TPPE 000 Technical Presentation Proficiency Examination

Analysis of .NET Obfuscation Applications.

Shashank Sabhlok
Quality Assurance Developer
Savision Inc.
2B Candidate for B.A.Sc. in Electrical Engineering
2nd July 2015

Outline

Company & Problem Overview

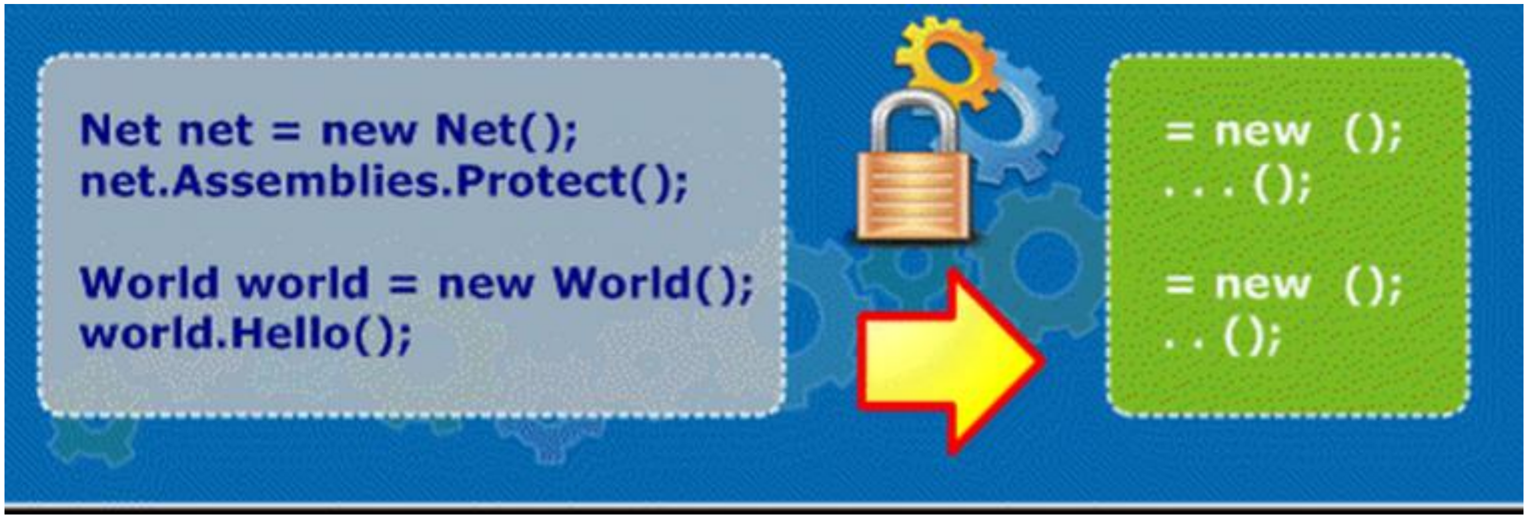
- Savision is a Dutch start-up, that provides IT monitoring solutions to commercial organizations.
- Their latest product was a HTML 5 application that transforms complex IT data into actionable and well organized dashboards.
- Problem : Prevent clients or competition from either going through or altering the .NET code of the application.
- Solution : Obfuscate the code.

savision

https://www.savision.com

What is Code Obfuscation ?

- Obfuscation is an anti-reverse engineering method to cipher the source code of a specific application.
- It doesn’t tamper with the execution of the intended logic, and yet makes it difficult for malicious attackers to decipher the code.
- Overcomes the drawbacks of classical security methods like server-side execution and code cryptography.



```
Net net = new Net();
net.Assemblies.Protect();

World world = new World();
world.Hello();
```

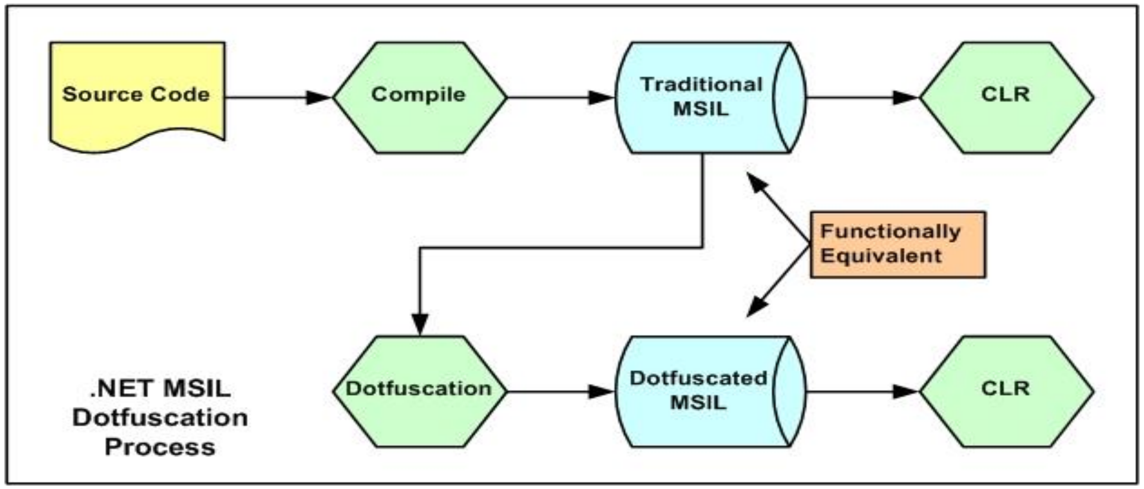
```
= new  ();
. . . ();

= new  ();
. . ();
```

http://www.codekicks.com/2008/02/net-obfuscation-using-dotfuscator-for.html

Why is it necessary for .NET ?

- Programmers coding in .NET compile their program into intermediate code called Common Intermediate Language in a portable execution file which is then managed and run by the Common Language Runtime.
- The .NET code that is run under CLR is compiled into language independent intermediate code (Microsoft Intermediate Language), which at compilation time can be extracted and decompiled using tools like ILDASM (Microsoft's MSIL disassembler).



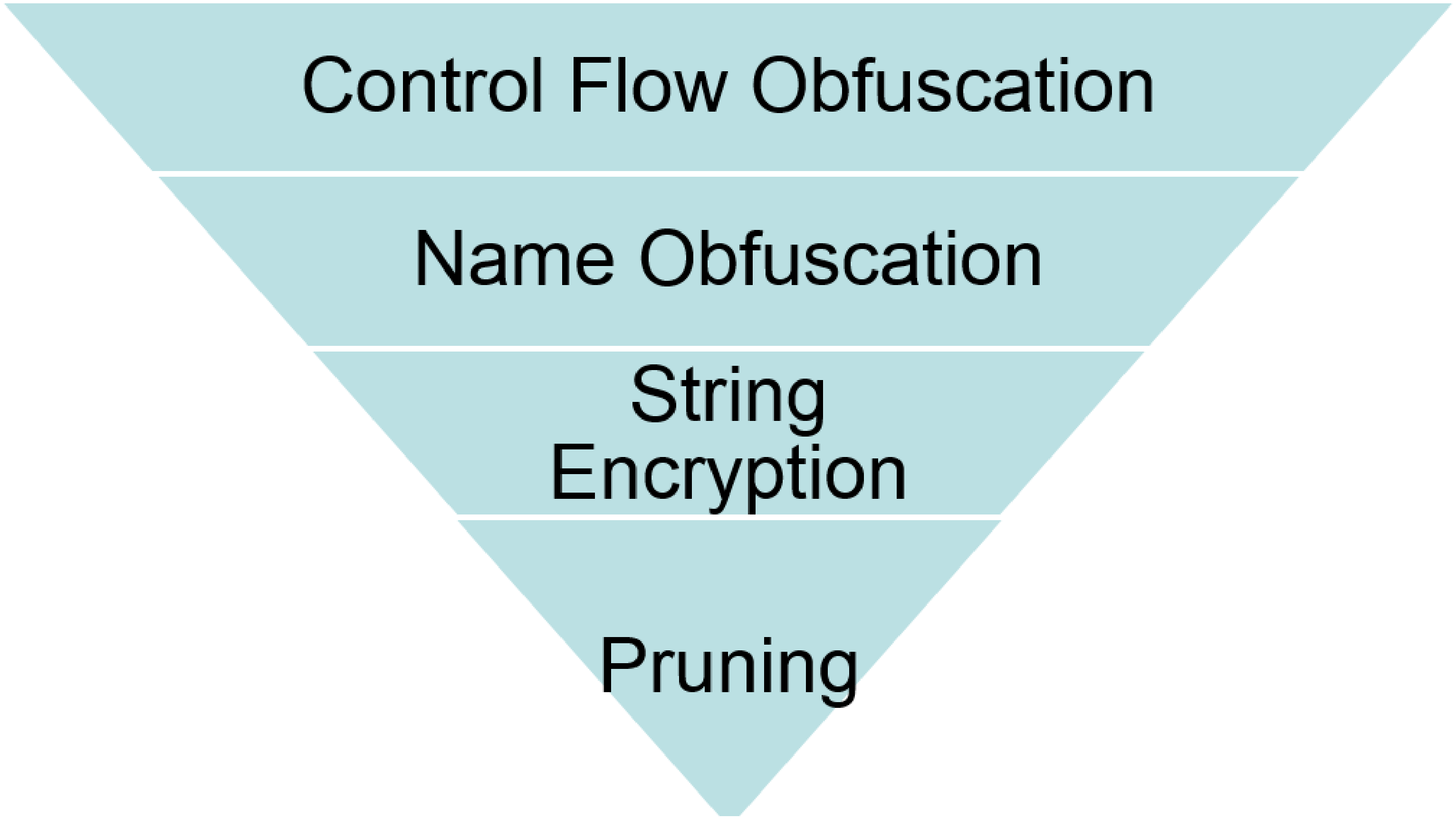
http://en.helpdoc-online.com/dotfuscator_3.0/source/dotf68q6.htm

Why is it necessary for .NET ?

This simple decompilation process can give attackers the freedom to analyze the entire software and subsequently do the following:

- Expose Software Licensing code and Business Logic.
- Expose secrets in code including passwords and encryption keys.
- Exploit security flaws.
- Enable and disable certain functionality.
- Modify the application logic and add backdoors to the original code.

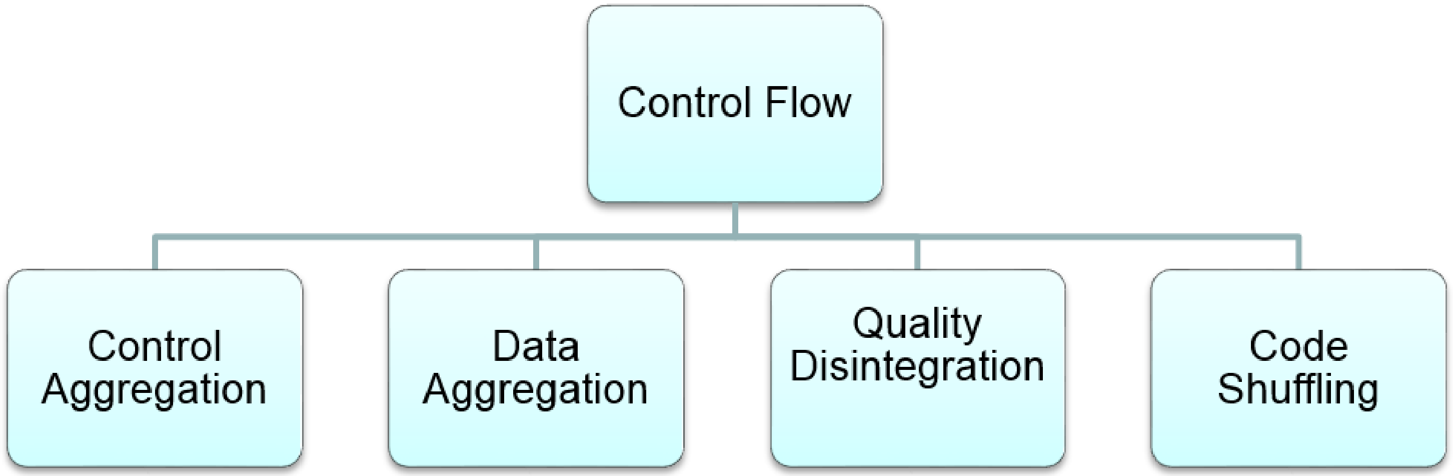
Features and Requirements of the Application



Control Flow Obfuscation

- This process produces executable logic, but populates the code with unnecessary iterative and conditional constructs in the code, hence making it very difficult to analyze upon decompilation.

- The result is that decompilers are unable to reconstruct your code. Most of the times they crash while trying to do so.



Control Flow Obfuscation - Example

Before Obfuscation

```
public int CompareTo(Object o) {

   int n = occurrences -
        ((WordOccurrence)o).occurrences;

   if ( n == 0 ) {
       n = String.Compare(
             word, ((WordOccurrence)o).word );
   }

   return(n);
}
```

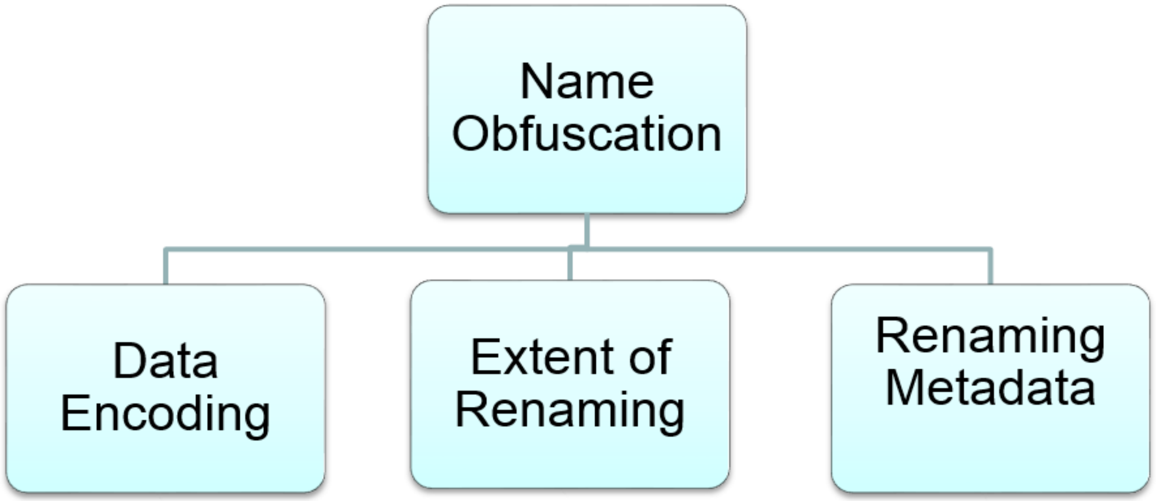
After Obfuscation

```
public virtual int _a(Object A_0) {
   int local0;
   int local1;
   local 10 = this.a - (c) A_0.a;
   if (local0 != 0) goto i0;
   while (true) {
       return local1;
       i0: local1 = local10;
   }
   i1: local10 =
   System.String.Compare(this.b,(c));
   goto i0;
}
```

Code Snippets from [1].

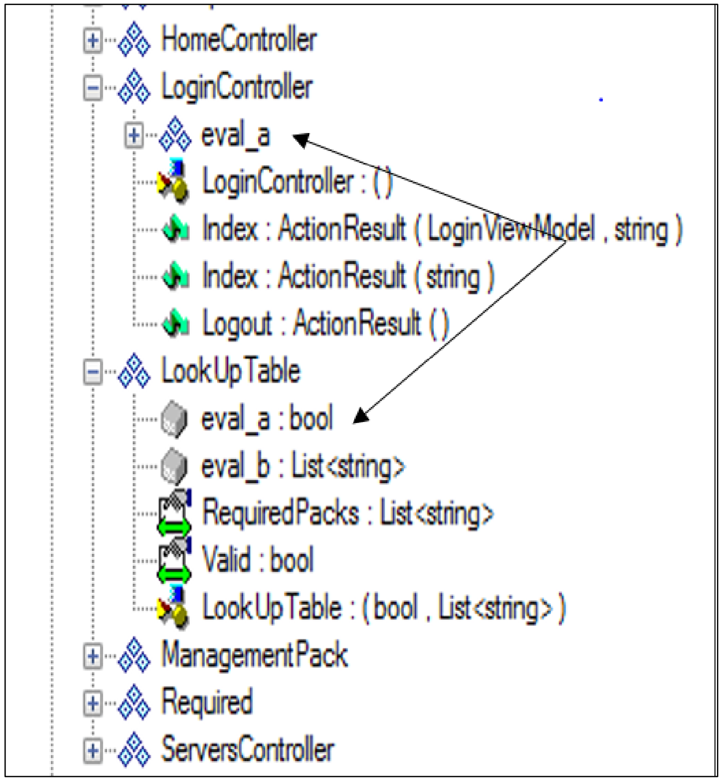
Name Obfuscation

- Name Obfuscation is usually considered the first line of defense.
- Makes code less documenting by replacing telling names with non-meaningful ones.
- For example, ‘*Home_View.cshtml’* clearly indicates the content of the file – the front-end logic of the homepage of the application. Good obfuscators pick up on this.



Name Obfuscation

- Data Encoding : This process simply adds unnecessary constructs and operators to declarations and indexes for different data structures e.g. `Array[i] → D[7(i-8)/2]`.
- Extent of renaming : An obfuscator can rename several function, function parameters, variables and classes to the same name e.g. `a(int a)` is called by `d("zzz")`.
- Renaming Metadata : Metadata are the .dll files created upon compilation. These can be renamed to add another barrier between an attacker and the code.


HomeController
LoginController
eval_a
LoginController : ()
Index : ActionResult (LoginViewModel , string)
Index : ActionResult (string)
Logout : ActionResult ( )
LookUpTable
eval_a : bool
eval_b : List<string>
RequiredPacks : List<string>
Valid : bool
LookUpTable : (bool , List<string>)
ManagementPack
Required
ServersController

Name Obfuscation - Example

Before Obfuscation	After Obfuscation

`private void CalcPayroll (SpecialList employeeGroup) {`
 	
`while(employeeGroup.HasMore()) {`
 	
` employee =employeeGroup.GetNext(true);`
 	
` employee.UpdateSalary();`
 	
` DistributeCheck(employee);`
` }`
`}`
	`private void a(a b) {`
	
	`while (b.a()) {`
	
	`  a = b.a(true);`
	
	`  a.a();`
	
	`  a(a);`
	`  }`
	`}`

Code Snippets from [1].

String Encryption

- Literal strings present in .NET assemblies contain sensitive information, and are in plain view.
- They can contain login information, passwords, parameters or SQL Queries.
- For example, if someone wants to remove time locks they can easily search for instances of strings like “timeout”, “expired", "session” etc.
- Obfuscation encrypt strings by replacing strings with either a series of random characters or function calls.

String Encryption- Example

Before Obfuscation

```
internal static string CheckLicense(string
LicenseKey){

 if(LicenseKey="ABCD")
  {
    return "Valid License";
  }
  return "Invalid License";
}
```

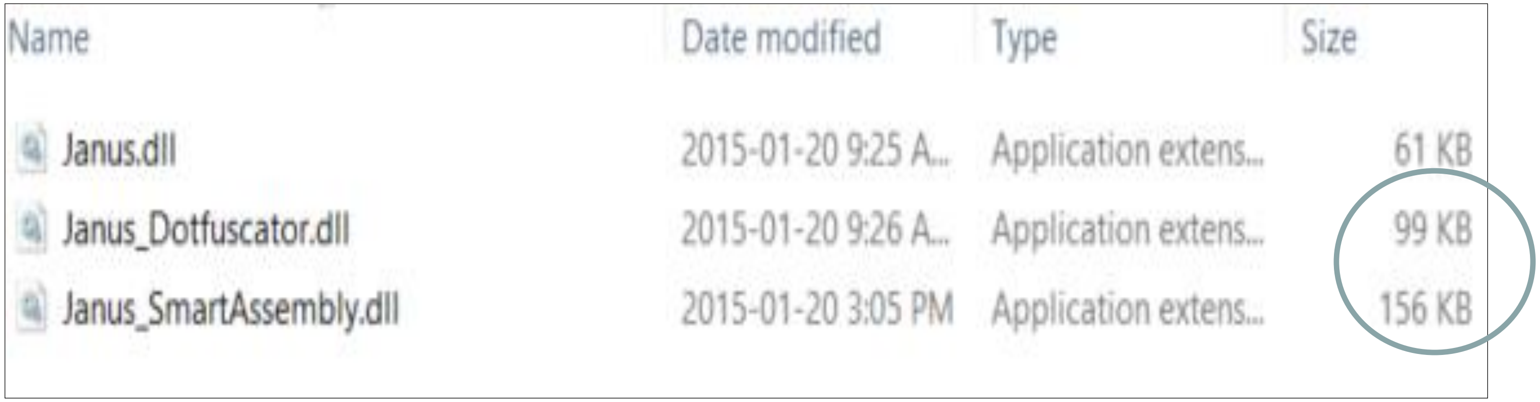
After Obfuscation

```
internal static string CheckLicense(string
LicenseKey)
{
 if(LicenseKey ==
A70f.A9f4(0x3d84e419,0xf504de2,0x126777ba8))
 {
   return
A70f.A9f4(0x3d84e419,0xf504de2,0x126777ba);
 }
   return
A70f.A9f4(0x1cfafefd,0xf50dff2,0x126777be);
}
```

Code Snippets from [2].

Size Reduction

- Makes the deployment of the application on client’s server quicker.
- There is always extra code in reusable libraries and classes that considers conditions or cases that are not necessary.
- Removal of this code results in a significantly smaller file size.



Name	Date modified	Type	Size
Janus.dll	2015-01-20 9:25 A...	Application extens...	61 KB
Janus_Dotfuscator.dll	2015-01-20 9:26 A...	Application extens...	99 KB
Janus_SmartAssembly.dll	2015-01-20 3:05 PM	Application extens...	156 KB

Candidates and Comparison

- After researching several products, two obfuscation applications were shortlisted – SmartAssembly and Dotfuscator.
- When comparing, the same .dll file was obfuscated twice and the results were compared on the same disassembler – JetBrains dotPeek.
- Dotfuscator by PreEmptive Solutions is a Microsoft backed product, and therefore can easily integrate into Microsoft Visual Studio.
- SmartAssembly by Red-Gate is the more popular of the two and is known to provide good value for its price

			✓	✓		✓
Name	**Price**	**Tamper Defense**	**Renaming**	**String Encryption**	**Control Flow Obfuscation**	**Size Reduction**
Dotfuscator	$2000	yes	yes	yes	yes	yes
Smart Assembly	$993–$1493	no	yes	yes	yes	yes

Table 1 : Comparison of Dotfuscator and SmartAssembly. From [3].

Comparison of Control Flow Obfuscation

- The major difference was Control Flow Obfuscation. Consider a dictionary declaration in C# :

```
public Dictionary<DashboardsKey, Dashboards> Dashboards1 { get; set;}
```

✓

SmartAssembly	Dotfuscator

SmartAssembly

```
public Dictionary<DashboardsKey, Dashboards>Dashboards1
{
  get
  {
     return this._;
  }
  set
  {
     this._ = value;
  }
}
```

Dotfuscator

```
public Dictionary<DashboardsKey,Dashboards>Dashboards1{
 get
  {
   Dictionary<DashboardsKey,Dashboards> dictionary;
   int i = -28946;
   i = -28949;
   switch (i == i)
   {
     case 1:
     goto label_0;
     i = 0;
     goto label_0;
     dictionary = eval_b;
     break;
   }
   return dictionary;
}
```

Summary

- Explored the need to obfuscate .NET applications.
- Considered the key features and of an obfuscator
  - Control Flow Obfuscation
  - Name Obfuscation
  - String Encryption
  - Size Reduction.
- Control Flow Obfuscation is the most crucial aspect of an obfuscation application.
- Dotfuscator was better than SmartAssembly due to extensive amounts of Control Flow Obfuscation present.

References

- [1] : 8 Ways To Protect And Obfuscate Your .NET Code Against Reverse-Engineering Using Crypto Obfuscation, Pre-Emptive Solutions, [Online] 2015, http://www.preemptive.com/products/dotfuscator/features

- [2] : WHY YOUR .NET APPLICATIONS NEED PROTECTION?, Code-Wall Technologies,[Online] 2015, http://www.codewall.net/

- [3]: List of obfuscators for .NET, Wikipedia, [Online] 2015, https://en.wikipedia.org/wiki/List_of_obfuscators_for_.NET

Thank You for your attention !
Questions ?