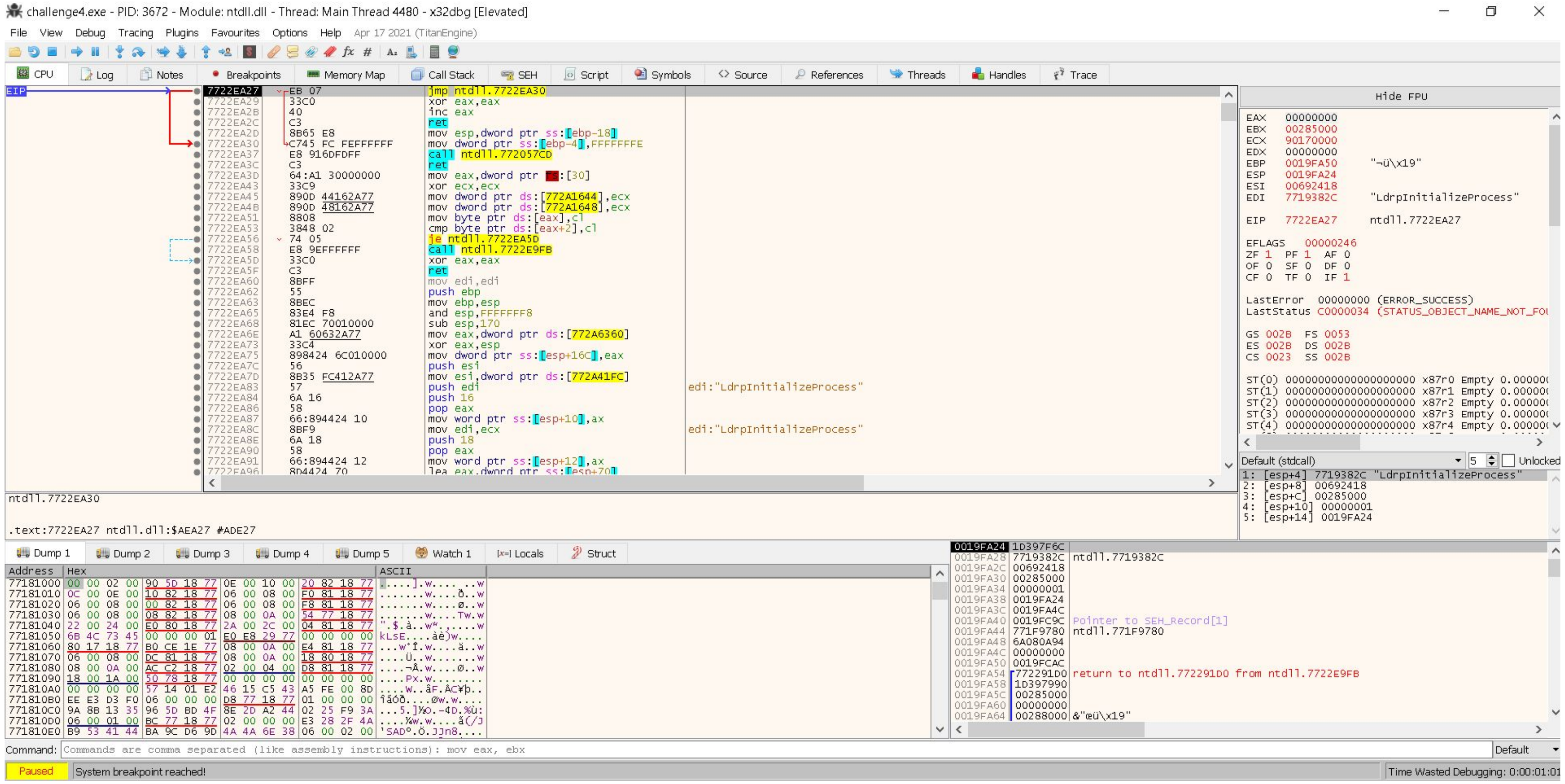
- From Wikipedia
    - Debugging is a methodical process of finding and reducing the number of bugs, or defects (…)
    - Debugger is a computer program that is used to test and debug other programs.
- Programmers usually debug their compiled program with debugging info (PDB file)
- In malware analysis, debugging has nothing to do with finding bugs.
    - We want to analyze the behaviors of unknown programs.
    - Malware author won’t distribute malware with debugging info, will be a bit harder

Why x64dbg

- The best user-mode (ring-3) debugger
- Open source
- Very powerful and user-friendly
- Can be extended with plugins
- **Suitable to debug program without source code/debugging info**

x64dbg Windows

- CPU
- Registers
- Executables modules
- Stacks
- Memory dump
- Other windows
    - Strings
    - Intermodular Calls
    - Memory Map
    - Threads
    - Breakpoints



```
7722EA27  EB 07              jmp ntdll.7722EA30
7722EA29  33C0               xor eax,eax
7722EA2B  40                 inc eax
7722EA2C  C3                 ret
7722EA2D  8B65 E8            mov esp,dword ptr ss:[ebp-18]
7722EA30  C745 FC FEFFFFFF   mov dword ptr ss:[ebp-4],FFFFFFFE
7722EA37  E8 916DFDFF        call ntdll.772057CD
7722EA3C  C3                 ret
7722EA3D  64:A1 30000000     mov eax,dword ptr fs:[30]
7722EA43  33C9               xor ecx,ecx
7722EA45  890D 44162A77      mov dword ptr ds:[772A1644],ecx
7722EA4B  890D 48162A77      mov dword ptr ds:[772A1648],ecx
7722EA51  8808               mov byte ptr ds:[eax],cl
7722EA53  3848 02            cmp byte ptr ds:[eax+2],cl
7722EA56  74 05              je ntdll.7722EA5D
7722EA58  E8 9EFFFFFF        call ntdll.7722E9FB
7722EA5D  33C0               xor eax,eax
7722EA5F  C3                 ret
7722EA60  8BFF               mov edi,edi
7722EA62  55                 push ebp
7722EA63  8BEC               mov ebp,esp
7722EA65  83E4 F8            and esp,FFFFFFF8
7722EA68  81EC 70010000      sub esp,170
7722EA6E  A1 60632A77        mov eax,dword ptr ds:[772A6360]
7722EA73  33C4               xor eax,esp
7722EA75  898424 6C010000    mov dword ptr ss:[esp+16C],eax
7722EA7C  56                 push esi
7722EA7D  8B35 FC412A77      mov esi,dword ptr ds:[772A41FC]
7722EA83  57                 push edi                               edi:"LdrpInitializeProcess"
7722EA84  6A 16              push 16
7722EA86  58                 pop eax
7722EA87  66:894424 10       mov word ptr ss:[esp+10],ax
7722EA8C  8BF9               mov edi,ecx                            edi:"LdrpInitializeProcess"
7722EA8E  6A 18              push 18
7722EA90  58                 pop eax
7722EA91  66:894424 12       mov word ptr ss:[esp+12],ax
7722EA96  8D4424 70          lea eax,dword ptr ss:[esp+70]
```

```
ntdll.7722EA30

.text:7722EA27 ntdll.dll:$AEA27 #ADE27
```

```
EAX   00000000
EBX   00285000
ECX   90170000
EDX   00000000
EBP   0019FA50     "¬ü\x19"
ESP   0019FA24
ESI   00692418
EDI   7719382C     "LdrpInitializeProcess"

EIP   7722EA27     ntdll.7722EA27

EFLAGS   00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOU

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 00000000000000000000 x87r0 Empty 0.00000
ST(1) 00000000000000000000 x87r1 Empty 0.00000
ST(2) 00000000000000000000 x87r2 Empty 0.00000
ST(3) 00000000000000000000 x87r3 Empty 0.00000
ST(4) 00000000000000000000 x87r4 Empty 0.00000
```

```
1: [esp+4] 7719382C "LdrpInitializeProcess"
2: [esp+8] 00692418
3: [esp+C] 00285000
4: [esp+10] 00000001
5: [esp+14] 0019FA24
```

```
Address  Hex                                              ASCII
77181000 00 00 02 00 90 5D 18 77 0E 00 10 00 20 82 18 77  .....].w.... ..w
77181010 0C 00 0E 00 10 82 18 77 06 00 08 00 F0 81 18 77  .......w....ð..w
77181020 06 00 08 00 00 82 18 77 06 00 08 00 F8 81 18 77  .......w....ø..w
77181030 06 00 08 00 08 82 18 77 08 00 0A 00 54 77 18 77  .......w....Tw.w
77181040 22 00 24 00 E0 80 18 77 2A 00 2C 00 04 81 18 77  ".$.à..w*.,....w
77181050 6B 4C 73 45 00 00 00 01 E0 E8 29 77 00 00 00 00  kLsE....àè)w....
77181060 80 17 18 77 B0 CE 1E 77 08 00 0A 00 E4 81 18 77  ...w°Î.w....ä..w
77181070 06 00 08 00 DC 81 18 77 08 00 0A 00 18 80 18 77  ....Ü..w.......w
77181080 08 00 0A 00 AC C2 18 77 02 00 04 00 D8 81 18 77  ....¬Â.w....Ø..w
77181090 18 00 1A 00 50 78 18 77 00 00 00 00 00 00 00 00  ....Px.w........
771810A0 00 00 00 00 57 14 01 E2 46 15 C5 43 A5 FE 00 8D  ....W..âF.ÅC¥þ..
771810B0 EE E3 D3 F0 06 00 00 00 D8 77 18 77 01 00 00 00  îãÓð....Øw.w....
771810C0 9A 8B 13 35 96 5D BD 4F 8E 2D A2 44 02 25 F9 3A  ...5.]½O.-¢D.%ù:
771810D0 06 00 01 00 BC 77 18 77 02 00 00 00 E3 28 2F 4A  ....¼w.w....ã(/J
771810E0 B9 53 41 44 BA 9C D6 9D 4A 4A 6E 38 06 00 02 00  ¹SADº.Ö.JJn8....
```

```
0019FA24 1D397F6C
0019FA28 7719382C ntdll.7719382C
0019FA2C 00692418
0019FA30 00285000
0019FA34 00000001
0019FA38 0019FA24
0019FA3C 0019FA4C
0019FA40 0019FC9C Pointer to SEH_Record[1]
0019FA44 771F9780 ntdll.771F9780
0019FA48 6A080A94
0019FA4C 00000000
0019FA50 0019FCAC
0019FA54 772291D0 return to ntdll.772291D0 from ntdll.7722E9FB
0019FA58 1D397990
0019FA5C 00285000
0019FA60 00000000
0019FA64 00288000 &"œü\x19"
```

```
Command: Commands are comma separated (like assembly instructions): mov eax, ebx
Paused  System breakpoint reached!  Time Wasted Debugging: 0:00:01:01
```

Debugging In General

- Start debugging by creating a new process or attach to an existing one
- Step or trace through code
- Set breakpoints
- Read & write memory
- Read & write registers and flags
- View the call stack
- View a disassembly of the code

Basic Debugging and Breakpoints

- Single Stepping
- Step into, step over, animate, run
- Run trace

- Software Breakpoint (INT3)
- Hardware Breakpoint
- Memory Breakpoint

Single stepping

- Single stepping means executing the application one instruction at a time
  - A very typical debugger feature
- Implemented by using EFLAGS.TF (Trace Flag)
- When TF=1, the processor generates a debug exception for the next execution

Software Breakpoints (BP)

- Used to break the execution of the debuggee at a specific address.
- Typically implemented using INT3 (0xCC)
    - Usually transparent, modification is not visible in memory view.
- Good thing
    - No limitation to the amount of software BP
- Bad
    - Modifies the actual code bytes
    - Cannot monitor reads or writes, just execution

Hardware Breakpoints

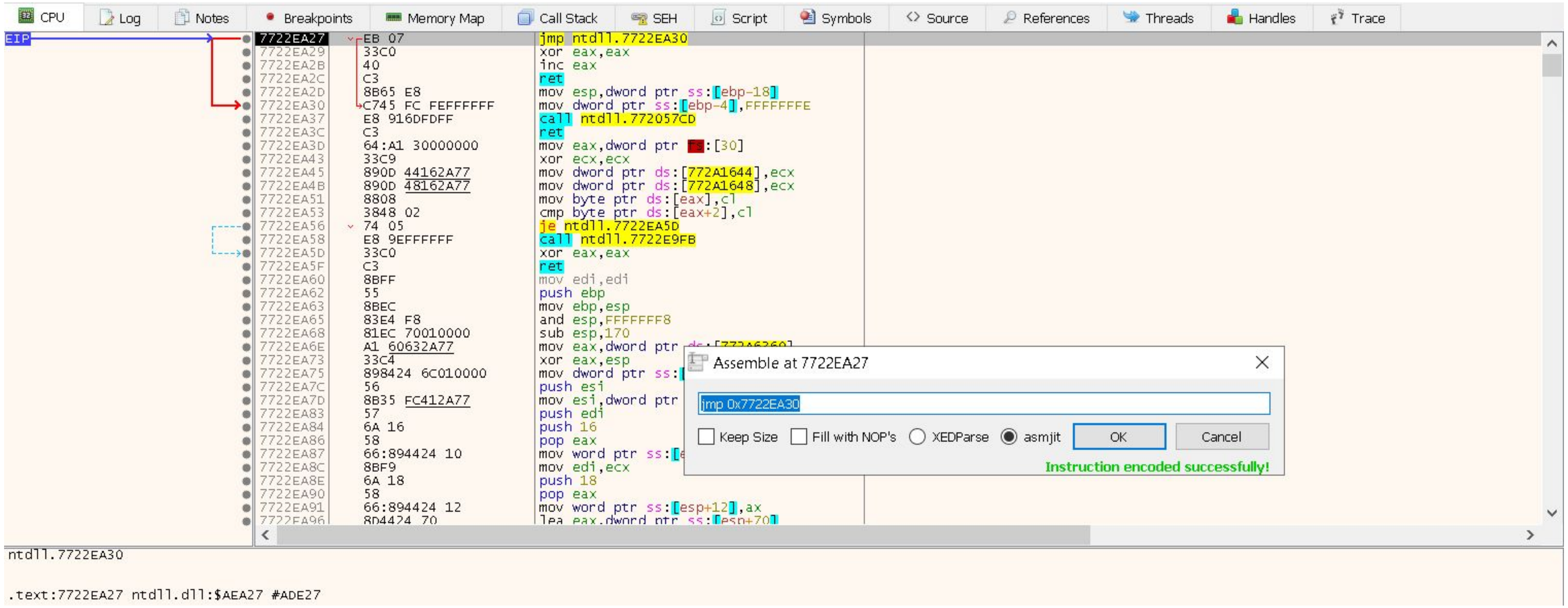
- CPU debug registers provide support for up to 4 hardware Breakpoints
- DR0-3 store the linear addresses to be monitored
- DR7 configures the type of event
  - Break on execution, break on read, break on RW
  - Length of data item to be monitored (1, 2 or 4 bytes)

Initial Breakpoint

- First time the debugger gets control of the target
- x64dbg has many options for initial BP, the most commonly used ones are
  - System BP
    - Loader breaks into debugger before any application code is run
  - Entrypoint of main module
    - First break is at the entrypoint as defined by the main module PE header

x64dbg as Assembler

Press Space to Assemble



```
7722EA27   EB 07              jmp ntdll.7722EA30
7722EA29   33C0               xor eax,eax
7722EA2B   40                 inc eax
7722EA2C   C3                 ret
7722EA2D   8B65 E8            mov esp,dword ptr ss:[ebp-18]
7722EA30   C745 FC FEFFFFFF   mov dword ptr ss:[ebp-4],FFFFFFFE
7722EA37   E8 916DFDFF        call ntdll.772057CD
7722EA3C   C3                 ret
7722EA3D   64:A1 30000000     mov eax,dword ptr fs:[30]
7722EA43   33C9               xor ecx,ecx
7722EA45   890D 44162A77      mov dword ptr ds:[772A1644],ecx
7722EA4B   890D 48162A77      mov dword ptr ds:[772A1648],ecx
7722EA51   8808               mov byte ptr ds:[eax],cl
7722EA53   3848 02            cmp byte ptr ds:[eax+2],cl
7722EA56   74 05              je ntdll.7722EA5D
7722EA58   E8 9EFFFFFF        call ntdll.7722E9FB
7722EA5D   33C0               xor eax,eax
7722EA5F   C3                 ret
7722EA60   8BFF               mov edi,edi
7722EA62   55                 push ebp
7722EA63   8BEC               mov ebp,esp
7722EA65   83E4 F8            and esp,FFFFFFF8
7722EA68   81EC 70010000      sub esp,170
7722EA6E   A1 60632A77        mov eax,dword ptr ds:[772A6360]
7722EA73   33C4               xor eax,esp
7722EA75   898424 6C010000    mov dword ptr ss:[
7722EA7C   56                 push esi
7722EA7D   8B35 FC412A77      mov esi,dword ptr
7722EA83   57                 push edi
7722EA84   6A 16              push 16
7722EA86   58                 pop eax
7722EA87   66:894424 10       mov word ptr ss:[e
7722EA8C   8BF9               mov edi,ecx
7722EA8E   6A 18              push 18
7722EA90   58                 pop eax
7722EA91   66:894424 12       mov word ptr ss:[esp+12],ax
7722EA96   8D4424 70          lea eax,dword ptr ss:[esp+70]
```

Assemble at 7722EA27

jmp 0x7722EA30

□ Keep Size □ Fill with NOP's ○ XEDParse ● asmjit OK Cancel

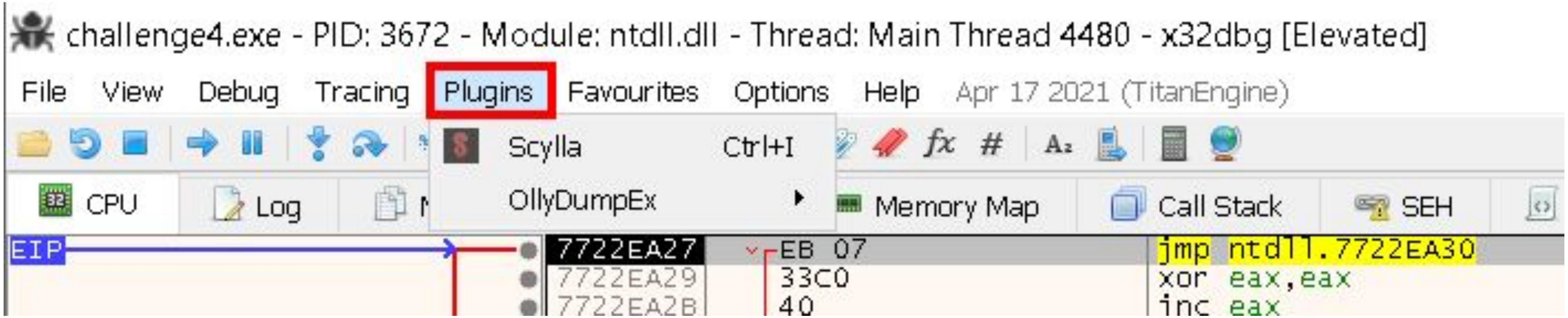
Instruction encoded successfully!

```
ntdll.7722EA30

.text:7722EA27 ntdll.dll:$AEA27 #ADE27
```

Useful plugins

https://github.com/x64dbg/x64dbg/wiki/Plugins


challenge4.exe - PID: 3672 - Module: ntdll.dll - Thread: Main Thread 4480 - x32dbg [Elevated]
File
View
Debug
Tracing
Plugins
Favourites
Options
Help
Apr 17 2021 (TitanEngine)
Scylla
Ctrl+I
OllyDumpEx
CPU
Log
Memory Map
Call Stack
SEH
EIP

```
7722EA27    EB 07    jmp ntdll.7722EA30
7722EA29    33C0     xor eax,eax
7722EA2B    40       inc eax
```

Useful shortcut keys

- Run(**F9**): This starts or resumes the process normally.
- Pause(**F12**): This suspends the current process.
- Restart(**Ctrl+F2**): This terminates the debugged process and reloads it.
- Close(**Alt+F4**): This terminates and unloads the debugged process.
- Step Into(**F7**): This allows us to enter a routine or execute the next step in a repeat instruction.
- Step Over(**F8**): This allows you to execute an entire subroutine or repeat instruction without stepping through it instruction by instruction.

…

Note on Debugging and security

Warning!!

- Debugging involves EXECUTING unknown code
- Even if you are doing careful, there is a good chance your debuggee will go through and start running.
  - The malware may spreading into the network
  - You may leak something (Key, password, data)
- Always debug in non-production environment (dedicated machine on separate network or virtualized environment)

Demo, Q&A, and Exercise

Further readings

- x64dbg
  - http://x64dbg.com/
  - https://github.com/x64dbg
- Documentation
  - https://help.x64dbg.com/en/latest/index.html