SECURITY BOOT CAMP FOR .NET DEVELOPERS

Philippe Arteau
Security Researcher

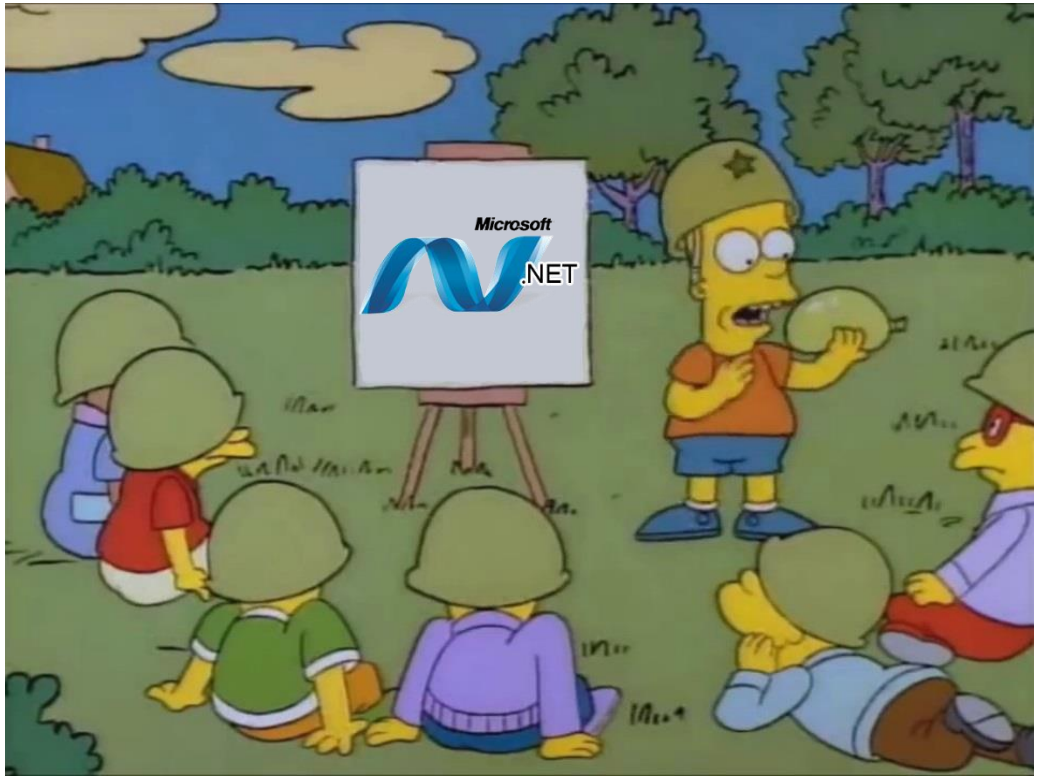
ConFoo.CA

GoSecure

06/12/2017

AGENDA

- Introduction
- Methodology for Code Review
- Vulnerabilities in .NET Context
  - Path Traversal
  - XSS
  - Cryptography
  - Hardcoded secret
- Automating Checks
  - Visual Studio / MsBuild
- Recent Trends
  - Deserialization
  - Double Parsing
- Conclusion



INTRODUCTION

WHY .NET SPECIFIC?

- Agnostic presentations are excellent for high level explanation
- By focusing on one language, we can highlight specific components
  - Deep-dive in C# code samples
  - Framework specific features
  - Tools specific
- Every platform have advantages and weaknesses
- Because there is only 45 minutes :)

SECURITY CODE REVIEW

- Code review is the systematic examination of source code[1] with the specific goal of findings security bugs.

- Security Bugs?
  - Injections
  - XSS
  - Cryptographic weakness
  - Logic flaw
  - And many more...

[1] Wikipedia: Code review

WHY SECURITY CODE REVIEW?

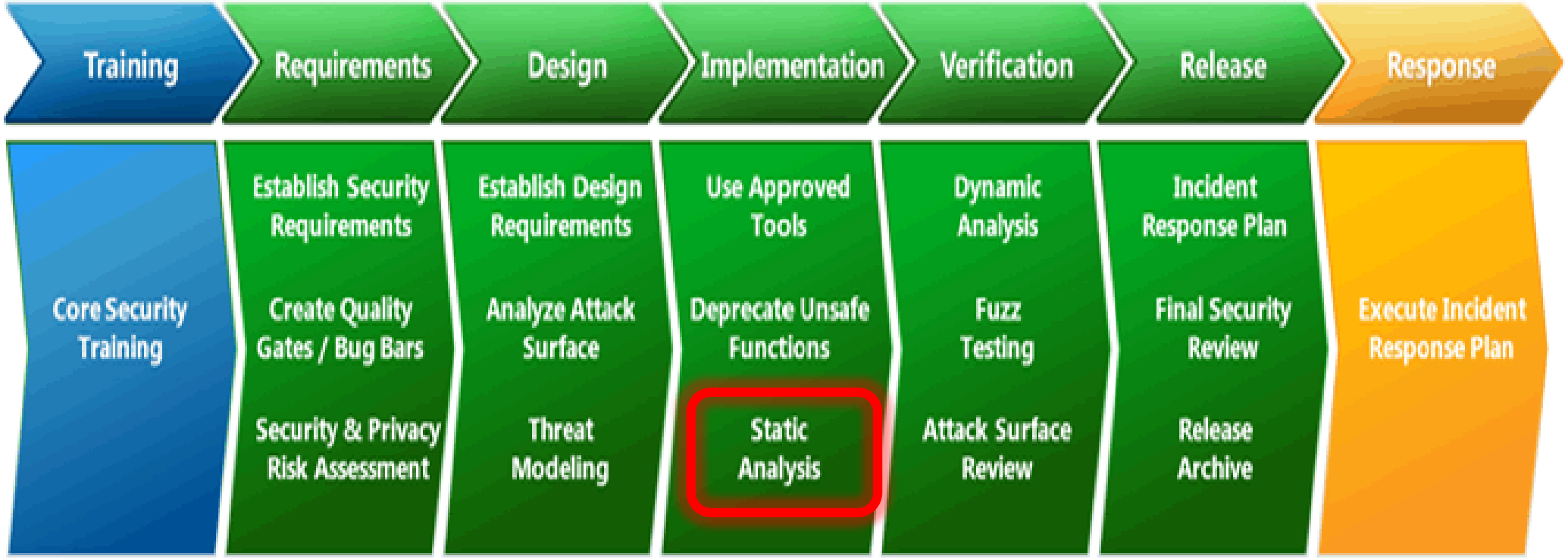
- Complementary to dynamic techniques (penetration testing, fuzzing, etc.)
- Every technique has its advantages

- Code review advantages:
  - Coverage
  - Finding all instances of a vulnerability
  - Accessible activity for developer
  - Excellent for doing defense in depth

METHODOLOGY FOR CODE REVIEW

CODE REVIEW IN SDLC

- First thing first, code review is ONE of the security activities thad need to be integrated in the development lifecycle.



CODE REVIEW STEPS

1. Threat modeling [1]
2. Analysis
3. Reporting (Document or Opening ticket) *
4. Bug fixing *

[1] OWASP Code Review Guide v2 p.32

* Not covered in this presentation

THREAT MODELING : DECOMPOSING THE APPLICATION

- Identify assets to protect
  - Personal information
  - Documents
  - Passwords
- Identify entry points
  - MVC Controller
  - Web Services
  - Forms
- Identify external dependencies
- Imagine possible threats (STRIDE : **S**poofing, **T**ampering, **I**nformation **D**isclosure, **D**enial of Service, **E**levation of privilege)

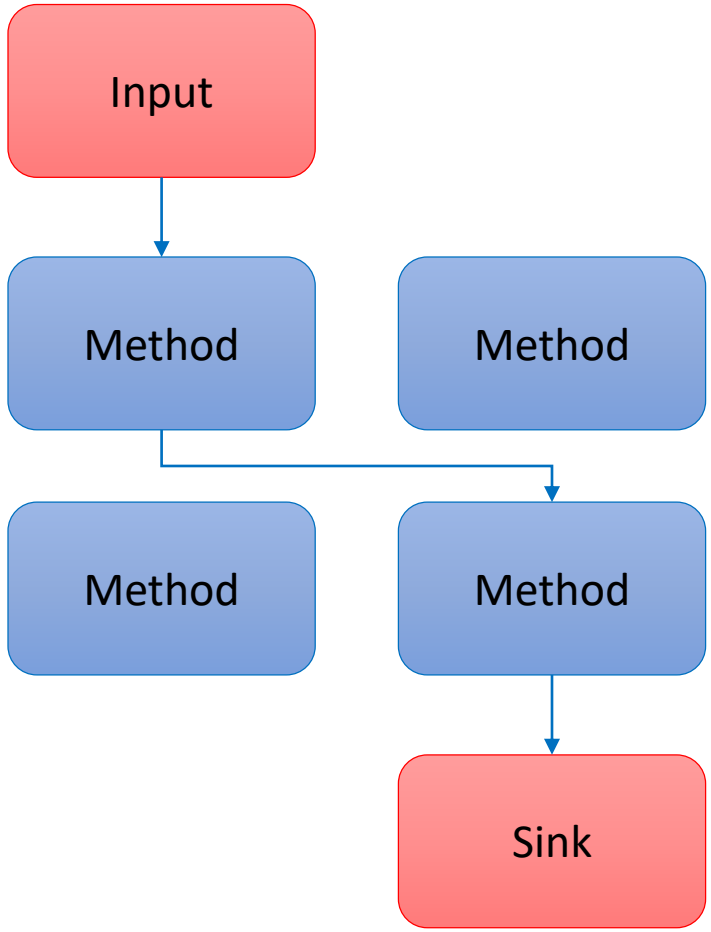
ANALYSIS

1. Tools configuration
2. Automate scan
   - Review potential issues
3. Manual review

Static analysis tools can also be run in parallel with the manual review.

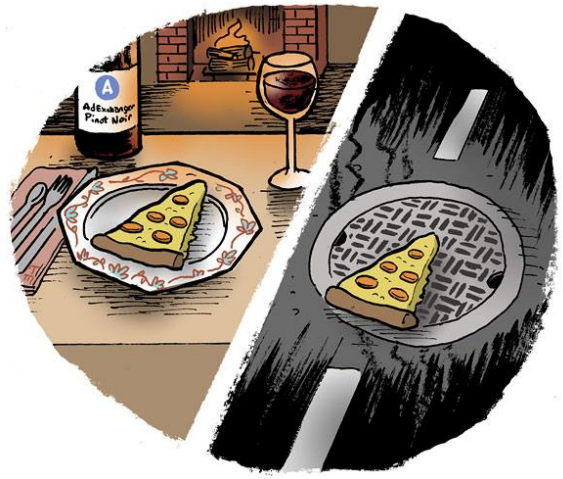
DATAFLOW

- Mapping between inputs and APIs
- Categories of bugs
  - Injection
  - Path traversal
  - Static IV (Cryptography)
  - Deserialization



CONTEXT

- APIs are not vulnerable by default
- APIs are designed to be used in a certain context
- Categories of bugs
  - Random number generation
  - Oracle Padding Attack or any other active attack
  - Control based on Host header
  - Insecure communication (internal communication vs network communication)
  - Configuration files vs Upload files



Context Matters

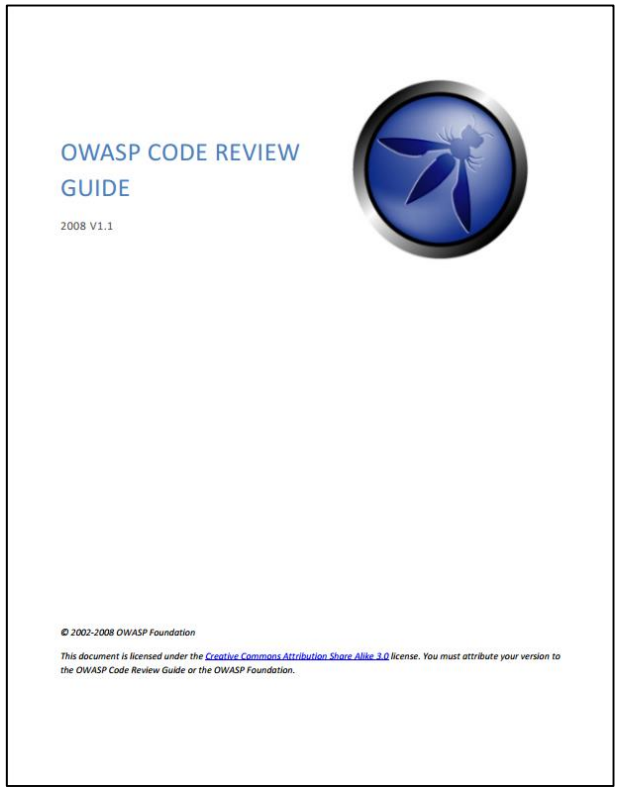
CHECKLIST

- Intended for baseline verifications
- Guidelines
- Reproducibility

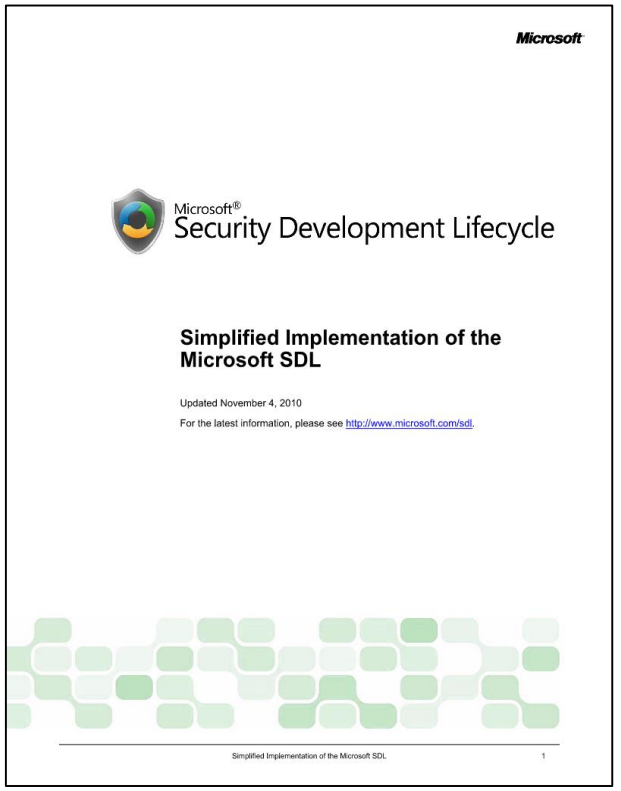
#	Description	1	2	3	Since
5.1	Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.	✓	✓	✓	1.0
5.3	Verify that server side input validation failures result in request rejection and are logged.	✓	✓	✓	1.0
5.5	Verify that input validation routines are enforced on the server side.	✓	✓	✓	1.0

Listing taken from: [OWASP Application Security Verification Standard Project]

GOOD RESOURCES



[Code Review Guide]



[Development Lifecycle]

[Verification List]

VULNERABILITIES IN .NET CONTEXT

GoSecure

PATH TRAVERSAL

- SQL injection are *easy* to manage with **prepare statement**
- **Path traversal** is a source of injection that is often overlooked

- When does it matter?
  - File upload (writing to filesystem)
  - Document loading (reading to filesystem)
  - Can be applied in rare cases to URL [1]

[1] Example: https://sakurity.com/blog/2015/03/15/authy_bypass.html

CROSS-SITE SCRIPTING (XSS)

- Encoding with Razor template is usually secure
  - HTML entities are escaped by default

Special cases

- Use of `@Html.Raw()`
- Placing values in JavaScript **/!**
- JavaScript client-side template

PADDING ORACLE AND INTEGRITY

- .NET Framework provided symetric encryption primitive
  - Namespace System.Security.Cryptography
  - Include the mode: CBC, ECB, OFB, …
  - Does not provided integrity

HARDCODED PASSWORD

- Password
- Service account
- API keys

- Store value in configuration
- Encrypt the value

Identity Server

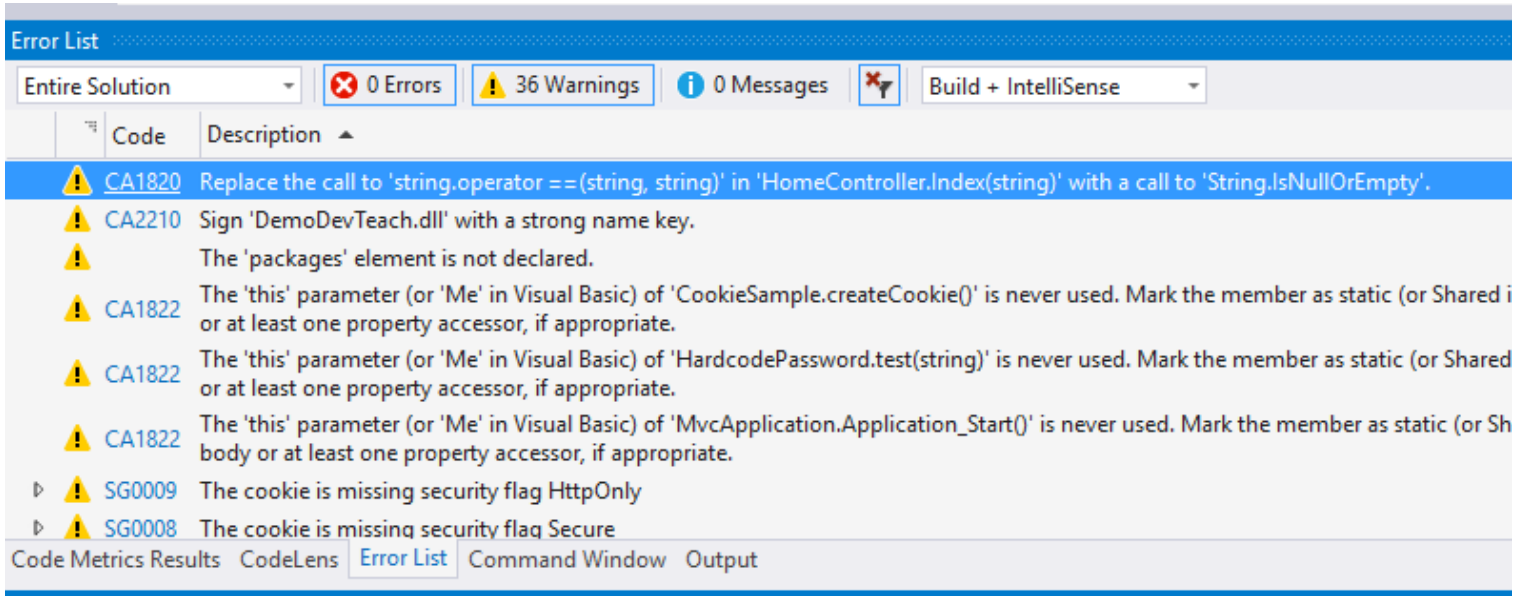
```
new Client
 {
      ClientId = "client",
      AllowedGrantTypes = GrantTypes.ClientCredentials,
      ClientSecrets =
      {
         new Secret("secret".Sha256())
      },
      AllowedScopes = { "api1" }
   }
```

AUTOMATING CHECKS

AUTOMATE CODE ANALYSIS

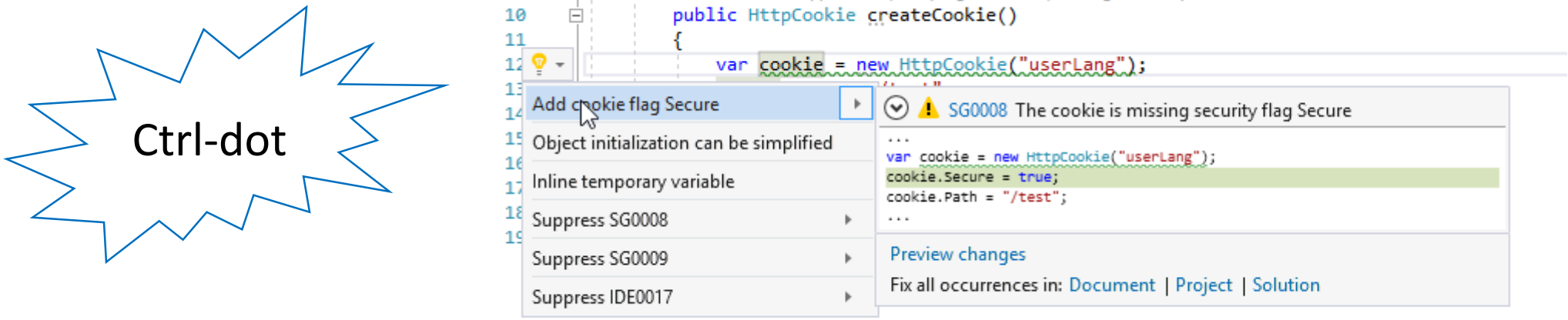
- Identifying bugs and vulnerabilities is nice but…

```
0 references | Philippe Arteau, 3 days ago | 1 author, 1 change
public class HomeController : Controller
{
    0 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
    public ActionResult Index(string input)
    {
        if (input == "") {
            return View();
        }
```



AUTOMATE CODE REFACTORING

- Remediation is event better!



- Some vulnerabilities require high-level understanding of the application

RECENT TRENDS

JSON DESERIALISATION

- History repeats itself
  - 2016: Numerous Java application were found vulnerable to native deserialization
  - 2017: Researchers [1] found issues in .NET JSON serializer
    - Some libraries have issued updates
    - The vulnerability was called: JSON Friday 13th
- Two ingredients needed for a successful attack
  - Gadgets
  - Unsafe deserialization

[1] Alvaro Muñoz, Oleksandr Mirosh and James Forshaw

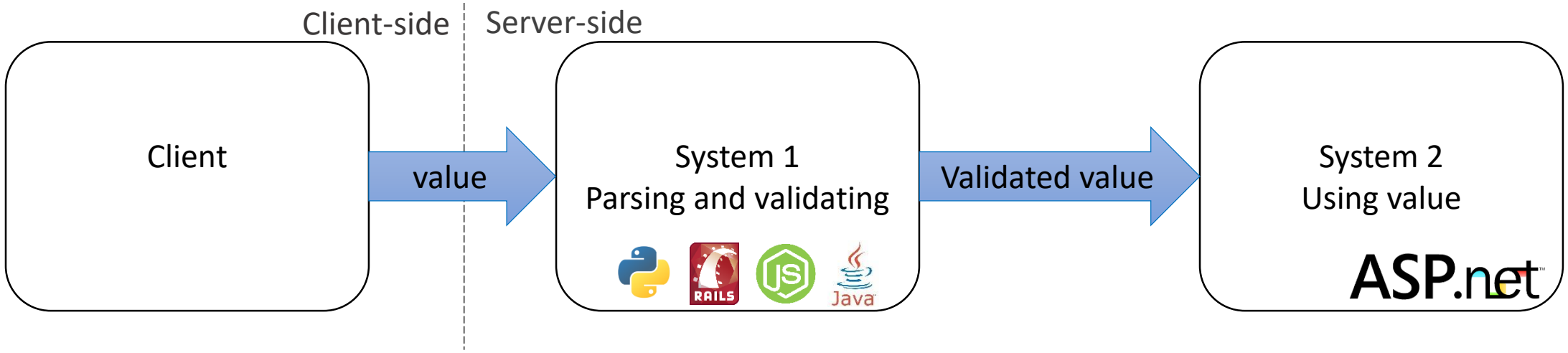
JSON DESERIALIZATION

Affected librairies

- FastJSON
- Json.NET (use of TypeNameHandling.All)
- FSPickler
- Sweet.Jayson
- JavascriptSerializer
- DataContractJsonSerializer

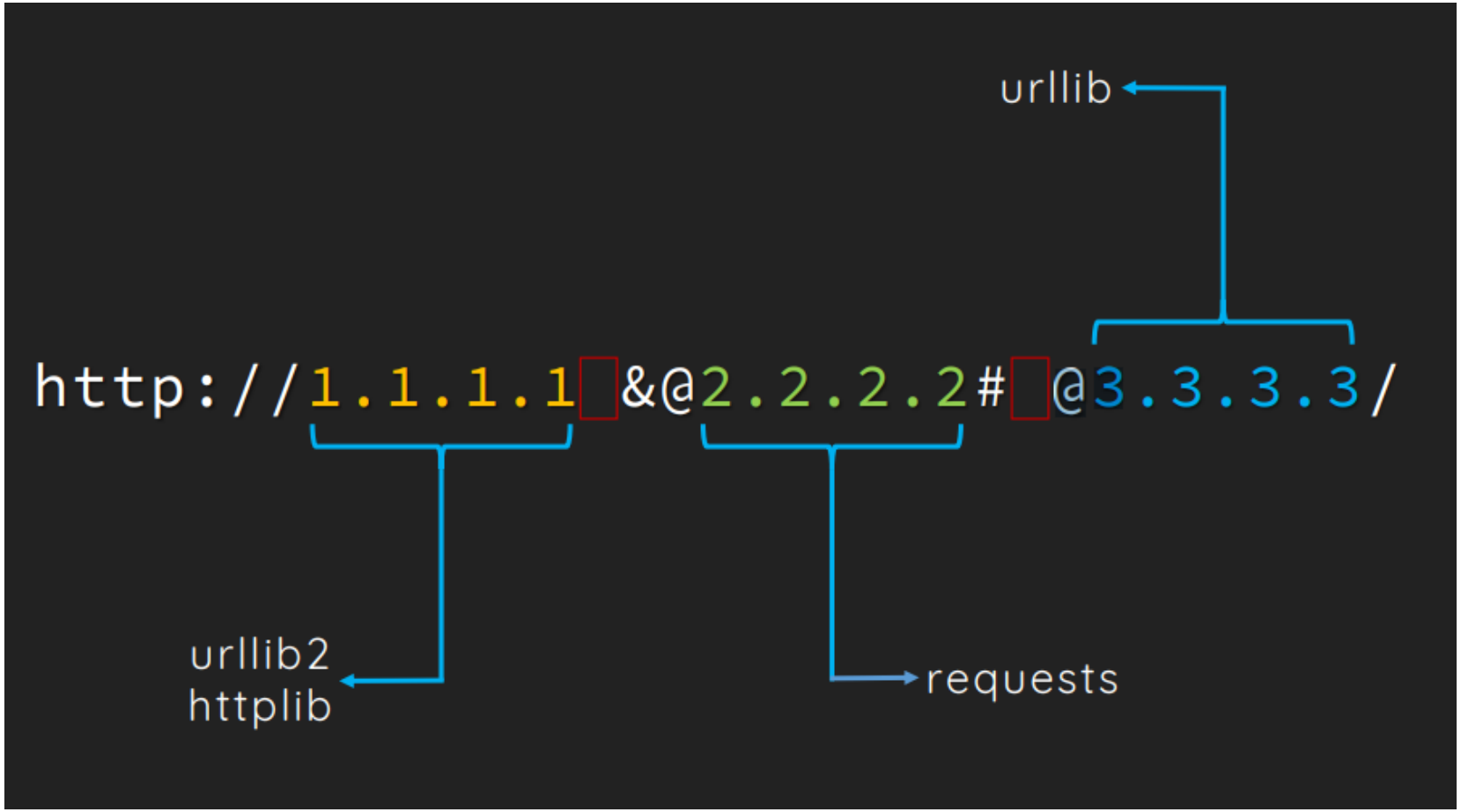
Ref: https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf

DOUBLE PARSING



- What if the **system validating** and **using the value** was not the same

DOUBLE PARSING: URLS



Reference: [A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!]

DOUBLE PARSING: URLS

- Less likely to happen in .NET
  - Small numbers of URI parser
- High probability when interacting **in other systems**
- DNS rebinding needed to be considered for host whitelisting

- Conclusion
  - Do not trust validated input that was parsed differently

FIND THE BUGS

```
{
  "username": "philippe",
  "fullname": "Philippe A.",
  "newPassword": “C0nf00"
}
```

IdentityValidator.cs

```
boolean IsValidRequest(json) {
    var jsonReader = JsonReaderWriterFactory.CreateJsonReader(json,[…]);
    var root = XElement.Load(jsonReader);

    return root.XPathSelectElement("//username ").Value == HttpContext.Current.User.Identity.Name;
}
```

UpdateUser.cs

```
void ProcessUpdateUser(json) {
   if(IsValidRequest(json)) {
       JObject user = JObject.Parse(json);

      var usersToUpdate = context.User.Where(u => u.username == user.GetValue(“username")).ToList();
      usersToUpdate.ForEach(u => u.Password = user.GetValue("newPassword"));
      context.SaveChanges();
   }
}
```

*Pseudocode is highly simplified

JSON PARSER IN .NET

```
{
 "username": "philippe",
 "username": "yannlarrivee",
 "fullname": "hihihi",
 "newPassword": “C0nf00"
}
```

using System.Runtime.Serialization.Json;

```
{
 "username": "philippe",
 "username": "yannlarrivee",
 "fullname": "hihihi",
 "newPassword": “C0nf00"
}
```

using Newtonsoft.Json;

```
{
 "username": "philippe",
 "username": "yannlarrivee",
 "fullname": "hihihi",
 "newPassword": “C0nf00"
}
```

Inspired by: https://justi.cz/security/2017/11/14/couchdb-rce-npm.html

REFERENCES

REFERENCES

- OWASP .NET Project

https://www.owasp.org/index.php/Category:OWASP_.NET_Project

- .NET Security Cheat Sheet

https://www.owasp.org/index.php/.NET_Security_Cheat_Sheet

- .NET Security Guard

https://dotnet-security-guard.github.io/

ROSLYN REFERENCES

- .NET Compiler Platform ("Roslyn"): Analyzers and the Rise of Code-Aware Libraries

https://www.youtube.com/watch?v=Ip6wrpYFHhE

- Roslyn Wiki

https://github.com/dotnet/roslyn/wiki

- Learn Roslyn Now: Part 10 Introduction to Analyzers by Josh Varty

https://joshvarty.wordpress.com/2015/04/30/learn-roslyn-now-part-10-introduction-to-analyzers/

- .NET Compiler Platform SDK

https://marketplace.visualstudio.com/items?itemName=VisualStudioProductTeam.NETCompilerPlatformSDK

.NET JSON DESERIALIZATION

- Ysoserial.net : Payload generator

https://github.com/pwntester/ysoserial.net

- Friday The 13th JSON Attack - White Paper

https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf

QUESTIONS ?

Contact

parteau@gosecure.ca

gosecure.net/blog/

@h3xStream @GoSecure_Inc