

Microsoft .NET

Source: <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2004-02/1052.html>

From: Novice (6tc1_at_qlink.queensu.ca)

Date: 02/11/04

Date: 11 Feb 2004 11:35:45 -0800

Hi all, I am a C++ and Java developer with over 3 years of industry experience. I've written low level C++ code, in addition to web clients that use web services. I've just recently installed the Visual Studio .net Professional trial version 2003. I have been reading up various documents that discuss - "What is Microsoft .Net" and have found some enlightening information.

I'm trying to write a paper on security and software development using Microsoft .Net. So far the most difficult aspect has been determining - what is Microsoft .net/.NET/.Net? On the microsoft website they define it as:

<MICROSOFT_ARTICLE>

Microsoft® .NET is a set of software technologies for connecting information, people, systems, and devices. This new generation of technology is based on Web services—small building-block applications that can connect to each other as well as to other, larger applications over the Internet.

</MICROSOFT_ARTICLE>

Then they go on to say that:

<MICROSOFT_ARTICLE>

The Components of Microsoft .NET-Connected Software
.NET is infused into the products that make up the Microsoft platform, providing the ability to quickly and reliably build, host, deploy, and utilize connected solutions using Web services, all with the protection of industry-standard security technologies.

"Smart" client application software and operating systems enable PCs and other smart computing devices to act on Web services, allowing anywhere, anytime access to information.

Smart client software and .NET

Smart devices and .NET

Microsoft and others are developing a core set Web services—from authentication to calendaring—that can be combined with other Web services or used directly with smart client applications. The Microsoft MapPoint® Web Service allows you to integrate high-quality

maps, driving directions, and other location intelligence into your applications, business processes, and websites is an example of one of these services.

What are Web services?

Microsoft MapPoint Web Service

Microsoft provides the best server infrastructure—the Microsoft Windows Server System™—for deploying, managing, and orchestrating Web services.

Microsoft servers and .NET

Microsoft Visual Studio® .NET and the Microsoft .NET Framework are a complete solution for developers to build, deploy, and run Web services.

Developer tools and .NET

.NET Experiences

Building solutions with .NET technologies, you can create and connect to an infinite variety of personalized .NET experiences, with industry-standard technologies helping to protect your security and safety. Individuals can enjoy rich, tailored interactions—.NET experiences—when Web services are pulled together, allowing access to information across the Internet and from stand-alone applications, online or offline.

</MICROSOFT_ARTICLE>

I went on reading on the Microsoft website (training, etc) and when I looked at available .net training many of those courses were teaching Visual Basic .NET and ADO.NET and ASP.NET. However, the Visual Studio .net Professional application allows the developer to write C++, C#, J# applications all of which are supposedly transformed into MSIL (intermediate language) and then Just-In-Time compiled using CLR into native code. Does this mean that all of these languages, provided they are compiled into MSIL and then into native code using CLR, are part of .net?

If that is the case, then if I can find a security issue (i.e. one of the functions like printf in iostream provided by Visual Studio .NET has a security flaw) in a C++ program (compiled using Visual Studio .net) then does that mean that I have found a security flaw in .net or is it just a security flaw in Microsoft's implementation of the printf function? I.E. are they one and the same?

When I read this one .NET security document "Security in the Microsoft .NET Environment" it indicated the framework had a variety of security mechanisms:

Evidence-Based Security, Code Access Security, Role-Based Security, etc. Are these libraries available in all of the languages I mentioned like C++? The document even went on to show a line of code (from I assume ASP.NET):

```
if (HttpContext.IsCallerInRole("Admin")){...}
```

The best information I have found thus far was in a posting on another newsgroup – here is what the poster said:

<NEWSGROUP_POSTING>

To me, .NET is Microsoft's implementation of an enterprise level software development platform. Enterprise software development has changed a lot over the last five years, and COM, COM+ and DCOM just weren't cutting it any more on the level of security, scalability, and uniformity. In addition, Microsoft's previous software development environments all had their own way of doing things. If you're a Visual Basic developer, you develop software in a completely different way than, say, a C++ guy. Different programming language, of course, but also: different paradigm (object oriented vs. procedural), different development tools (Visual Studio vs. Visual Basic IDE), different capabilities (VB is much more limited in what you can do than C++), different data types (an integer in C++ is 32 bits, in Visual Basic it's 16 bits), different execution model (compiled vs. semi-interpreted) etc. There was no such thing as a "Microsoft Developer".

So Microsoft thought up .NET. At the lowest of .NET level sits something called the "Common Language Runtime", a sort of low-level set of functionalities that every .NET program can (and must) use, and that covers the entire width and breadth of software development on the Windows platform. Consider this the .NET alternative to the Win32 API.

On top of that, a number of programming languages with their respective compilers are available. The most visible ones are Visual Basic.NET, C# (see-sharp), and C++, but there are more than 20 others, like J# (which uses the Java programming language syntax), COBOL, Pascal, Haskell, etc. All of these languages must follow a number of rules, of which the most important ones are that they must support the Object Oriented development paradigm, and all of the standard .NET datatypes.

All .NET language compilers produce a sort of "intermediate code", called

MSIL (Microsoft Intermediary Language) or simply IL. This is what is stored in .EXE and .DLL files. When a .NET program is executed, the IL is "just-in-time-compiled" into processor instructions and run in a sort of "sandbox", called a "managed" environment. The managed environment takes care of (amongst other things) memory management (no more memory leaks!), data type checking, and security (access to resources is controlled, depending on where the program comes from: e.g.: a downloaded program by default cannot access the local harddisk).

There is only one common development environment for .NET, called Visual Studio .NET. Developers can use it with any .NET programming language they desire, and develop any type of .NET application with it.

.NET applications can be largely split up in three types: classical Windows form-based applications that are executed locally on a PC, web applications that have a browser-based user interface (these are executed using ASP.NET), and server applications that run on a server, and can be accessed by clients over the network.

Access to server apps can be done using proprietary Microsoft protocols, but also using something called "Web Services". Web services can be regarded as application components that are made accessible using industry-standard techniques like XML and SOAP. Web services can be accessed over the internet, through firewalls (something that was extremely difficult to do using COM), from and to different platforms. The techniques behind web services are based on open standards and are widely supported by most software vendors, including Microsoft's "opponents" like Sun, Oracle, etc. Using Visual Studio.NET, developers can produce Web Services that can be used by other systems, or they can themselves use web services produced by others (even if those web services run on totally different hard- and software environments). Web Services are completely hard- and software independent.

Over time, Microsoft will make the .NET environment a standard part of all versions of Windows and Windows Server products. They will incorporate .NET into the core of their servers. As an example: in the next version of SQL Server, developers will be able to write .NET programs that run "inside" the database (like stored procedures), and server products will make their functionalities available by exposing a .NET programming interface.

And to make all of this relevant to this newsgroup: there is a version of the .NET environment for Windows CE (it's still in beta currently, but will be available soon). This allows developers to write applications for all Windows CE-based devices. Due to the small footprint requirements of Windows CE devices, it has some limitations, but these are very minor and concern things that you would not need on a handheld device anyway. As a developer, there is no difference real between developing apps for the PocketPC and developing apps for a Windows PC: same programming languages, same tools, same object models. The only thing you need to take care of is the limited screen and keyboard functionality of handheld devices.

Summing up: .NET is a platform for developing and executing software, that spans (or at least: *will* span) all Windows platforms, and offers a uniform, scalable, and secure environment to run software on. In addition, it offers support for enterprise application integration (EAI) based on open standards. It will be the standard and default Windows software platform for the future.

</NEWSGROUP_POSTING>

In summary, I want to clearly understand where .NET begins and ends. I want to know if I find a security related problem in some of the supporting libraries (i.e. iostream) of C++ that are found within Visual Studio .net have I found a .NET security flaw or is it just a security flaw in Microsoft's implementation of the iostream library for C++? Is the security framework I mentioned earlier available to all of the previously mentioned languages like C#, J#, ASP.NET, etc in some form or another (due to their compilation to MSIL and subsequent

JIT compile to native code)? Also two additional questions:

1. If I compile a C++ program using Visual Studio .NET will that executable require the .NET framework to be installed? I.E. if I send it to my friend on a different computer (same OS), will he require the .NET framework to execute it.
2. Also, has that executable been generated using the JIT compiler in CLR?

Thanks for any advice in this regard,
Novice