On the use of C# Unsafe Code Context: An Empirical Study of Stack Overflow

Ehsan Firouzi*
Shiraz University, Shiraz, Iran
e.firouzi@shirazu.ac.ir

Ashkan Sami*
Shiraz University, Shiraz, Iran
sami@shirazu.ac.ir

Foutse Khomh
Polytechnique Montréal, Montréal,Canada
foutse.khomh@polymtl.ca

Gias Uddin
University of Calgary, Calgary, Canada
gias.uddin@ucalgary.ca

ABSTRACT

Background. C# maintains type safety and security by not allowing direct dangerous pointer arithmetic. To improve performance for special cases, pointer arithmetic is provided via an unsafe context. Programmers can use the C# `unsafe` keyword to encapsulate a code block, which can use pointer arithmetic. In the Common Language Runtime (CLR), unsafe code is referred to as unverifiable code. It then becomes the responsibility of the programmer to ensure the encapsulated code snippet is not dangerous. Naturally, this raises concern on whether such trust is misused by programmers when they promote the use of C# unsafe context. **Aim.** We aim to analyze the prevalence and vulnerabilities of share code examples using C# unsafe keyword in Stack Overflow (SO) code sharing platform. **Method.** By using some regular expressions and manual checks, we extracted C# unsafe code relevant posts from SO and categorized them into some software development scenarios. **Results.** In the entire SO data dump of September 2018, we find 2,283 C# snippets with the `unsafe` keyword. Among those posts, 27% of posts are about Image processing, where unsafe codes are mainly used for performance reasons. The second most popular category by 21% of the codes in the posts is used for 'Interoperability' reasons. That is `'unsafe'` is used to enable 'Interoperability' between C# managed codes and unmanaged codes. The `'stackalloc'` operator is the third category with 9% of unsafe code posts. The `stackalloc` operator allocates a block of memory on the stack. Since C# 7.2, Microsoft recommends against using `'stackalloc'` in unsafe context whenever possible. Manual inspection shows 67 code snippets with dangerous functions that can introduce vulnerability if not used with caution (e.g., buffer overflow). Finally, 35% of 'Interoperability' posts have 'P/Invoke' tag were used outside NativeMethods class, which is in contrast to Microsoft design suggestion. **Conclusion.** Our study leads to 7 main findings, and these findings show the importance of cautiously using this feature.

*Corresponding authors

ESEM '20, October 8–9, 2020, Bari, Italy

ACM ISBN 978-1-4503-7580-1/20/10…$15.00
https://doi.org/10.1145/3382494.3422165

CCS CONCEPTS

• **Software and its engineering → Language features**.

KEYWORDS

Unsafe code, Software Security, Unmanaged Code, C#, Crowd Sourced, Stack Overflow, Data Analysis

1 INTRODUCTION

C# is one of the most popular languages among developers. It is the fourth ranked in PYPL PopularitY index [9] and fifth ranked in TIOBE index[6]. C# is designed as a managed language. The Common Language Runtime (CLR) and the .NET architecture in C# provide many useful classes and services that let developers easily write secure codes. Even system administrators can customize the permissions in code so that access to protected resources is secure. Moreover, the runtime and the .NET facilitate the use of cryptography and role-based security by a set of special classes and services, which provides secure memory management [16].

In some cases, developers need to access the raw memory directly, such as to directly manipulate a memory buffer for increased runtime performance. In other languages, such as C/C++, this is normally achieved via the memory pointers. Given C# does not allow direct memory access by pointer, `unsafe` keyword is introduced to provide this capability instead. Unsafe code is required when one calls native functions that require pointers. In some cases, unsafe code may increase performance of the application by removing array bounds checking, but developers should be aware that using unsafe code introduces security and stability risks [17].

By allowing developers the freedom to access memory through the unsafe context, C# puts trust on the developer for the safe usage of the language. Naturally, this raises the question of how developers are actually using the unsafe context and in what context, and whether such usage can pose any security threat. One way to learn about this is to analyze how software developers are sharing C# code snippets with `unsafe` keyword in online developer forums, such as Stack Overflow. The unique popularity of Stack Overflow in the software development community makes it an ideal candidate for evaluating and analyzing security issues and the overall trends of unsafe code usage. Stack Overflow posts contain many tags and 'C#' tag is the third most popular tag in Stack Overflow.

We used the Stack Overflow data dump of September 2018. The data-set contains several tables such as Post table which contains more than 40 million posts, each has a text (e.g., body which contains two parts: the text block and the code block), Users which contains

about 9 million users' information, Comments, etc [25]. We analyze all the C# code snippets in the dump and answer three research questions:

RQ1: How prevalent are the C# unsafe contexts in the Stack Overflow shared code examples? Around 0.2% of all shared C# code snippets contain the `unsafe` keyword. Only 73.8% of the Stack Overflow question threads where we find the code snippets with `unsafe` keyword, are tagged as 'unsafe'.

RQ2: What types of software development scenarios are addressed in the shared C# code examples that use unsafe context? Most (27%) of the code snippets containing the `unsafe` keywords are labeled with tags related to 'image processing'. The tags related to 'interoperability' were found second most (21%).

RQ3. How vulnerable are the shared code examples using unsafe C# context? We frequently observed dangerous pointer operators inside the `unsafe` keyword, which is not surprising. However, we also observed `'stackalloc'` operator, which is not recommended by C# since version 7.2. Despite warned by Microsoft, P/Invokes are still used out of NativeMethods classes context. Therefore, unless properly guarded, the reuse of such unsafe C# code snippets can make a software application buggy with critical memory and security issues.

To the best of our knowledge, this is the first study that has focused on C# security issues based on Stack Overflow posts.

2 RELATED WORK

Vast number of researches focused on security assessment in different programming languages [22], [2], [31], [21], [28]. Security assessment in languages like Java, C and C++ have established guidelines, standardizations and even recommendations [12], [26], [23]. In contrast, C# does not have an established guideline verified by CERT. The only official documentation regarding software security issues was 'Security code scan – static code analyser for .NET', which is a Visual Studio extension covering some security vulnerability patterns and does not assess 'unsafe' code blocks [1]. Several books on different aspects and capabilities of C# security mechanisms exists like [8], [10], [19].

Although studies on C# are not vast, they are mainly focused on other aspects than security evaluations. For instance, [7] focused on error proneness of Resharper. The closest work we found was by Sharma et. Al [24] that only focused on code smells and no security assessment was mentioned. Based on best of our knowledge, this is the first study on security of C# codes. several studies investigated security related issues in languages like Java and Python [22], [11], [5]. However, as mentioned no study has focused on security of C# codes and thus no study on C# code snippets in community question and answer websites exist. C++ code snippets which are used in C# unsafe codes be prone to misuses (e.g., memory corruption bugs) that can easily lead to vulnerable code and exploitable applications [29], [18] .

3 DATA COLLECTION AND PREPROCESSING

To conduct a comprehensive empirical study, we use a whole Stack Overflow dataset which is publicly available on Stack Exchange Data Dump released 2018-09-05 [25]. Our dataset contains 41,782,536 posts and 9,321,924 users. Fig. 1 depicts the data preprocessing steps.

First, all the posts with `'unsafe'` tag were extracted and only 599 cases were detected, although Stack Overflow asks users to attach one or more tags with their question. We found out in majority of cases unsafe code was not labeled `'unsafe'` tag properly. After reviewing the tags and titles of this posts we extract some related keywords : `'Pointer'`, `'Image'`, `'Performance'`, `'P/invoke'`, `'Manage'`, `'Unmanage'`, `'Marshal'`, `'Memory'`, `'GDI'`, `'Optimize'`, `'DLL'`, `'Stack'`, `'Low-level'`, `'Pixel'`, `'COM'`,`'Jpg'`, `'Intptr'`, `'Bitmap'`, `'Stackalloc'`, `'Interop'`, `'Struct'` and `'Fixed'`. In the second step, we used three heuristics:

(1) Searching tags, for related tags to unsafe code (tags which contains extracted related keywords)
(2) Searching post titles for extracted related keywords. We just did not suffice for searching `'unsafe'` keyword alone since the code or answer may be unsafe but the part of the code that was posted did not contain an `'unsafe'` keyword explicitly.
(3) We used a regular expression for `'unsafe'` and '{' in the code snippets. We searched `'unsafe'` and having '{' within at maximum 200 characters after `'unsafe'`. Again the reason that we may have unsafe codes without `'unsafe'` keyword is that the answer may be unsafe by nature but `'unsafe'` keyword may not be present in the code snippets due to space limitations or as a part of a short answer. We performed a manual check of the extracted code snippets to make sure the extracted codes are indeed unsafe. After these three heuristics 1684 unsafe code snippets that were not labeled by `'unsafe'` tag were detected.

4 STUDY RESULTS

In this section, we present our results and discuss the main finding regarding our stated research questions.

4.1 Prevalence of C# Unsafe Contexts (RQ1)

4.1.1 Motivation. Given that unsafe codes allow programmers to write low-level functions and manipulate pointers directly, it would be interesting to know how frequently this feature is used by C# developers in Stack Overflow.

4.1.2 Approach. In order to collect posts related to unsafe codes, as we described in the preprocessing and is shown in the Fig. 1, first we used regular expressions for the `'unsafe'` tag. Then we used three heuristics. After that, we calculated their distribution over the years and also compared them with C# codes distribution.

4.1.3 Results. Our data set contains 41,782,536 posts, of which we find 2,283 C# unsafe code snippets. Calculating the percentage of unsafe codes leads to 73.76% of unsafe code not having an 'unsafe' label. Simply, nearly three-quarters of 'unsafe' posts are not properly tagged. Out of all the posts in Stack Overflow, 1,260,054 posts have C# tag. Around 30% of unsafe codes do not have a 'C#' tag.

In Fig. 2 the percentage of C# and the unsafe code posts used per year is illustrated. The percentages are calculated by summing all the relative posts together and calculating the percentage by dividing the number of posts of each year to the total. As can be seen in Fig. 2 unsafe posts follow the same distribution as C# posts. That is, the fraction of questions related to unsafe codes has not

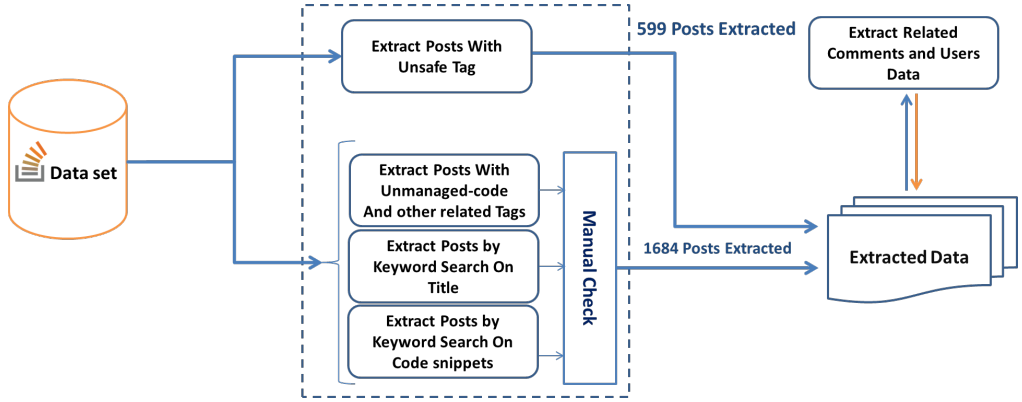

Figure 1: Workflow of data collection

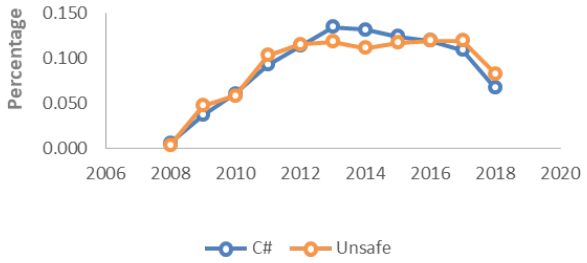

Figure 2: Distribution of C# and unsafe posts per year over the period of 2008 to 2018

Table 1: The number of posts per year and the fraction of unsafe to C# posts each year

year	unsafe	C#	Unsafe/C#	All
2008	9	7,547	0.0012	274,536
2009	108	46,507	0.0023	1,288,974
2010	134	77,027	0.0017	2,165,685
2011	236	116,932	0.0020	3,497,052
2012	264	143,789	0.0018	4,502,404
2013	271	170,323	0.0016	5,413,518
2014	255	166,391	0.0015	5,394,744
2015	268	157,178	0.0017	5,393,778
2016	274	151,228	0.0018	5,330,165
2017	274	137,934	0.0020	5,184,550
2018	190	85,198	0.0022	3,337,130
All	2283	1,260,054	0.0018	41,782,536

changed much through the past 10 years. The actual number of posts in each year and the fraction of unsafe to C# is shown in Table 1.

> Finding 1: Almost three-quarters of unsafe posts do not have `unsafe` tags. To be exact, 73.76% of unsafe codes do not have an `'unsafe'` tag, also 30% of unsafe codes do not have 'C#' tag.

4.2 Types of software development scenarios in SO that use Unsafe Context (RQ2)

4.2.1 Motivation. Our findings from RQ1 show that there is a considerable number of C# code snippets shared in SO with the `unsafe` keyword. It is, therefore, useful to learn what specific use case scenarios are supported by those code snippets.

4.2.2 Approach. First, we collect all the question threads in SO, where the C# code snippets with `unsafe` keyword are shared. Second, for each question, we collect all the tags. Third, we manually determine the underlying use case scenario of each code snippet by manually assessing their posts, their first used tag, and the corresponding title of the post. To categorize the 2,283 founded unsafe codes into these scenarios we employed the following three heuristics:

(1) Categorizing code snippets based on tags. For instance, tags such as ⟨image⟩, ⟨GDI⟩, ⟨BMP⟩ led us to categorize the post into the 'Image Processing" category. Since each post usually has more than one tag so the small number of overlap occurred. For instance, we had posts that categorized as 'Image Processing' and 'Performance'. However, most of the posts were only categorize into one category .
(2) Some posts do not have proper tags that help us categorize the post, for instance, post-Id 212155 has only 'C#' tag that cannot help us categorize it into one of these categories. Thus, we used keywords search to categorize the post into one of these categories. The keywords by themselves are excellent representatives of the type of post. For instance, the post was categorized into the Image-Processing category if keywords like `'image'`, `'IMG'`,`'GDI'`, `'picture'`, or similar keywords were found. If `'DLL'`, `'COM'`, `'Interoperability'`, `'marshal'`, or `'marshaling'` were among keywords, the post was categorized into 'Interoperability' category. Sometimes the regular expression was not sufficient by itself. For instance, having 'C++' keyword and the post that contains unsafe code that shares a form of data between C++ and C# would be a case of 'Interoperability' category.
(3) If the post could not be categorized into one of the categories or the post could be categorized into several categories, we manually read and checked the post, so based on manual inspection the category was defined.

4.2.3 Results. We find three types of use case scenarios where the `unsafe` keyword is used in the C# code snippets:

(1) Image processing
(2) Interoperability
(3) Memory Access, which is divided into following sub-categories:
  (a) Stack allocation,
  (b) Performance improvements,
  (c) Pointers, and
  (d) Memory Management.

We describe the scenarios with examples below.

Scenario 1. Image-processing. Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it [27]. Direct memory access to manipulate image data is provided through unsafe code to increases efficiency, which may introduce security vulnerabilities into code.

Scenario 2. Interoperability. Interoperability enables to preserve and take advantage of existing investments in unmanaged code. COM, COM+, C++ components, ActiveX components, and Microsoft Windows API are examples of unmanaged code[15]. The .NET framework provides different ways of achieving interoperability as described below.

- *Platform Invocation (P/Invoke)* allows for managed code to call native unmanaged functions implemented as *DLLs*. This method is ideal for when we have API-like functions written in C or C++ that need to be accessed from within a C# program. *Improper use of the Platform Invocation Services can render managed applications vulnerable to security flaws in other languages.*
  Microsoft suggests that Move 'P/Invokes' to NativeMethods classes (NativeMethods class,SafeNativeMethods class and UnsafeNativeMethods class)[14] :
  UnsafeNativeMethods class suppresses stack walks for unmanaged code permission. This class is for methods that are potentially dangerous. *Any caller of these methods must perform a full security review to make sure that the usage is secure because no stack walk will be performed.*
- *C++ Interop* is also known as implicit PInvoke . This mechanism consists of wrapping a native C++ class so that it can be consumed by C# code.
- *COM Interop* is a mechanism specifically for exposing COM components to a .NET language. In other words, the unmanaged code must be encapsulated as a COM object for this mechanism to be applicable [3].

Scenario 3. Memory Access. Unsafe code allows the following types of memory access[20]:

- *Stack allocation.* In C#, the `stackalloc` keyword is used in an unsafe code context to allocate a block of memory on the stack. This C# operator is used as a way of manually allocated memory buffers that can be used without type safety checks. There are several changes around `stackalloc` that were done for C# 7.2. Before C# 7.2 you could only use `stackalloc` as an initializer of local pointer variable and only within unsafe context. Since C# 7.2, `stackalloc` is an expression and is valid in conditional expressions and assignment expressions outside of unsafe contexts [32] .

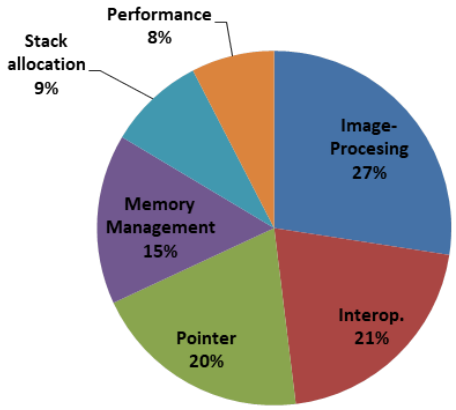

Figure 3: The distribution of 2,283 unsafe codes per scenario.

- *Performance.* In some cases, unsafe code may increase an application's performance by removing array bounds checks [13].
- *Pointers.* In C# pointers can point to only unmanaged types which include all basic data types, enum types, other pointer types, and structs which contain only unmanaged types and unlike reference types, pointer types are not tracked by the default garbage collection mechanism [30].
- *Memory management* was found in the following types:
  – Random access to arrays without bounds checks
  – Read from / write to "random" memory locations
  – Compare references (instead of the values they refer to)
  – Reinterpret (cast) a reference as a reference to a different type
  – Cast without dynamic type checks.

The distribution of unsafe codes based on their categories is shown in Fig. 3.

> Finding 2: unsafe categories we obtained were:
> (1) Image-processing
> (2) Interoperability
> (3) Memory Access (Stack allocation, Performance improvements, Pointers, Memory-Management (others))

As Fig. 3 depicts 'Image Processing' scenario which includes 27% of the unsafe codes (626 posts out of 2283 posts) is the most prevalent scenario in Stack Overflow among unsafe Codes. Moreover, 'Interoperability' is 20% of all unsafe code posts (471 out of 2283 posts) and is placed second. Also, using a pointer is placed on the third. However, each post may have more than one scenario and in our study, we have investigated the main scenario, which was more relevant to the content of the post. As we mentioned before, 626 posts out of 2,283 unsafe codes are related to 'Image processing'.These posts have focused more on the speed of the algorithm, the difference between unsafe and safe code in this scenario is the trade-off between speed and security (and stability).although, by using unsafe context even if the programmer assumes his/her

codes do not have any bugs or vulnerabilities, Based on the official Microsoft documentation the occurrence of unexpected behavior still exists [13].

> Finding 3: the Image-processing scenario by 626 posts includes 27% of the unsafe codes and is the most popular 'unsafe code' scenario in Stack Overflow.

Listing 1: using stakalloc in C# unsafe code- post id 24200904 in SO

```
public static unsafe void TestMethod1()
{
  float* samples = stackalloc float[12500000];
   for (var ii = 0; ii < 12500000; ii++) {
        samples[ii] = 32768; }
}
```

'stackalloc' category contains 202 of the total unsafe code snippets of our study. The stackalloc operator allocates one block of memory on the stack. A sample stackalloc usage is presented in Listing 1. This operator allocates memory in an unsafe context (so it should be used with caution) [26]. A stackalloc allocated memory block is not subject to garbage collection.

> Finding 4: we observed stackalloc (unsafe) operator 202 times. Since C# version7.2, stackalloc can be used outside of unsafe contexts, and Microsoft recommends using a safe version instead of unsafe wherever possible.

Listing 2: Generate random value with rand() - Answer id 9340784 in SO

```
namespace ConsoleApplication1 {
unsafe class Program : ILNumerics.ILMath {
public static List<Item> makeData(int n) {
    List<Item> ret = new List<Item>(n);
    using (ILScope.Enter()) {
     ILArray<double> A = rand(1,n);
     double[] values = A.GetArrayForRead();
      for (int i = 0; i < n; i++) {
            ret.Add(new Item() {
            Value = values[i] });}
    } return ret;
```

Listing 3: using strcpy in C# unsafe code- post id 789977 in SO

```
//C++ DLL
long func(struct name * myname)
{
     strcpy(myname–&gt;firstname,"rakesh");
     strcpy(myname–&gt;lastname,"agarwal");
     return S_OK;
}
//C# Part
[DllImport("C++Dll.dll")]
public unsafe static extern long func(name[]–
myname);
name[] myname = new name[1];
func(myname);
```

4.3 Unsafe C# Context Shared Code Examples Vulnerabilities(RQ3)

4.3.1 Motivation. Unsafe codes can create issues with stability and security, due to their inherent complex syntax and potential for memory-related errors, such as stack overflow, accessing, and overwriting system memory. Extra developer care is paramount for averting potential errors or security risks. Since the memory management of the unsafe codes is not checked by CLR, we investigated whether the programmers were obliged to use it or there are other ways, without using this dangerous feature to perform the tasks. In addition, we studied whether dangerous functions used in unsafe cods[4].

Listing 4: using memcpy in C# unsafe code- post id 14432361 in SO

```
 public unsafe void Set(Vector4 value)
{
    com.Set((int)&amp;value, sizeof(Vector4));
}
Then in C++:
void ShaderVariableCom::Set(__int32 data,int size)
{
  void* ptr = (void*)data;
  if (vertexOffset != -1)
    memcpy(vertexBytes+vertexOffset,ptr,size);
  if (pixelOffset != -1)
    memcpy(pixelBytes+pixelOffset,ptr,size);
}
```

4.3.2 Approach. Regular expressions have been used to find dangerous functions. Furthermore, the possibility of substitutions of unsafe codes with safe ones have been considered.

4.3.3 Results. Since the use of 'unsafe' makes the code unmanaged, using obsolete and dangerous functions may introduce a potentially severe vulnerability. For instance in Listing 2, 'unsafe' is used and has used 'rand()' function which is an obsolete function and causes the vulnerability. As in Listing 3 that strcpy() is used and in Listing 4 that has memcpy() in a code that is declared 'unsafe' may create serious vulnerabilities. The use of strcpy and memcpy when the code is run in CLR does not create a serious vulnerability since CLR provides the protection mechanisms. However, the use of 'unsafe' disables those mechanisms and may cause buffer overrun. atoi was the other dangerous function that we observed in the Stack Overflow 'unsafe' code snippets. We observed that if speed and performance are not the main concern of the developer, managed code can be used instead of unsafe codes for the Image-Processing, and also while the stackalloc function is used by some slight changes, we can change it to the safe version.

> Finding 5: In unsafe code snippets, 67 dangerous functions were observed:
>
> - 29 times Strcpy()
> - 24 times memcpy()
> - 12 times rand()
> - 2 times atoi()

Improper use of the Platform Invocation Services can render managed applications vulnerable to security flaws in other languages. Microsoft suggests that Move 'P/Invokes' to NativeMethods class [14]. We found 471 'Interoperability' scenario in Stack Overflow posts among them 165 are tagged as 'P/Invokes'. It is recommended to move 'P/Invokes' to NativeMethods class. NativeMethods are not safe either and use of these methods must follow a thorough security review so no vulnerability may be introduced.

Finding 6: Most of the Image-Processing snippet codes have used unsafe codes in order to access to the pixels, if speed and performance are not the developer main concerns using .Net libraries can decrease security risks and improve stability.

Finding 7: Despite of Microsoft design warning, 165 'P/Invokes' still are used out of NativeMethods class context.

5 THREATS TO VALIDITY

There are several threats that may potentially affect the validity of our study. Threats to internal validity relate to errors in our analyses. We have double-checked our results. The extraction phases may miss several unsafe code related post due to our extraction approach. Threats to external validity relate to the generalizability of our results. We have conducted the empirical study on C# questions on Stack Overflow. In the future, we plan to reduce the threats further by investigating more question and answer websites and ensure the generalizability of our conclusions.

6 CONCLUSION

C# maintains type safety and security by not directly supporting dangerous pointer arithmetic.However, programmers can use the C# `unsafe` keyword to encapsulate a code block, which can use pointer arithmetic. In the Common Language Runtime (CLR), unsafe code is referred to as unverifiable code. Naturally, this raises concern on whether such trust is misused by programmers when they promote the use of C# unsafe context. One way to lean about this is to analyze the C# unsafe code contexts shared in Stack Overflow. Stack Overflow (SO) is the most popular online Q&A site for developers to share their expertise in solving programming issues. In the entire Stack Overflow data dump of September 2018, we find 2,283 C# snippets with the `unsafe` keyword. Among those, 27% of posts are about image processing, and 21% are about interoperability. The `'stackalloc'` operator is the third category with 9% of unsafe code posts. The `stackalloc` operator allocates a block of memory on the stack. Since C# 7.2, Microsoft recommends against using `'stackalloc'` in unsafe context whenever possible. Manual inspection shows 67 code snippets with dangerous functions that can introduce vulnerability if not used with caution (e.g., buffer overflow).

REFERENCES

[1] 2020. Security Code Scan - static code analyzer for .NET. https://security-code-scan.github.io/

[2] Nikolaos Bafatakis, Niels Boecker, Wenjie Boon, Martin Cabello Salazar, Jens Krinke, Gazi Oznacar, and Robert White. 2019. Python coding style compliance on stack overflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 210–214.

[3] Mark Borg. 2017. C++/C# interoperability. https://mark-borg.github.io/blog/2017/interop

[4] c sharpcorner. 2020. understanding-unsafe-code-in-C-Sharp. https://docs.microsoft.com/en-us/dotnet/standard/security

[5] M. Chen, F. Fischer, N. Meng, X. Wang, and J. Grossklags. 2019. How Reliable is the Crowdsourced Knowledge of Security Implementation?. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 536–547.

[6] TIOBE Company. 2020. TIOBE Index for January 2019. https://www.tiobe.com/tiobe-index/

[7] E. Firouzi and A. Sami. 2019. Visual Studio Automated Refactoring Tool Should Improve Development Time, but ReSharper Led to More Solution-Build Failures. In *2019 IEEE Workshop on Mining and Analyzing Interaction Histories (MAINT)*. 2–6.

[8] Adam Freeman and Allen Jones. 2003. *Programming. NET Security: Writing Secure Applications Using C# or Visual Basic. NET.* " O'Reilly Media, Inc.".

[9] PYPL Index. 2020. PYPL PopularitY of Programming Language. http://pypl.github.io/PYPL.html

[10] Brian A LaMacchia, Sebastian Lange, Matthew Lyons, Rudi Martin, and Kevin T Price. 2002. *NET framework security*. Addison-Wesley Reading.

[11] N. Meng, S. Nagy, D. Yao, W. Zhuang, and G. Arango-Argoty. 2018. Secure Coding Practices in Java: Challenges and Vulnerabilities. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 372–383.

[12] Joe McManus MGR. 2018. SEI CERT Java Coding Standard. https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java

[13] Microsoft. 2015. Unsafe code and pointers. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/unsafe-code-pointers/

[14] Microsoft. 2016. CA1060: Move P/Invokes to NativeMethods class. https://docs.microsoft.com/en-us/visualstudio/code-quality/ca1060?view=vs-2019

[15] Microsoft. 2020. Interoperability (C# Programming Guide). https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/interop/

[16] Microsoft. 2020. Microsoft Security. https://docs.microsoft.com/en-us/dotnet/standard/security

[17] microsoft. 2020. unsafe-code-pointers. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/unsafe-code-pointers

[18] Andrew C Myers. 1999. JFlow: Practical mostly-static information flow control. In *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 228–241.

[19] Christian Nagel. 2018. *Professional C# 7 and. NET Core 2.0.* John Wiley & Sons.

[20] NICOLAS PORTMANN. 2019. Unsafe array access and pointer arithmetics in C#. https://ndportmann.com/system-runtime-compilerservices-unsafe/

[21] Sazzadur Rahaman, Ya Xiao, Sharmin Afrose, Fahad Shaon, Ke Tian, Miles Frantz, Murat Kantarcioglu, and Danfeng Yao. 2019. Cryptoguard: High precision detection of cryptographic vulnerabilities in massive-sized java projects. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2455–2472.

[22] Akond Rahman, Effat Farhana, and Nasif Imtiaz. 2019. Snakes in paradise?: Insecure python-related coding practices in stack overflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories*. IEEE, 200–204.

[23] Robert Schiela. 2020. SEI CERT C++ Coding Standard. https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

[24] Tushar Sharma, Marios Fragkoulis, and Diomidis Spinellis. 2017. House of cards: code smells in open-source C# repositories. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 424–429.

[25] stackexchange. 2020. Stack Exchange Directory Listing-Internet Archive. https://archive.org/download/stackexchange

[26] David Svoboda. 2018. SEI CERT C Coding Standard. https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard

[27] university of TARTU. 2020. imageprocessing. https://sisu.ut.ee/imageprocessing/book/1

[28] Morteza Verdi, Ashkan Sami, Jafar Akhondali, Foutse Khomh, Gias Uddin, and Alireza Karami Motlagh. 2019. An Empirical Study of C++ Vulnerabilities in Crowd-Sourced Code Examples. *arXiv preprint arXiv:1910.01321* (2019).

[29] J. Viega, J. T. Bloch, Y. Kohno, and G. McGraw. 2000. ITS4: a static vulnerability scanner for C and C++ code. In *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*. 257–267.

[30] Rajesh VS. 2019. pointers in C#. https://www.c-sharpcorner.com/article/pointers-in-C-Sharp/

[31] Chamila Wijayarathna and Nalin Asanka Gamagedara Arachchilage. 2019. Why Johnny can't develop a secure application? A usability analysis of Java Secure Socket Extension API. *Computers & Security* 80 (2019), 54–73.

[32] Chris Woodruff. 2018. C# 7.2 and 7.3 updates for stackalloc. https://blog.jetbrains.com/dotnet/2018/09/17/c-sharp-updates-for-stackalloc/