Security vulnerabilities of the top ten programming languages:
C, Java, C++, Objective-C, C#, PHP, Visual Basic, Python, Perl, and Ruby

Stephen Turner

Known-Quantity.com, part of Turner & Associates, Inc.

ABSTRACT

Programming languages are like genetics, in that there are a few ancestors with common traits that have proliferated. These can be traced back over time. This paper will explore the common traits between the top ten programming languages, which runs almost 80 percent of all of software used today. These common traits are both good and bad. In programming, the bad traits equate to security vulnerabilities that are often exploited by hackers. Many programmers are not aware of these flaws and do not take the time to take corrective action as they build software applications. They cannot simply fix the problems at the end; they must continually adjust how they program throughout the process. This paper will also provide guidance on what can be done to make computing environments more secure.

Chief information officers (CIOs) have a responsibility to oversee all aspects of software development, and should consider assigning a project manager familiar with the security challenges inherent with a particular programming language. A CIO should also sign-off at every stage, starting with system conceptualization, system requirements of the system, analysis of benefits, and scope of project. The business and system analyses are critical and must include security parameters for programmers in software requirement specifications. Once the planning and design stages are approved, unit development, software and system integration, followed by testing and retesting, are essential. All of this precedes installation, site testing and acceptance, training/documentation, implementation and maintenance.

Keywords: Software, Computer Programming, Applications, Security, System Analysis, CIO

INTRODUCTION

The National Vulnerability Database is a comprehensive website that allows risk managers and security professionals to track security problems, and rate the level of risk. There are 56,009 Common Vulnerabilities and Exposures (CVE) listed, and 2,708 US-CERT vulnerability notes, along with 245 US-CERT alerts (DHS National Cyber Security Division/US-CERT, 2013). Only a fraction of known problems are listed in CVE.

According to Robert Seacord, 100,000 software vulnerabilities are identified in a given year, and 400,000 incidents occur during that same timeframe (Seacord, 2009). Seacord leads the secure coding initiative at the Software Engineering Institute. It is very difficult to keep up with so many threats, but there are some common themes that emerge again and again. Most problems are related to buffer overflows, un-validated input, race conditions, access-control problems, authentication or authorization weaknesses, or cryptographic practices (Mac Developer Library, 2013). This paper will analyze common errors that can be corrected with proper development processes.

Over the past twenty-five years, some languages have been consistently popular among programmers, such as C, C++, Visual Basic, Perl, Lisp and Ada. According to the TIOBE Programming Community Index (TIOBE) (2013), other languages have grown in usage very rapidly, such as PHP and Ruby. Objective-C was ranked 45th most used in 2008, and it is now third. Java first appeared in the fourth position in 1998 and was number one by 2008.

THE TOP TEN LANGUAGES

Number One: Java

Java remains most used in 2013; its praxis grew by 1.34 percent from the previous year, and 18.39 percent of lines of code are developed using Java (TIOBE Software, 2013). Naturally, the most popular language is subject to more scrutiny than others. Fred Long asserts that Java is secure if used properly, but engineers can misuse it or improperly implement it. Java is a multi-threaded programming language built on the type-safe model, which does not permit a buffer's boundary to be superseded. Hoglund and McGraw (2004) assert that buffer overflows – in the classic sense – do not happen in Java because "falling off the end of an object and spilling elsewhere is not possible." However, external code that interacts with Java – such as code written in C – is an exception (Hoglund & McGraw, 2004).

Attackers understand that internal buffer overflow attacks are not easy, but have demonstrated that a type confusion attack could exploit a weakness by using a public field like a private field – as early as Java 5. This made Java's Security Manager accessible to manipulation or foul play (Long, 2005). This is also known as a language-based attack.

CERT has warned of log injection attacks in Java – often through web browsers. These attacks can be averted through validation, or authentication of submitted input. For example, attackers have manipulated Java by entering a Carriage Return and Line Feed (CRLF) sequence, which often deceives system administrators. Sanitizing the data assures that it matches the requirements of each field and keeps administrators alert to un-validated input (Mohrindra & Jumde, 2012). Unchecked input can include Structured Query Language (SQL) injection in Java (Livshits & Lam, 2013).

Because Java is a multi-threaded programming language, it does enable deadlock

and race conditions (Java Revisited, 2012). Deadlock occurs when two or more threads are waiting for each other, which can permanently obstruct progress (Java Tutorials, 2013). This is a bug, but not as much of a liability as are race conditions, which occur when synchronization has not been properly programmed. For example, *check and act* should start with a *check* thread and finish with an *act* thread. It is not uncommon that a null value is recorded when there is a delay that was not anticipated by the programmer (Java Revisited, 2012). Two or more threads can share physical memory as directed by Java. If the result of a thread is recorded before another thread, out of sequence, an attacker can manipulate the system in a way not anticipated by programmers (IBM Developer Works, 2004). One method is for a malicious user to claim to be a trusted user, and the system can be tricked into authentication. As a result, information submitted to logs could be accepted without being validated (OWASP, 2009).

Trust exploits follow access-control vulnerabilities. Code-signing errors occur when there is no valid certificate that matches a user profile (Web Builders, 2013). Developer Richard Dallaway experienced many problems that were only resolved by obtaining digital certificates: "I needed to hand over money to a certificate authority so they could run some background checks on me to prove that I am who I say I am" (Dallaway, 2013). Essentially, it is possible to get a certificate from authorities that are less stringent, leaving Oracle Java 7 wide open for exploitation on Windows, OS X, and Linux platforms. CERT found that "a remote attacker may be able to execute arbitrary code on a vulnerable system" (CERT/Software Engineering Institute/Carnegie Mellon, 2013).

Java patches have been frequently introduced to fix bugs after exploits expose vulnerabilities. H.D. Moore, CSO of Rapid7, indicated that Update 11 for Java 7 was not sufficient, and users should not keep Java on their desktop, nor rely on it to surf the web. He suggests that it will probably take two years to correct these vulnerabilities (Smith, 2013). Despite these challenges, Java is used on the Android platform, so it is likely to have staying power (Taft, 2012).

Number Two: C

Java and C are the only two programming languages in which more than 10 percent of code is used to write new software. In February 2013, TIOBE found that the usage of C grew by only 0.56 percent from the year before, and 17.080 percent of code is written in C (TIOBE Software, 2013). Vulnerabilities with the C programming language have been known for some time, so they are less shocking than recent warnings about Java. C is very influential, so you will see references back to C in other languages, such as Perl (number nine) and Ruby (number ten). The relationship with C++, C#, and Objective-C are more obvious. C's vulnerabilities have rippled throughout software programming for decades.

C traces back to Bell Labs in the early 1970s before multithreading was feasible on hardware. It's a procedural language that has evolved over the decades as hardware has become more sophisticated (University of Michigan, 2013). This evolution has triggered race conditions, which has caused significant functionality problems and enabled some exploits.

Java is object-oriented, while C is function-oriented. C's basic programming unit is a function (Princeton University, 2013). C is vulnerable to code injection attacks. It allocates memory with local variables in a function (automatic), global variables (static) and malloc

(dynamic). Some programmers do not know that they are responsible for allocating and de-allocating memory, including bounds and type checks. Compilers – which transform code into an executable program – do not detect common errors in C at run-time. Errors include pointers to de-allocated memory, reserving memory indefinitely after it is needed, and writing past bounds of allocated memory (Younan, 2013).

C is much more vulnerable to buffer overflows than Java is. According to James Joyce, "C includes no way of telling when the end of an array or allocated block of memory is overrun. The only way of telling is to run, test, and wait for a segfault. A spectacular crash is also a telltale sign. Or a slow, steady leakage of memory from a program is a symptom of a buffer overflow in C" (2012). He suggests that C's greatest weakness is that it does not keep track of the ends of strings (Joyce, 2012). Null-termination errors, string truncation, and unbounded string copies create vulnerabilities (Seacord, 2005). Unlike Java, C lacks type safety. C has become more prone to errors in recent years because data across the web is exchanged between programs using strings (Seacord, 2005).

An array is a string with a series of characters. Programmers are limited with arrays in C, and have to go to great lengths to correct this limitation. A hacker familiar with this limitation can use a web field to launch an attack. They intentionally exceed the fixed-length static array and can actually manipulate string operations to write outside the bounds of the statically allocated character array, which might be 80 characters. Therefore, un-validated input is a particular concern with strings in C. Programmers must take an extra step to test the length of the input and can then dynamically allocate memory (Seacord, 2005). This erects a barrier to an exploit based on a commonly known error in C. Programmers have access to libraries that enable them to validate characters submitted in C. For example, a zip code could be restricted to five numeric characters. Alphabetical characters could be rejected in a zip code or other numeric field, such as a phone number (C Programming, 2013).

Robert Seacord asserts that C "is intended to be a lightweight language with a small footprint" (2005). He added "this characteristic of C leads to vulnerabilities when programmers fail to implement the required logic, because they assume it is handled by C when it is not. This problem is magnified when programmers are already familiar with superficially similar languages such as Java, Pascal, or Ada, leading them to believe that C protects the programmer better than it actually does. These false assumptions have led to programmers failing to prevent writing beyond the boundaries of an array, failing to catch integer overflows and truncations, and calling functions with the wrong number of arguments" (Seacord, 2005).

Number Three: Objective-C

C is a procedural function-oriented language, while Objective-C is object-oriented. In some ways it is not a different language than C, but a series of extensions based on an early object-oriented language called Smalltalk (iOS Developer Library, 2010). TIOBE research found that usage of Objective-C grew more than any of the other top ten languages – by 2.74 percent from the year before. Currently 9.803 percent of programmers use Objective-C (TIOBE Software, 2013). To a large degree, this is because Apple's operating systems and application programming interfaces use Objective-C (Apple Developer, 2012). This has allowed developers of apps to tap into the vastly growing market of iPhone and iPad users.

Objective-C does address potential access-control problems through the use of objects that encapsulate data. Based on this approach, the optimal method is to show only a public interface that hides the internal operations. Therefore, a hacker has a far more difficult time gaining unauthorized access (iOS Developer Library, 2012).

Many hackers understand the limits of Objective-C, and enter malformed data to see what will happen (Secure Coding Guide, 2011). Un-validated input is a concern in Objective-C, which can enable code insertion vulnerabilities, format string vulnerabilities, and buffer overflows. Because of the addition of objects, developers have an advantage in reducing the risk of buffer overflows with Objective-C. *NSString* is an example of a Cocoa object that can manipulate strings (Programming4Us, 2011). Cocoa is a collection of APIs and runtime objects for Objective-C. According to mobile security expert Jeremy Allen, "All object allocations go on the heap, which helps prevent stack overflows since memory controlled by the coder does not live on the stack" (Allen, 2010).

Objective-C is a multi-threaded programming language that creates race conditions. These race conditions can be exploited. "Time of check-time of use" exposes a momentary gap. This is between the time a file or string is located and the time it is accessed. The attacker can then place a bad file in its place, and the software can write to it (Mac Developer Library, 2012).

Attackers can also exploit signal handing vulnerabilities. Objective-C's signal handlers can execute code at random intervals. Even though one action may be expected to occur at a moment in time, if the signal handler sets a string in motion, it can create a race condition. Objective-C programmers should take extra effort not to use signal handlers (Mac Developer Library, 2012).

Number Four: C++

Objective-C and C++ are both based on C and are object-oriented, but both have very different syntax. Because of the complexity and precision of the syntax, Linus Torvalds said that C++ is a "horrible language" (Harmful Stuff, 2007). Indian Institute of Science assistant professor Gaurav Tomar explains that "C++ [was] designed for general object oriented-programming in the days when the typical computer was a standalone machine running a command line-based user interface" (2008).

TIOBE found that C++ usage grew by only 0.91 percent from the year before, and 8.758 percent of programmers used it in February 2013 (TIOBE Software, 2013). C++ is a mid-level language and not as relevant to the mobile environment. Perhaps it is also less popular because it lacks runtime bindings, proxies, categories, and Interface Builder (Rutman, 2013).

Kragen Javier Sitaker is an Argentinian developer (Canonical, 2013). Sitaker compared the two languages and concluded they both "have a string class and containers in the standard library, and support for some automatic memory management in the language. This turns out to make a big difference in practice in reducing [those] vulnerabilities" (Y Combinator, 2013).

The C++ object library addresses buffer overflows. String library *std::string* combined with stream operators (>> and <<) reduces the likelihood of overflows. However, these can be compromised when a program uses C API functions instead of C++ (eTutorials, 2013).

Among other issues, shared variables can cause race conditions in C++ with conditions such as simple increments of variables, loop indices, and shared class objects. Sun Development Network author Phyllis Gustafson advises developers to pay particular attention to even the most innocuous variables with critical regions or atomic directives. Any modifications to variables should be carefully considered in advance, because pitfalls in C++ are unpredictable (2013).

C++ enforces access-control restrictions within thread safety attributes, which is not customary. These attributes are attached to methods or fields. These methods have many intricacies that are used a compile, and only the most experienced programmers are likely to understand how to make certain compromises to get around the limitations of the C++ model without creating significant vulnerabilities (Blaikie, 2012).

Matasano Security's founder observed, "C programs are susceptible to memory corruption. Programs written in practically every mainstream high level language are not susceptible to those problems (until they start using third-party C extensions). That's the security win of not using C code." Thomas H. Ptacek suggested, "One way to get C/C++ code fit into a web application is via 'nosql' databases, particularly Redis; let something like Ruby/Rack or Python/WSGI or Java Servlets soak up the hostile HTTP traffic, and use it to drive an async API over Redis. The Redis interface is also so simple that it's very easy to hook C code up to it, and Redis is somewhat 'typed,' which reduces the amount of parsing you have to do" (Y Combinator, 2013). Needless to say, C++ is filled with landmines that many programmers cannot see.

Number Five: C#

The fundamental operators and style of C++ are leveraged in C# (called "C Sharp"), but also add concepts of Visual Basic (see Number Eight). C# executes the code in a controlled sandbox called the virtual machine, which compels software to be type safe. Objective-C, C++, and C# are all object-oriented. All code in C# must be enclosed within an object (Wikibooks, 2013).

According to TIOBE, C# is ranked 5th and is used for development on Microsoft .NET. It is considered by some to be appropriate for Internet programming, but it is not exactly a key enabler of today's mobile apps (Godel, 2001). It's usage shrank by a whopping -1.97 percent from the year before, decreasing more than any of the other top ten languages. Approximately 6.680 percent of code uses C# (TIOBE Software, 2013).

C# is viewed by some as a significant improvement on C++, especially in the areas of versioning, events, and garbage collection. C# programmer Hafeez Mohammed explained that the "framework allows you to forget about memory management. It's incredibly scalable, and even brings an end to the infamous 'DLL Hell' by getting rid of globally unique identifiers (GUIDs), registration, and all that automatically." C# says goodbye to buffer overflows and significantly improves the state of race conditions (Mohammed, 2013). Yet, the runtime system's garbage collector in C# is not foolproof. However, in conjunction with destructors (*dtors*) and finalizers, it is sufficiently reliable to assume that programmers don't need to stress about freeing of memory resources (Mehra, 2009).

In the .NET framework, C# allows applications to be scanned for vulnerabilities (Mehra, 2009). This tool is good as a final check before an application goes live or ships, but developers need to deal with C# vulnerabilities as they work. SQL injection, packet-

sniffing, session hacking and cross-site scripting (XXS) problems are known weaknesses. All SQL commands should be replaced with parameterized queries or stored procedures to avoid SQL injection. Secure https should always be used to avert session hijacking and packet-sniffing. HTTP-only is the secure standard when setting cookies, in order to avoid cross-site scripting predicaments (Ideal Programmer, 2009). These are best practices, but it is essential that a programmer stay abreast of vulnerabilities as soon as they are made public.

When C#.NET vulnerabilities can be corrected, Microsoft releases patches on the first Tuesday of each month. This is called "Patch Tuesday." The following day is known as "Exploit Wednesday" (Peterson, 2009). David Aucsmith made a startling statement about patch releases: "We have never had vulnerabilities exploited before the patch was known." Aucsmith is Senior Director at the Microsoft Institute for Advanced Technology in Governments (Bradley, 2013). In essence, the release of patches is a clue to hackers as to where they can take advantage of vulnerabilities that have not been patched. C# is somewhat stable, and major new flaws are not frequently discovered, which is not true of PHP.

Number Six: PHP

.NET and PHP are the two leading development frameworks for dynamic web sites. Dynamic sites provide user experiences that vary from person to person, depending upon the choices of the user, or cookies that the person's browser contains. PHP is the foundation for popular web development applications such as Drupal, Joomla, and Wordpress. TIOBE concluded that PHP is the sixth language in terms of usage. It's popularity decreased by a modest -0.57 percent from the year before, and 5.074 percent of programmers used PHP as of February 2013 (TIOBE Software, 2013). A low level of security may be one reason for the decline.

Interestingly, the first version was a collection of Perl (see Number Nine) scripts in 1995, called *Personal Home Page* (PHP) tools. Hence the name PHP was born, but what it represented changed dramatically. In 1997, the next version was rewritten in C and was called PHP Form Interpreter. The third version emerged with APIs, databases and protocols and was called PHP Hypertext Preprocessor. This name has stuck, but the structure changed again in 2000. Version 4 was based on an open-source scripting engine called Zend Engine (Grubb, 2012).

Object-oriented PHP seemed like a real breakthrough in terms of ease of use for programmers, but in 2006 a major vulnerability was discovered with Zend Engine hashtables (Esser, 2013). Arrays in PHP are ordered hashtables (Swanson, 2013). Not surprisingly, hash indices in PHP are both alphabetical and numerical. The function *zend_hash_del_key_or_index* contained a flaw that inadvertently deleted a bucket-slot belonging to numerical keys, even though instructions may have called for the alphabetical bucket to be deleted. As a result, the wrong variables are deleted in PHP's *symboltable*, and the incorrect elements are removed from arrays (Esser, 2013). This vulnerability was fixed with a patch of PHP4 six months after it was discovered, but it was the first of many major problems. Over time, it has become apparent that PHP is terribly flawed. Approximately one-third of all the problems listed in the National Vulnerability Database are associated with PHP (DHS National Cyber Security Division/US-CERT, 2013).

The most recent patch of PHP that was released in 2013 is 5.4.12 (PHP Group, 2013). It includes fixes for Core, Date, FPM, Litespeed, sqllite3, PDO_OCI and PDO_sqllite (PHP Group, 2013). PHP does not have Patch Tuesday; it actually publishes a list of 372 categories of bugs and vulnerabilities, with a total 64,156 items. Anyone – including hackers – can go to this site and see 2080 bugs or vulnerabilities that just can't be fixed (PHP Group, 2013). Some are false reports, while others are minor and leave little for room for hackers to exploit. Some bugs are legacy problems that PHP Group no longer supports. However, as innocuous as some bugs might appear, bad actors could find ways to exploit a limitation. For example, "Mod doesn't return correct values" is a posting about a problem with the calculation of large numbers. Derick Rethans of PHP Group responded that "PHP only supports signed integers (range is -2147483648 to 2147483647 on Linux), so this is actually the expected behavior" (PHP Group, 2013). Therefore, attackers might be able to exploit this limitation by creating a calculation that would result in a number greater than 2,147,483,647. In large transactions, this might be a disaster.

"Beware: Some MySQL table types (storage engines) do not support transactions." This is a warning that appears in the PHP online manual. The manual continues: "When writing transactional database code using a table type that does not support transactions, MySQL will pretend that a transaction was initiated successfully. In addition, any data definition language (DDL) queries issued will implicitly commit any pending transactions" (PHP Group, 2013). That is almost unbelievable.

PHP is a scripting language, while MySQL is a relational database management system that both use the same server nearly 99 percent of the time (Yahoo Answers, 2012). They are connected with PHP Data Objects (PDO) (PHP Group, 2013). There are five dynamic MySQL options or variables that present security problems: *automatic_sp_privileges* (system variable)*, local_infile* (system variable)*, old_passwords* (system variable)*, safe-show-database* (option file and system variable) and *secure-auth* (option file and system variable) (MySQL, 2013). Server access is a huge issue that must be carefully controlled and monitored with PHP and MySQL.

There are several steps that can be taken to address PHP/MySQL vulnerabilities. Installation of patches as soon as they are released is essential. SQL injections are enabled by bugs in PHP code, and injections can be averted by disabling the use of *LOCAL INFILE*. Remote access should be restricted, or better yet, disabled. The command *bind-address=(IP address)* will cause the server to only respond to *localhost*. Accounts must be frequently reviewed. Obsolete and anonymous accounts should be deleted, while system and database privileges should be limited. Simple things, like changing the root username to another name, can make it much more difficult to for attackers to assume the root identity. The root directory can also be changed, so that hackers cannot run UNIX commands to manipulate the directory system. PHP does not automatically run logs of activity on the database. Logging needs to be enabled so that intrusions and errors can be tracked (Green SQL, 2013). Administrators must check these logs frequently in PHP because it is so deeply flawed in this regard.

Number Seven: Python

In contrast to PHP, the Python Security Response Team only shares vulnerabilities with what it describes as a "highly trusted cabal of Python developers" (Python, 2013). They

actually attempt to avoid the mistakes that lead to Exploit Wednesday.

The Django framework is based on Python, but it is not strictly a language used for web development. Many games are built with it (Fuecks, 2013). Python is a high level language that uses natural language elements, intended to be easy to learn, perhaps easier than PHP (Python, 2013). Y Combinator co-founder Paul Graham observed, "People don't learn Python because it will get them a job; they learn it because they genuinely like to program and aren't satisfied with the languages they already know." He asserted that the source code is more attractive and less complicated (Graham, 2004). This may account for its relatively robust growth. TIOBE found that Python grew significantly by 1.8 percent from the previous year, and 4.949 percent of programmers used Python as of February 2013 (TIOBE Software, 2013).

Python has different versions, such as CPython and IronPython. The Python Interpreter in CPython is based on C and can lead to buffer overflows. Other versions, such as IronPython and Jython, are based on Java and C# and have an added layer of protection (Velocity Reviews, 2013).

Concurrency is parallel processing. *Message passing* and *shared data* are two standard models of concurrency, an approach that attempts to address race conditions with most routine Python programs (Nagle, 2013). *Message passing* relies on a queue to break a job into smaller segments that are organized by simple objects. More complex jobs contain a poison pill that can halt the processes and avoid race conditions (Hellman, 2013). Python programmer John Nagle noted, "*Shared data* is permitted, using *AtomicObject* or *SynchronizedObject* objects. So, if there's a big data structure being updated by multiple threads, it can be shared between threads. Within an *AtomicObject* or *SynchronizedObject* object, race conditions are automatically prevented by the automatic locking" (Nagle, 2013). However, the National Vulnerabilty Database has found that Python 2.6 through 3.2 does create race conditions that allow users to obtain credentials after they access *~/.pypirc* under certain circumstances (DHS, 2013). This is also an access-control vulnerability. Yet, Python seems to have fewer vulnerabilities than some of the other languages.

Number Eight: Visual Basic

Programmer David Rutten claims that Visual Basic (VB) is "much friendlier than Python." He doesn't like Python because it is both case sensitive and indentation sensitive (Rutten, 2013). In February 2013, TIOBE ranked VB at number eight. Its usage grew by a meager 0.33 percent from the previous year, and 4.648 percent of new programs on Microsoft platforms have VB code (TIOBE Software, 2013). It is more than a language, but rather a system that creates new possibilities for developers who work with the Windows operating system. It enables developers to actually visualize the interface (Mabutt, 2013).

VB is not without buffer overflows. For example, Microsoft Animation ActiveX control in 6.0 allows remote hackers to run arbitrary code with an audiovisual interleave (AVI) file. This causes memory corruption and an 'allocation error.' (CVE Details, 2013). Enterprise Edition 6.0 is also vulnerable when a long *CommandName* line or *ConnectionName* line is abused in a Micrografx Designer Graphic (.dsr) file (CVE Details, 2013). There are more examples, but space is limited.

Race conditions are also a known vulnerability in VB. Microsoft recommends the locking of shared variables. If properly written, a sequence allows one thread to access to

the shared variable at a time (Microsoft Support, 2013). Access-control is another common problem that also plagues VB. Therefore, security policies and appropriate access permission levels are essential ways to control access and manage risk (Microsoft Support, 2013).

VB 11.0 was rolled out in 2012 on the .NET framework. Objects have continued to be a vital component of VB throughout its evolution, yet syntax has changed dramatically since the release of VB 6.0. It is no longer backward compatible. One of the biggest changes was a radical new *edit-and-continue* function at the source code level. As a result, security vulnerabilities are much more difficult to fix. For example, Microsoft acknowledges that "you must change the target platform and compile the application as a 32-bit application" if a coder is debugging a 64-bit application and wishes to use *Edit and Continue* (MSDN, 2013). Several efforts to improve security have been criticized by programmers because it creates much more work and slows processes, such as start-up time. Common Language Runtime virtual machine from Microsoft is designed to run secure managed code, while operating with unmanaged code. From a security standpoint, this is good because there are error-handling mechanisms (Keserovic et al., 2013). In the end, many of these processes introduced by Microsoft force the programmer to do the right thing, and that is time consuming. More programming languages and support communities could learn from Microsoft, at least in this regard.

Number Nine: Perl

VB only runs on Windows, while Perl runs on multiple operating systems. Both are object-oriented (Perlmonks, 2013). Usage of Perl declined slightly by -0.68 percent from the previous year; TIOBE found that 2.252 percent of programmers use Perl as of February 2013 (TIOBE Software, 2013). It has a reputation among some coders as archaic and arcane. Paul Venezia observed that, "Perl has been an instrumental part of the innovation and technological advancements of the last two decades, and it's served as a catalyst for a significant number of other languages that have contributed heavily to the programming world in general" (Slashdot, 2013). That being said, Perl is not exactly a dinosaur. Its code is the basis for web development applications such as Movable Type, Ticketmaster and LiveJournal. It is also the basis for web application development frameworks, including Catalyst and Titanium (Home of Szabgab, 2013).

It began as a UNIX scripting language, and Perl does draw more than inspiration from C. Shell scripting; data extraction utility AWK and stream editor (*sed*) parser are all borrowed directly from C. Parsing is a big part of what made Perl a big deal in the 1990s, and enabled common gateway interfaces (CGIs) to serve dynamic content as the dot.com boom was unfolding (W3C, 2013). The power of Perl's CGI capacity is also a security weakness. Clients can gain unauthorized access. As a result, there are many examples of havoc. The simplest way to limit the risk is to cut off the access of certain functions. For example, server-side includes should be turned-off. Access via UNIX – especially in the Bourne shell – should be restricted. In particular, special characters can be used by an attacker in Bourne and can confuse the script, thus gaining access without authorization (Apache Admin, 2013).

Perl has a number of features that are designed to prevent attacks. For example, it has a sophisticated mechanism to avert algorithmic complexity attacks. These attacks often lead

to denial of service (Crosby & Wallach, 2013). Developers of Perl have ingeniously created a hashing function to recalculate keys to change the order of the elements of the message. However, as clever as this fortification is, Yves Orton found that there is a vulnerability that is exploitable (Best Practical, 2013). To be successful, attackers induce an execution time that is "worst-case" (Crosby & Wallach, 2013). In normal usage, the worst case is nearly impossible, but the vulnerability is that it creates a condition in which hash tables experience collisions. This can happen when there are two identical values, either the 32-bit hash value or when the result of a division (modulus) operation becomes identical (Crosby & Wallach, 2013).

Perl was not developed as an object-oriented language. Many programmers wish to use objects because they allow programs to locate, access, modify and secure data. Based on this, semantics, syntax, special variables, modules and packages have been added by some to make Perl object-oriented (Conway, 1999). It can be argued that this added complexity additional security vulnerabilities.

Number Ten: Ruby

Perl and Ruby are diametrically opposed on many levels, with the exception of their syntaxes (C2, n.d.). Both borrow a great deal from C (Cunningham & Cunningham, 2013). Perl offers many ways to accomplish the same thing. Ruby was written as an object-oriented language from the beginning and is much more rigid. This rigidity could also be viewed as more consistent, simple, and even elegant, so that fewer things that can go wrong (Morin, 2013). TIOBE found that Ruby's usage grew by 0.19 percent from the previous year, and 1.75 percent of programmers use Ruby as of 2013. Ruby displaced JavaScript as the tenth most popular language (TIOBE Software, 2013).

Media and conference entrepreneur Tim O'Reilly is keen on the open source framework: "Ruby on Rails is a breakthrough in lowering the barriers of entry to programming. Powerful web applications that formerly might have taken weeks or months□to develop can be produced in a matter of days" (Ruby on Rails, 2013). Ruby may be number ten, but is popular enough to have been used to build web applications for Groupon, Lumosity, and the White Pages (Ruby on Rails, 2013).

Such sites use eXtensible Markup Language (XML). XML is a very popular format to share structured data over networks and the web. An XML parser allows the receiving computer to read the incoming structured data. Needless to say, having to disable the parser because of a security flaw would be an interruption to eCommerce. An access control flaw was first noted in passing by a developer in a web forum in 2007. Ramifications of the flaw in the XML parser were explained this year by Aaron Patterson, which finally prompted a patch by 37Signals, the firm behind Rails. Patterson explained that Ruby on Rails allows an attacker to "bypass authentication systems, inject arbitrary SQL, inject and execute arbitrary code, or perform a DoS attack on a Rails application." During the six years after the flaw was found, Ruby on Rails was vulnerable to significant mischief (Lee, 2013).

Ruby developers can manage security vulnerabilities, but their supervisors should support the additional time needed to test for insufficient transport layer protection, injection, cross-site scripting, broken authentication (and session management), insecure cryptographic storage, insecure direct object references, security misconfiguration, cross-site request forgery, failure to restrict URL access, and un-validated redirects and forwards.

Ruby was created with the intention of reducing security vulnerabilities, but its C-like syntax does not help that cause. Ruby has a filter that offers a workaround to prevent SQL injections. However, Ruby programmers need to be aware that they must use *Model.find(id)* or *Model.find_by_some thing(something)*. This avoids line breaks and written symbols (VeraCode, 2013).

There are security review platforms for many languages and applications, including Ruby. Veracode claims to provide governance, operating controls, eLearning, and application intelligence, in addition to its scanning capabilities (VeraCode, 2013). However, it is not enough to run tests after an application is "complete." It is essential for programmers to be aware of how to minimize vulnerabilities as they build software. They must also stay informed on a daily basis of flaws that become known. They must also download patches as soon as they become available and avoid posting vulnerabilities in online forums.

CONCLUSION

Deloitte conducted a survey and found that 87 percent of CIOs were most concerned about software development quality as the top threat in the pantheon of risks they face (Deloitte, 2007). One should not just point their finger at programmers. In many cases, flawed specifications and designs are the root cause. In other cases, there is more complexity than the budget or team can handle. Complexity is closely associated with insecurity because it creates more variables that can be exploited because of unintended or unexpected interactions. However, it is very common to see a lack of understanding in secure coding practices among programmers.

The Software Development Process Lifecycle requires the close supervision of a project manager who understands the critical nature of security. A CIO should also sign-off at each stage, starting with system conceptualization, system requirements of the system, analysis of benefits, and scope of project. The business and system analyses are critical and must include security parameters for programmers in software requirement specifications. System, architectural and detail design then provide additional opportunities for management to maintain quality controls for developers. Once the planning and design stages are approved, unit development, software and system integration, followed by testing and retesting, are critical. Clients are often most aware of the software during installation, site testing and acceptance, training/documentation, implementation and maintenance. However, that is too late for countermeasures for the vulnerabilities to be effective (Center for Technology in Government, University at Albany/SUNY, 1998).

REFERENCES

Allen, J. (2010). "Top 5 iPhone application development security issues." Intrepiduc Group Mobile Security. Retrieved from http://intrepidusgroup.com/insight/2010/05/top-5-iphone-application-development-security-issues/.

Apache Admin. (2013). "Writing secure CGI scripts." Retrieved from http://www.apacheadmin.com/CGI/security.html.

Apple Developer. (2012). "OS X: Cocoa." Retrieved from https://developer.apple.com/technologies/mac/cocoa.html.

Best Practical. (2013). "Security vulnerability in Perl." Retrieved from http://blog.bestpractical.com/2013/03/security-vulnerability-in-perl.html.

Blaikie, D. (2012). "Discussion: Should we enforce access control in C++ attributes?" University of Illinois at Urbana-Champaign. Retrieved from http://lists.cs.uiuc.edu/pipermail/cfe-dev/2012-October/025474.html.

Bradley, T. (2013). "Hackers use patches To develop exploits." About.com Internet/Network Security. Retrieved from http://netsecurity.about.com/cs/generalsecurity/a/aa022904.htm.

C2. (n.d.). "Ruby vs. Perl." Retrieved from http://c2.com/cgi/wiki?RubyVsPerl.

Canonical. (2013). "Resume, Kragen Sitaker." Retrieved from http://canonical.org/~kragen/resume.html.

Center for Technology in Government, University at Albany/SUNY. (1998). "A survey of system development process models." Retrieved from http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf.

CERT/Software Engineering Institute/Carnegie Mellon. (2013). "Vulnerability note VU#625617." Retrieved from http://www.kb.cert.org/vuls/id/625617.

Conway, D. (1999). "What is object-oriented Perl?" Monash University. Retrieved from http://www.csse.monash.edu.au/~damian/papers/PDF/cyberdigest.pdf.

C Programming. (2013). "Character input validation." Retrieved from http://cboard.cprogramming.com/c-programming/139475-character-input-validation.html.

Crosby, S. & Wallach, D. (2013). "Denial of service via algorithmic complexity attacks." Rice University Computer Science Department. Retrieved fromhttp://www.cs.rice.edu/~scrosby/hash/CrosbyWallach_UsenixSec2003/.

Cunningham & Cunningham. (2013). "Cee language." Retrieved from http://c2.com/cgi/wiki?CeeLanguage.

CVE Details. (2013). "Visual Basic: Security vulnerabilities." Retrieved from http://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-322/Microsoft-Visual-Basic.html.

Dallaway, R. (2013). "Java web start and code signing." Retrieved from http://www.dallaway.com/acad/webstart/.

Deloitte. (2007). "Deloitte 2007 global security survey: The shifting security paradigm." Retrieved from http://www.deloitte.com/dtt/cda/doc/content/dtt_gfsi_GlobalSecuritySurvey_20070901(1).pdf.

DHS. (2013). "National cyber awareness system." Retrieved from

http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-4944.
DHS National Cyber Security Division/US-CERT. (2013). "National vulnerability database." Retrieved from http://nvd.nist.gov/.
DHS National Cyber Security Division/US-CERT. (2013). "National vulnerability database version 2.2." Retrieved from http://nvd.nist.gov/
Esser, S. (2013). "Zend_Hash_Del_Key_Or_Index Vulnerability." Retrieved from http://www.hardened-php.net/hphp/zend_hash_del_key_or_index_vulnerability.html.
eTutorials. (2013). "C/C++ secure programming." Retrieved from http://etutorials.org/Programming/secure+programming/Chapter+3.+Input+Validation/3.3+Preventing+Buffer+Overflows/.
Fuecks, H. (2013). "The real difference between PHP and Python." Retrieved from http://www.sitepoint.com/the-real-difference-between-php-and-python/.
Godel, J. (2001). "A comparison of C/C++ and C#." Developer Fusion. Retrieved from http://www.developerfusion.com/article/1743/a-comparison-of-cc-and-c/2/.
Graham, P. (2004). "The Python paradox." Retrieved from http://www.paulgraham.com/pypar.html.
Green SQL. (2013). "MySQL security best practices (hardening MySQL tips)." Retrieved from http://www.greensql.com/articles/mysql-security-best-practices.
Grubb, K. (2012). "What does PHP stand for?" Bright Hub. Retrieved from http://www.brighthub.com/internet/web-development/articles/62713.aspx.
Gustafson, P. (2013). "Detecting and avoiding OpenMP race conditions in C++." Sun Developer Network. Retrieved from http://dsc.sun.com/solaris/articles/cpp_race.html.
Harmful Stuff. (2007). "Linus torvalds on C++." Retrieved from http://harmful.cat-v.org/software/c++/linus.
Hellman, D. (2013). "Communications between processes." Retrieved from http://pymotw.com/2/multiprocessing/communication.html.
Hoglund, G. & McGraw, G. (2004). *Exploiting software: How to break code.* Addison Wesley, Boston, 2004.
Home of Szabgab. (2013). "What is Perl used for?" Retrieved from http://szabgab.com/what-is-perl-used-for.html.
IBM Developer Works. (2004). "Secure programmer: Prevent race conditions." Retrieved from http://www.ibm.com/developerworks/library/l-sprace/index.html.
Ideal Programmer. (2009). "ASP.NET web site security vulnerability cheat sheet." Retrieved from http://idealprogrammer.com/net-languages/asp/aspnet-web-site-security-vulnerability-cheat-sheet/.
iOS Developer Library. (2010). "Introduction: Object-oriented programming with objective-C." Retrieved from http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/OOP_ObjC/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005149-CH1-SW2.
iOS Developer Library. (2012). "Programming with Objective-C: Encapsulating Data." Retrieved from http://developer.apple.com/library/ios/#documentation/cocoa/conceptual/ProgrammingWithObjectiveC/EncapsulatingData/EncapsulatingData.html.
Java Revisited. (2012). "What is race condition in multithreading – 2 examples in Java."

Retrieved from http://javarevisited.blogspot.com/2012/02/what-is-race-condition-in.html.
Java Tutorials. (2013). "Deadlock." Retrieved from http://docs.oracle.com/javase/tutorial/essential/concurrency/deadlock.html.
Joyce, J. (2012). "Why C is not my favorite programming language." Retrieved from http://www.kuro5hin.org/story/2004/2/7/144019/8872.
Keserovic, S., Mortenson, D., & Nathan, A. (2013). "An overview of managed/unmanaged code interoperability." Retrieved from http://msdn.microsoft.com/en-us/library/ms973872.aspx.
Lee, M. (2013). "Ruby on Rails vulnerable to six year old flaw." Retrieved from http://www.zdnet.com/ruby-on-rails-vulnerable-to-six-year-old-flaw-7000009559/.
Livshits, V.B. & Lam, M.S. (2013). "Finding security vulnerabilities in Java applications with static analysis." Computer Science Department, Stanford University. Retrieved from http://suif.stanford.edu/papers/usenixsec05.pdf.
Long, F. (2005). "Software vulnerabilities in Java." Software Engineering Institute, Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/library/abstracts/reports/05tn044.cfm.
Mabbutt, D. (2013). "What is Visual Basic?" Retrieved from http://visualbasic.about.com/od/applications/a/whatisvb.htm.
Mac Developer Library. (2012). "Race conditions and secure file operations." Retrieved from https://developer.apple.com/library/mac/#documentation/security/conceptual/SecureCodingGuide/Articles/RaceConditions.html.
Mac Developer Library. (2013). "Types of security vulnerabilities." Retrieved from https://developer.apple.com/library/mac/#documentation/security/Conceptual/SecureCodingGuide/Articles/TypesSecVuln.html.
Mehra, P. (2009). "Garbage collection in C#." Retrieved from http://www.c-sharpcorner.com/uploadfile/puranindia/garbage-collection-in-C-Sharp/.
Microsoft Support. (2013). "Description of race conditions and deadlock." Retrieved from http://support.microsoft.com/kb/317723.
Microsoft Support. (2013). "Security policy." Retrieved from http://msdn.microsoft.com/en-us/library/tha13y5z(v=vs.80).aspx.
Mohammed, H. (2013). "C#.NET getting started." Information & Technology Management. Retrieved from http://www.itmusa.net/netcsharpdayone.asp.
Mohrindra, D. & Jumde, P. (2012). "IDS03-J. Do not log unsanitized user input." CERT / Software Engineering Institute/Carnegie Mellon. Retrieved from https://www.securecoding.cert.org/confluence/display/java/IDS03-J.+Do+not+log+unsanitized+user+input.
Morin, M. (2013). "How does Ruby compare to Perl?" About.com. Retrieved from http://ruby.about.com/od/beginningruby/a/vsperl.htm.
MSDN. (2013). "Edit and continue (visual basic)." Retrieved from http://msdn.microsoft.com/en-us/library/ba77s56w.aspx.
MySQL. (2013). "6.1.4. Security-related mysqld options and variables." Retrieved from http://dev.mysql.com/doc/refman/5.0/en/security-options.html.
Nagle, J. (2013). "Newthreading – Safer concurrency for Python." Retrieved from http://www.animats.com/papers/languages/newthreadingintro.html.

OWASP. (2009). “Race conditions.” Retrieved from https://www.owasp.org/index.php/Race_Conditions.
Perlmonks. (2013). “Perl vs. VB.” Retrieved from http://www.perlmonks.org/?node_id=227771.
Peterson, D. (2009). “Patch Tuesday leads to exploit Wednesday.” Digital Bond. Retrieved from http://www.digitalbond.com/blog/2009/10/15/patch-tuesday-leads-to-exploit-friday/.
PHP Group. (2013). “Bugs.” Retrieved from https://bugs.php.net/stats.php.
PHP Group. (2013). “Bug #23791 Mod doesn’t return correct values.” Retrieved from https://bugs.php.net/bug.php?id=23791.
PHP Group. (2013). “Introduction to PDO.” Retrieved from http://www.php.net/manual/en/intro.pdo.php.
PHP Group. (2013). “MySQL functions (PDO_MYSQL).” Retrieved from http://php.net/manual/en/ref.pdo-mysql.php
PHP Group. (2013). “PHP5 changelog.” Retrieved from http://www.php.net/ChangeLog-5.php#5.4.12.
PHP Group. (2013). “Unsupported historical releases.” Retrieved from http://php.net/releases/index.php.
Princeton University. (2013). “C programming vs. Java programming.” Retrieved from http://introcs.cs.princeton.edu/java/faq/c2java.html.
Programming4Us. (2011). “Mobile application security testing.” Retrieved from shttp://programming4.us/mobile/1676.aspx.
Python. (2013). “Python programming language – Official website.” Retrieved from http://www.python.org/.
Python. (2013). “Security advisories.” Retrieved from http://www.python.org/news/security/.
Ruby on Rails. (2013). “Ruby on Rails.” Retrieved from http://rubyonrails.org/.
Ruby on Rails. (2013). “Ruby on Rails applications.” Retrieved from http://rubyonrails.org/applications.
Rutman, M. (2013). “C++ versus objective-C.” MacTech. Retrieved from http://www.mactech.com/articles/mactech/Vol.13/13.03/CandObjectiveCCompared/index.html.
Rutten, D. (2013). “Visual Basic vs. Python.” Retrieved from http://www.grasshopper3d.com/forum/topics/visual-basic-vs-python?xg_source=activity.
Seacord, R. (2009). CERT. “CS 15392 secure programming.”
Seacord, R. (2005). “Secure coding in C and C++: C-style strings.” Retrieved from http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature120061.cfm.
Secure Coding Guide. (2011). “Validating input and interprocess communication.” Retrieved fromhttp://developer.apple.com/library/ios/#documentation/Security/Conceptual/SecureCodingGuide/Articles/ValidatingInput.html#//apple_ref/doc/uid/TP40007246.
Slashdot. (2013). “Perl’s glory days are behind it, but it isn’t going anywhere.” Retrieved from http://developers.slashdot.org/story/13/01/29/0235220/perls-glory-days-are-behind-it-but-it-isnt-going-anywhere.
Smith. (2013). “Oracle releases emergency Java patch; Experts warn flaws may take 2 years

to fix." Network World. Retrieved from http://www.networkworld.com/community/blog/oracle-releases-emergency-java-patch-experts-warn-flaws-may-take-2-years-fix.

Swanson, E. (2013). "How PHP's foreach works." Hacker News. Retrieved fromhttps://news.ycombinator.com/item?id=5295034.

Taft, D. (2012). "Java drops from top programming language spot, C rules." eWeek. Retrieved from http://www.eweek.com/c/a/Application-Development/Java-Drops-From-Top-Programming-Language-Spot-C-Rules-626622/.

Tomar, G. (2008). "Difference between C++ and C#," C# corner." Retrieved from http://www.c-sharpcorner.com/uploadfile/gtomar/difference-between-cpp-and-C-Sharp/.

TIOBE Software. (2013). "TIOBE programming community index for February 2013." Retrieved from http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

University of Michigan. (2013). "The C programming language." Retrieved from http://groups.engin.umd.umich.edu/CIS/course.des/cis400/c/c.html.

Using C# Net. "Check .NET code for security vulnerabilities." Retrieved from http://www.usingcsharp.net/2010/05/check-net-code-for-security-vulnerabilities/.

Velocity Reviews. (2013). "Re: Is python buffer overflow proof?" Retrieved from http://www.velocityreviews.com/forums/t693508-re-is-python-buffer-overflow-proof.html.

VeraCode. (2013). "Ruby on Rails secure development guidelines." Retrieved from http://www.veracode.com/security/ruby-security

W3C. (2013). "CGI: Common Gateway Interface." Retrieved from http://www.w3.org/CGI/.

Web Builders. (2013). "Code sign errors." Retrieved from http://webbuilders.wordpress.com/2009/12/25/code-sign-errors-profile-doesnt-match-any-valid-certificateprivate-key-pair-in-the-default-keychain/.

Wikibooks. (2013). "C++ programming: C# comparison with C++." Retrieved from http://en.wikibooks.org/wiki/C%2B%2B_Programming/Programming_Languages/Comparisons/C_Sharp.

Yahoo Answers. (2012). "What is the relation between MySQL, PHP and Wordpress?," Retrieved from http://answers.yahoo.com/question/index?qid=20100219142110AAytgyT.

Y Combinator. (2013). "Hacker news." Retrieved from https://news.ycombinator.com/item?id=3449388.

Younan, Y. (2013). "C and C++: Vulnerabilities, exploits and countermeasures." Security Research Group. Retrieved from http://secappdev.org/handouts/2012/Yves%20Younan/C%20and%20C++%20vulnerabilities.pdf.