Writing Secure Code
Fundamentals and Coding Techniques

secure: [si-'kyur]
1: free from danger
2: free from risk of loss
3: affording safety

DI Andreas Schabus
aschabus@microsoft.com
Academic Relations Manager
Microsoft Österreich GmbH

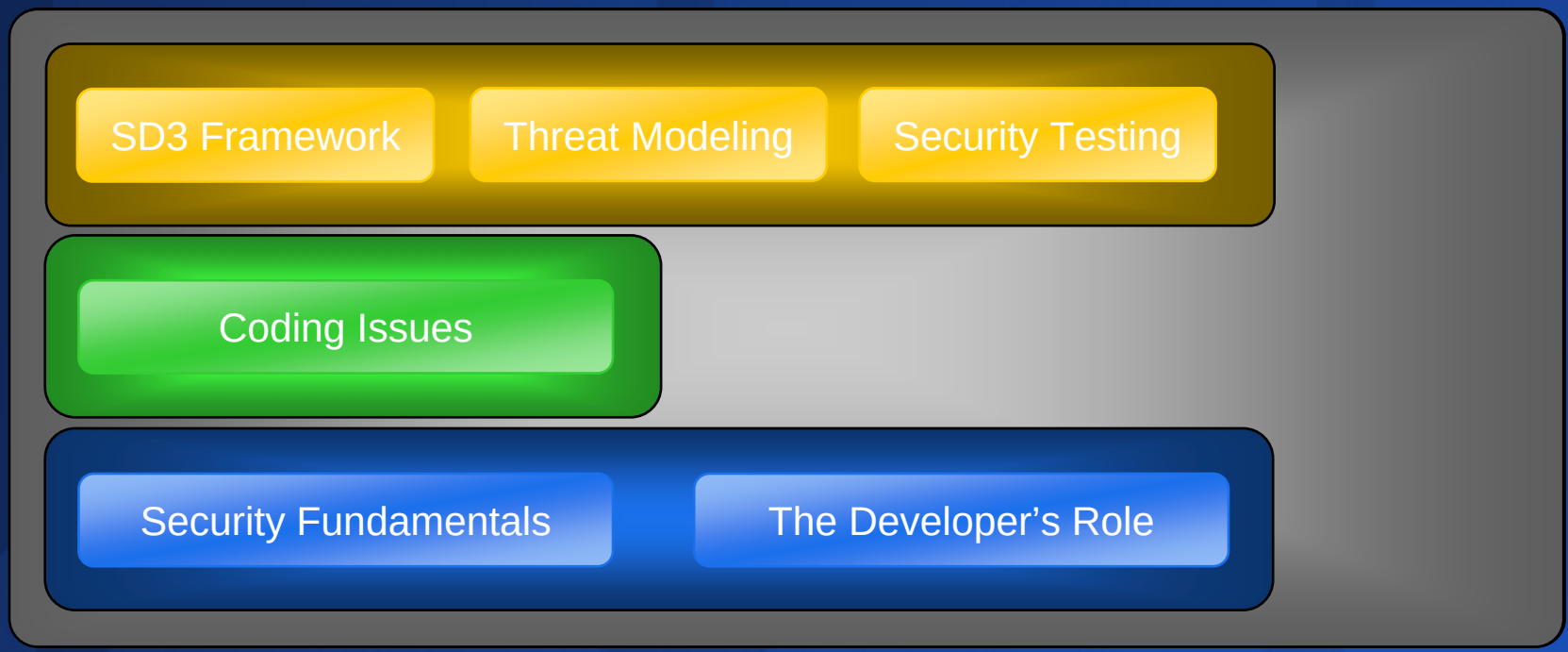
Security

Security: [si-'kyur-&-tE]

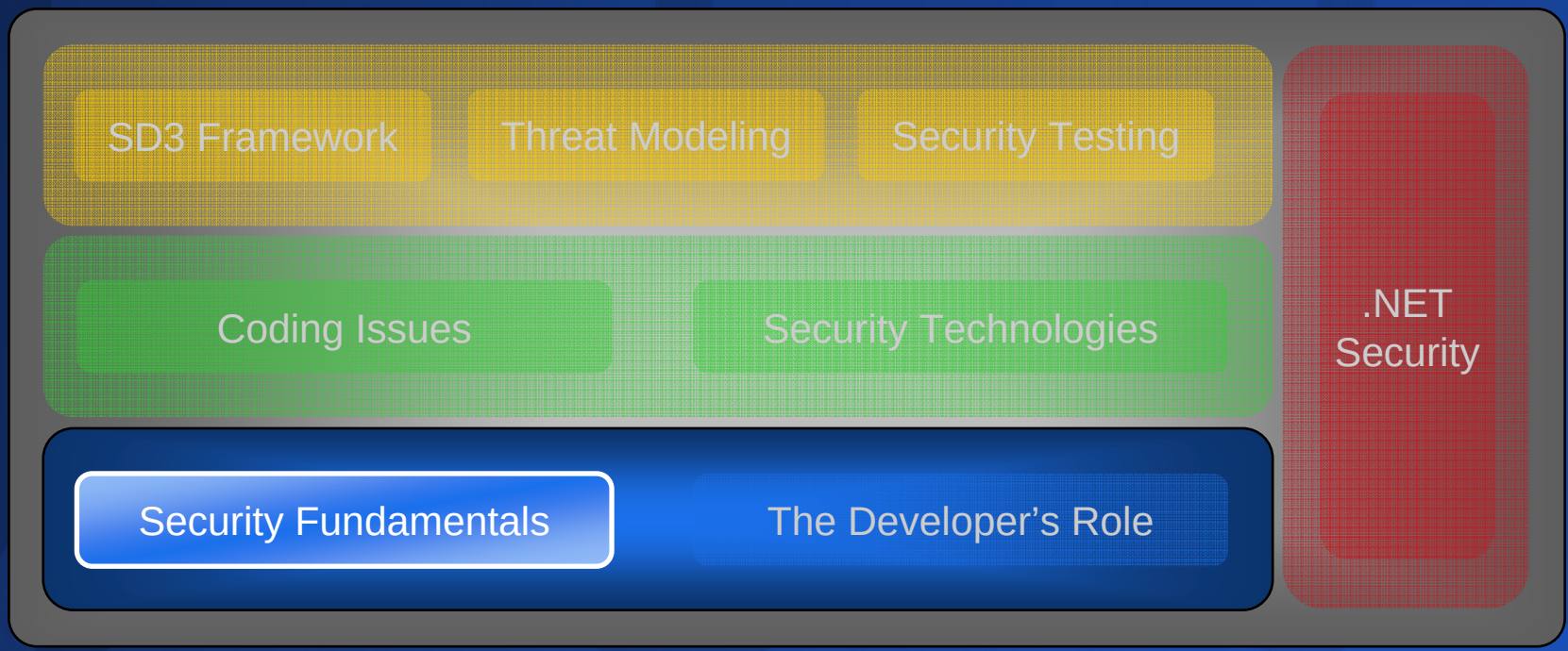
1: the quality or state of being secure
2: freedom from danger
3: measures taken to guard against espionage or sabotage, crime, attack, or escape

- Reality
- Call to action
  - Holistic approach
  - Raising the bar
  - Being as secure as possible
  - Security as Research Topic

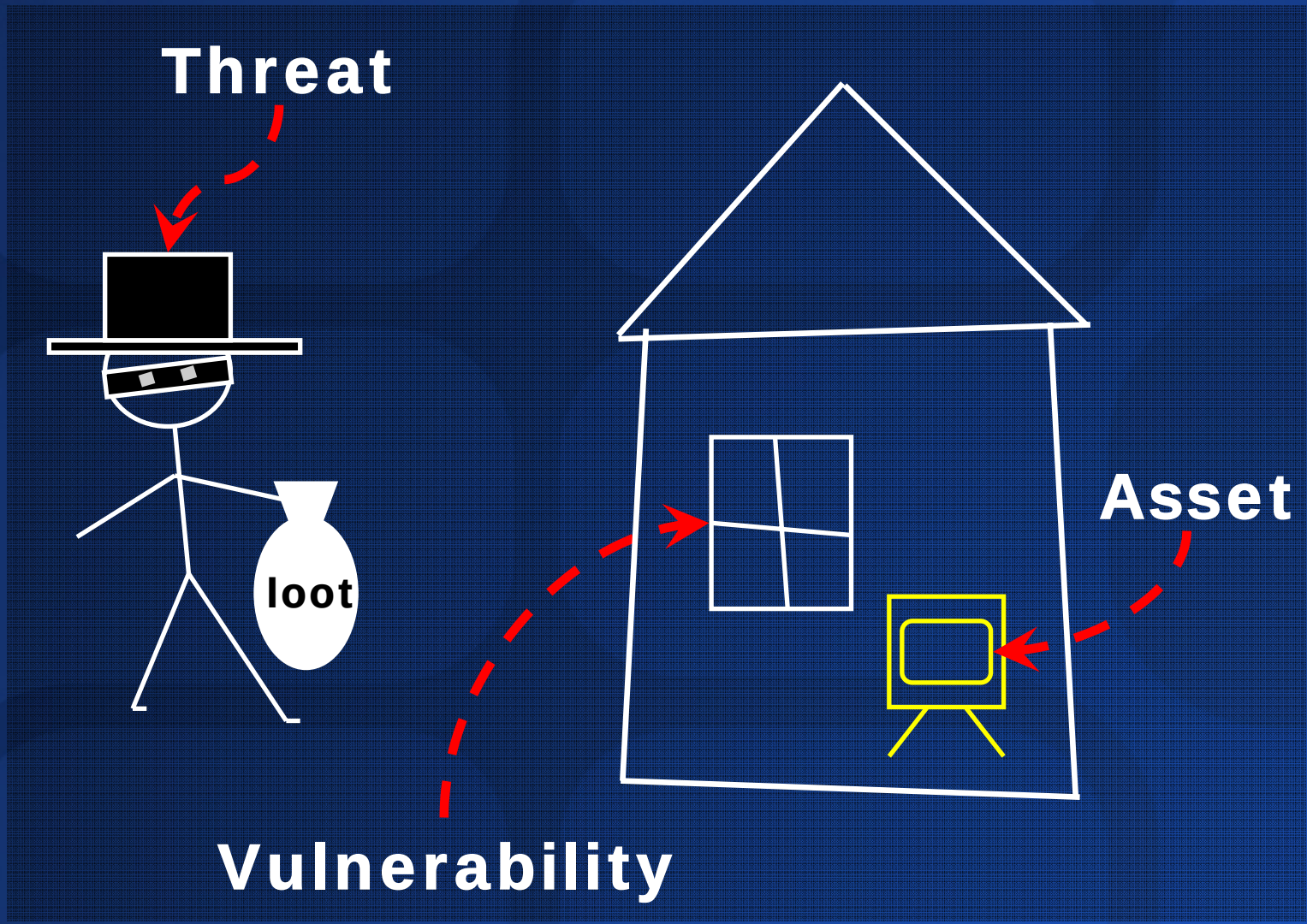
Writing Secure Code



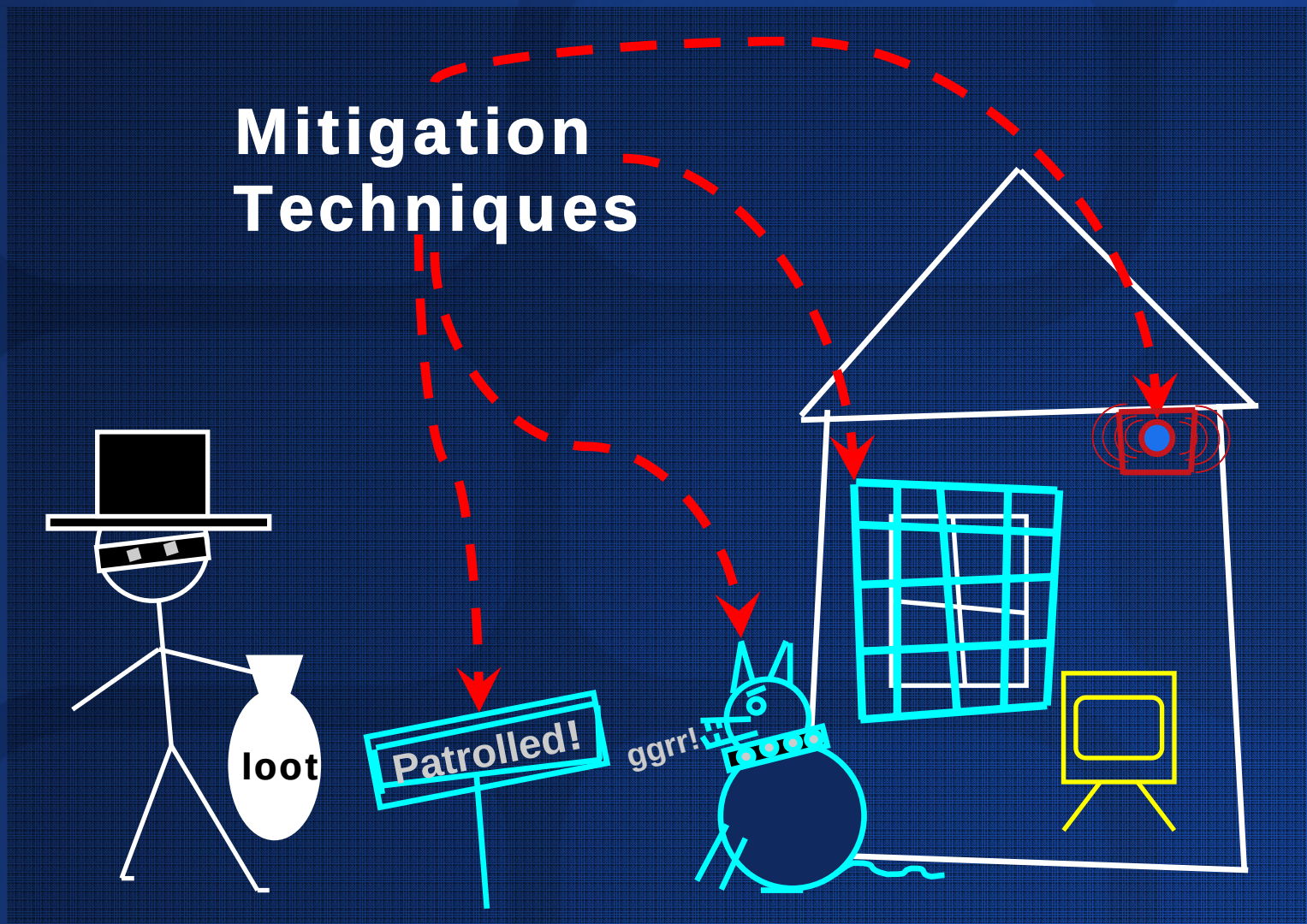
Security Fundamentals



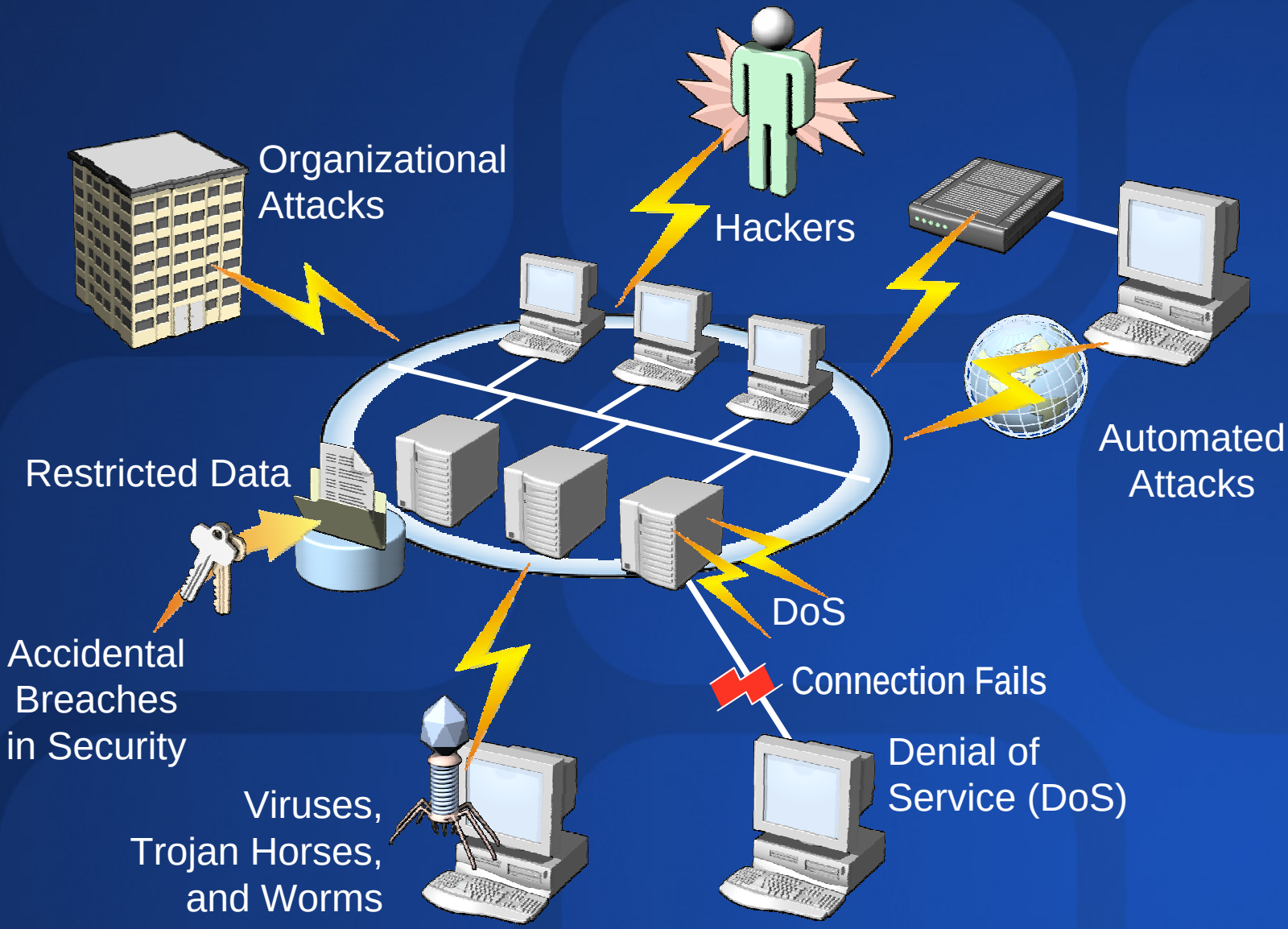
Common Security Terms (1/2)



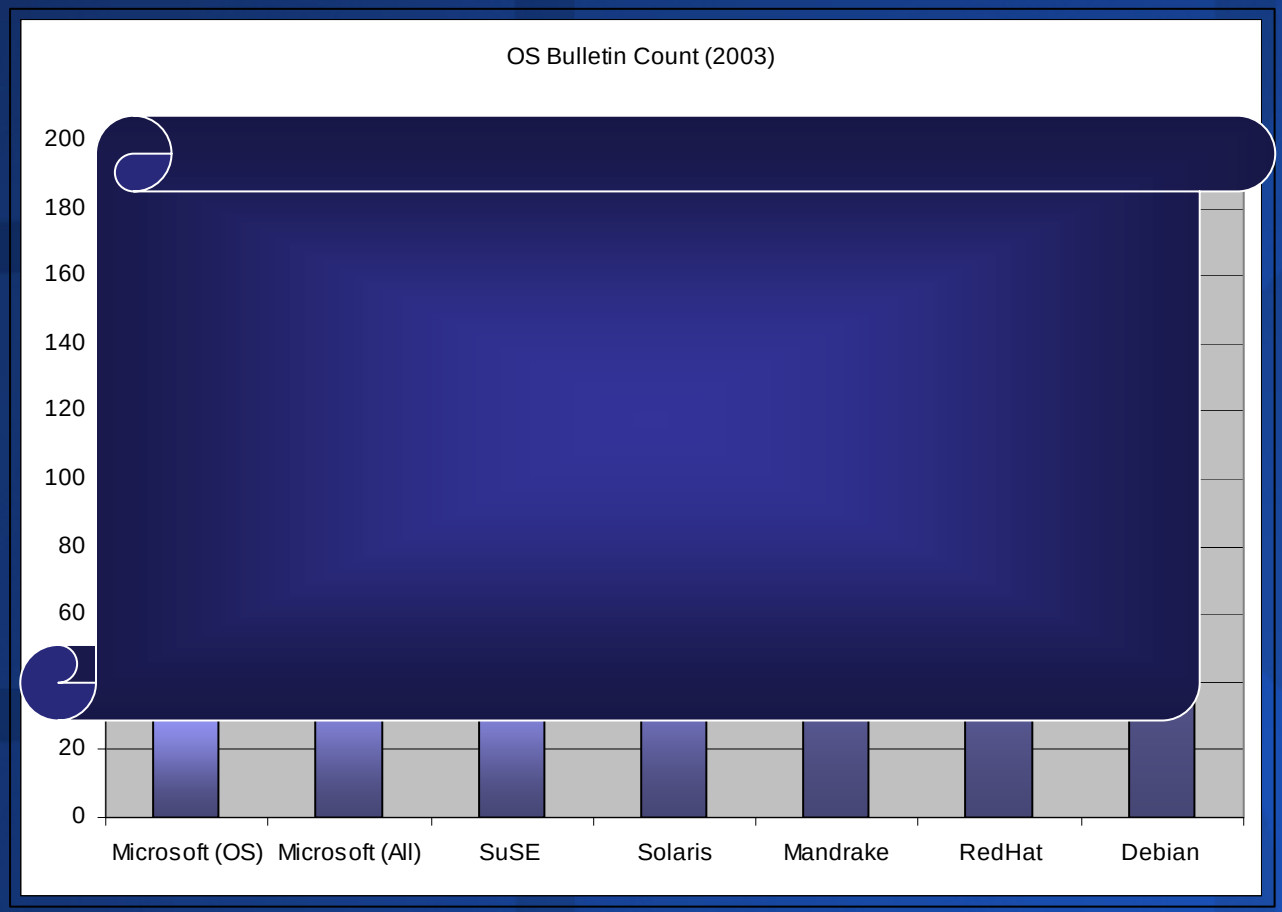
Common Security Terms (2/2)



Common Types of Attack



What Sort Of Figures Are We Talking About?



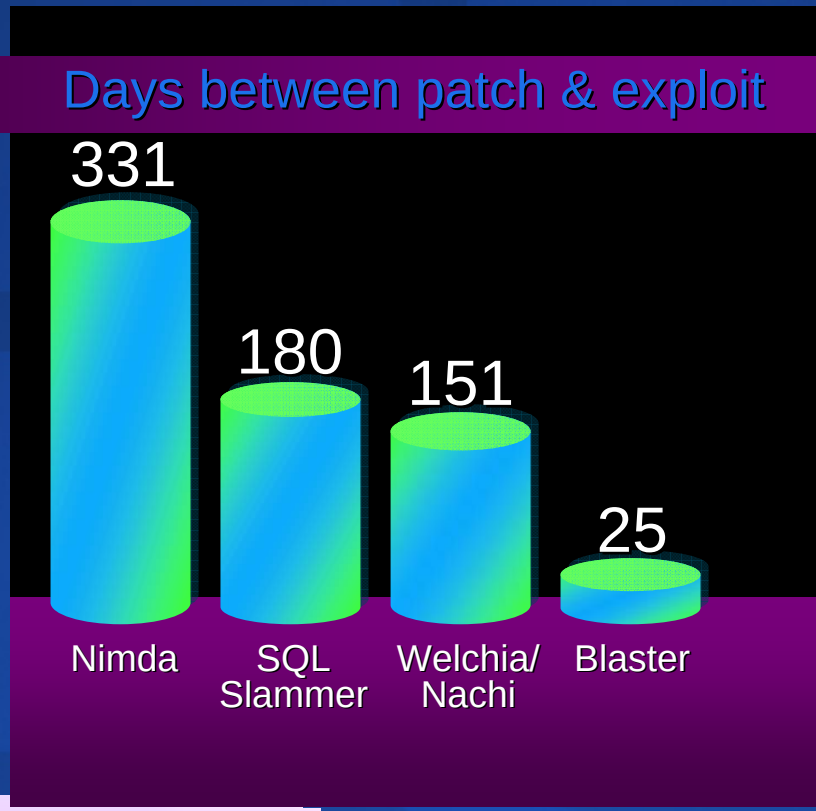
Source: Vendor Web Sites 1-Jan-03 to 31-Dec-03

Root Causes

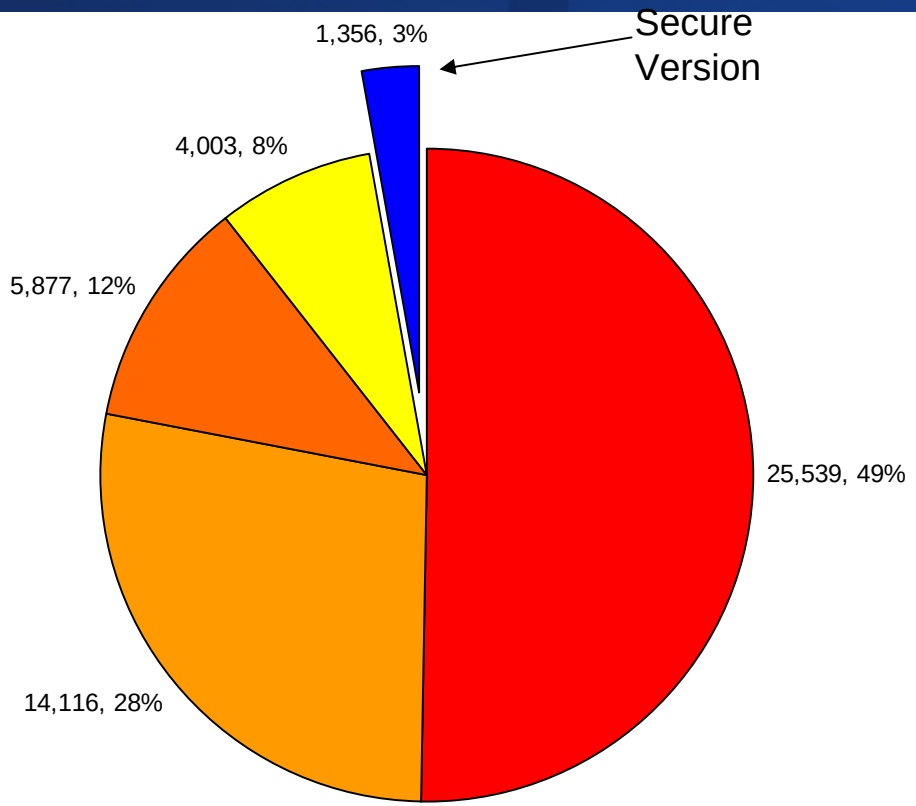
- Complexity
- Lack of Transparency
- Conflicting security models
- Large Monolithic Trusted Computing Bases
  - Vulnerability in on part of large system exposes the entire system
- Once attacked it’s hard to recover
- You’re now connected

The Challenge

- Patches proliferating
- Time to exploit decreasing
- Exploits more sophisticated
- Customer frustration



Time To Apply Patch - OpenSSL



49% have not applied a patch in 1 ½ years

0.9.6d and earlier	25,539	30-Jul-02
0.9.6e-h and 0.9.7	14,116	19-Feb-03
0.9.6i and 0.9.7a	5,877	17-Mar-03
0.9.6j and 0.9.7b	4,003	30-Sep-03
0.9.6k and 0.9.7c	1,356	

Source: “Vulnerable versions of OpenSSL apparently still widely deployed on commerce sites” netcraft.com 11/03

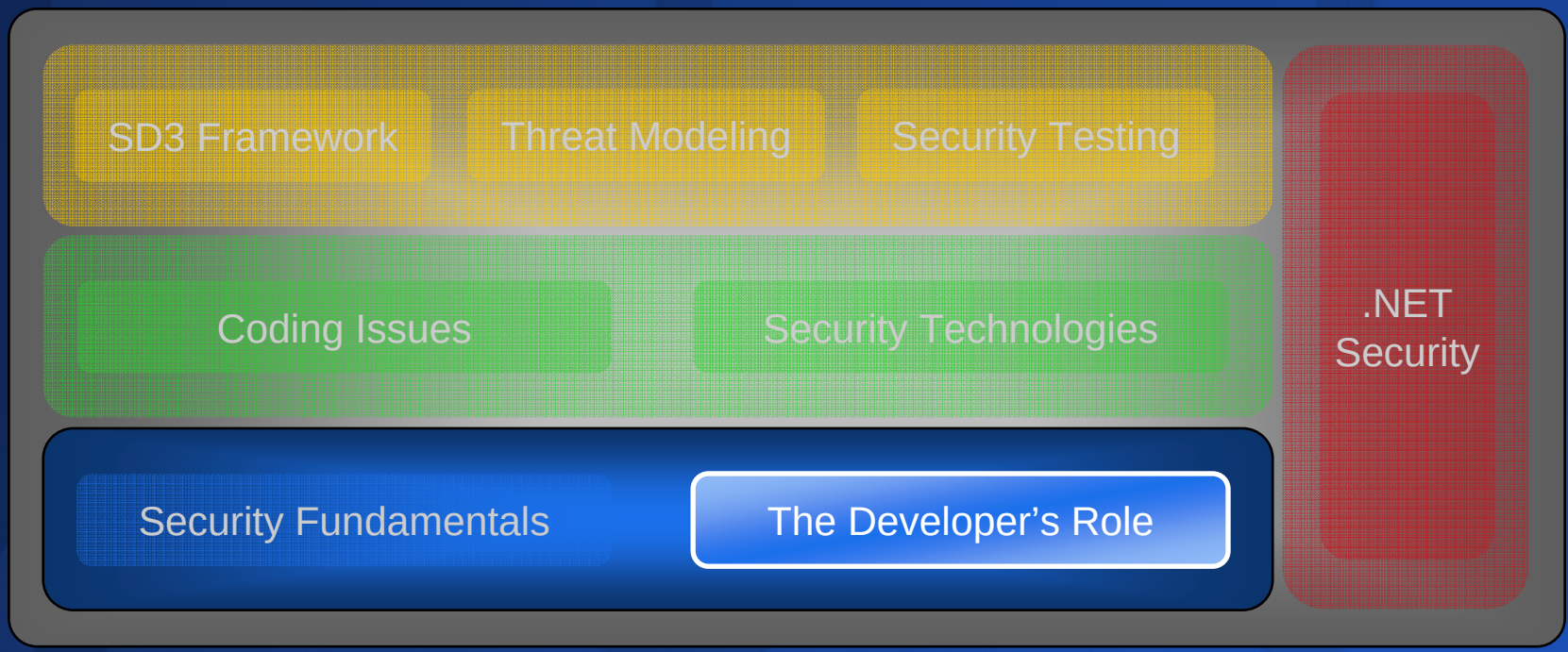
Why Is Security So Difficult?

- Attackers have extraordinary resources
- Attackers need to master only one attack
- Defenders constrained by ethics and laws
- Attackers have no rules and play dirty
- Defenders must serve business goals
- Defenders must win all the time
- Attackers have to find the weakest point only
- Attackers try any vulnerabilities
- Attackers attack at will

Security Today

- Technology alone will not solve your problem
- Nobody believes anything bad can happen to them, until it does
- Security works only if the secure way also happens to be the easiest way
- In you do not keep up with security fixes, your network will not be yours for long
- There really is someone out there trying compromise your systems
- Your data and systems are of value to someone
- Security is not about risk elimination; it is about risk management

The Developer's Role



demo

DOS Sample

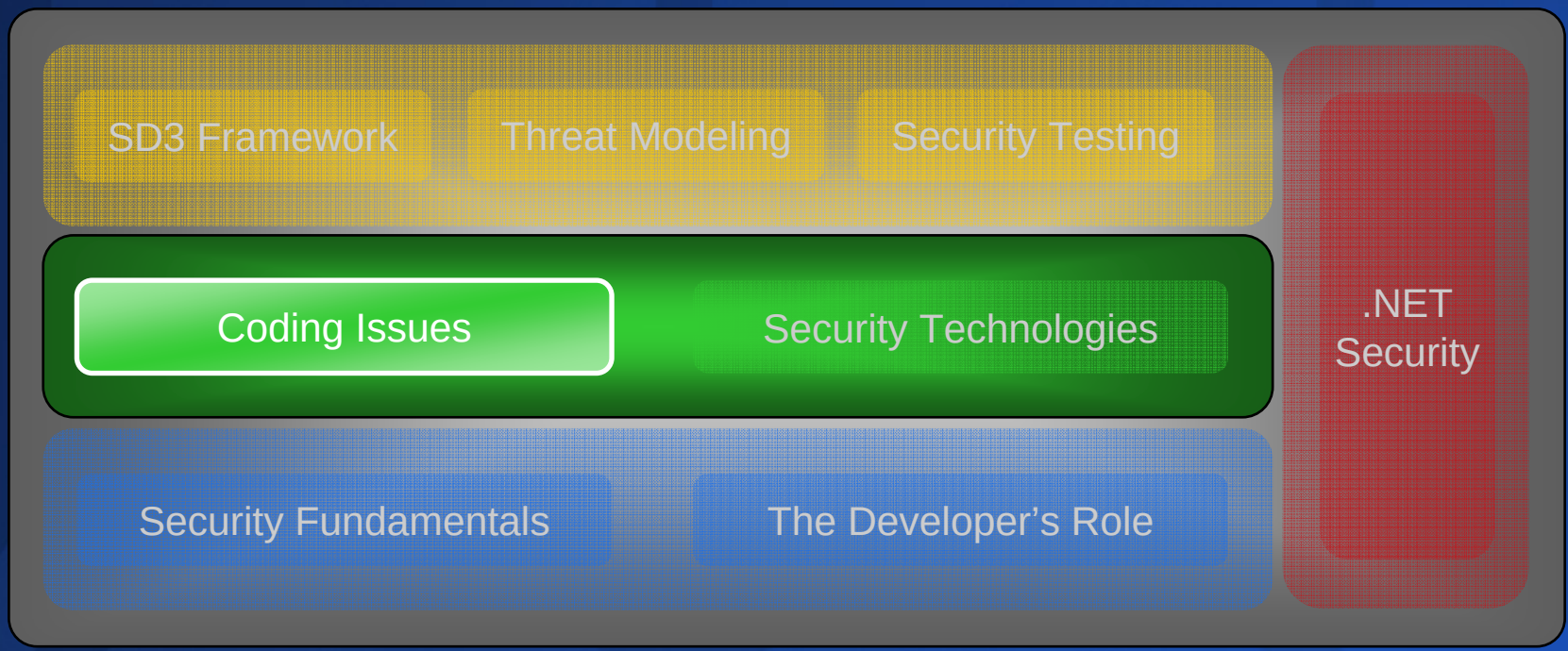
“Many Eyeballs Makes all Bugs Shallow”

- If you don’t know what to look for – you will find nothing
- People must want to review old code
  - Not just create sexy new features
- It misses the point
  - We should not be putting the security bugs in the code in the first place!
  - Too much emphasis on code
  - What about design?

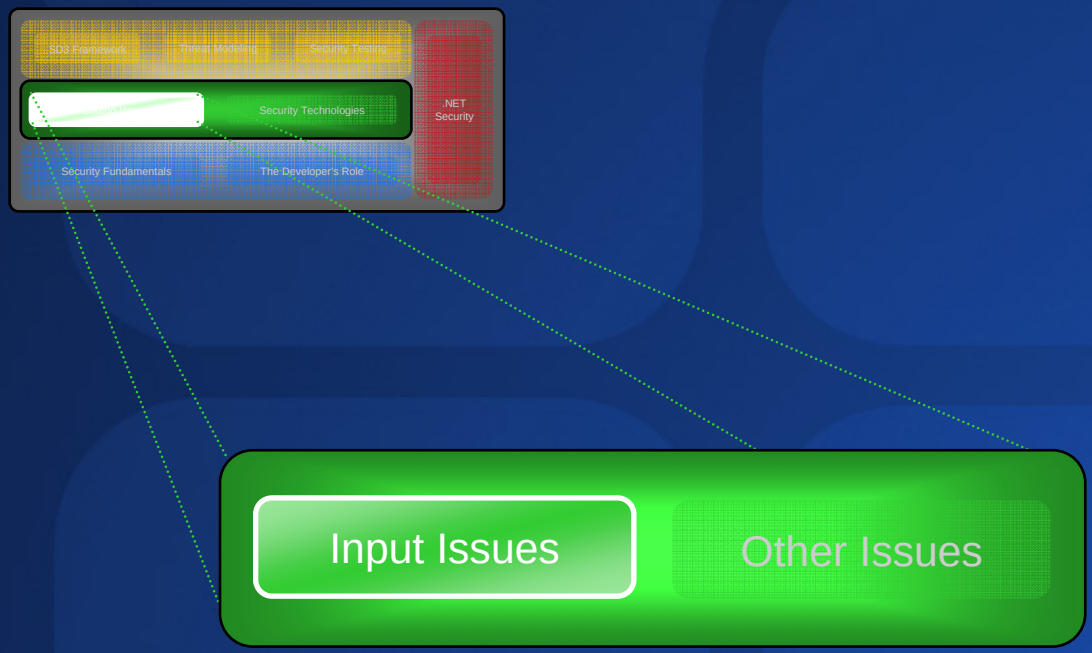
demo

EBay

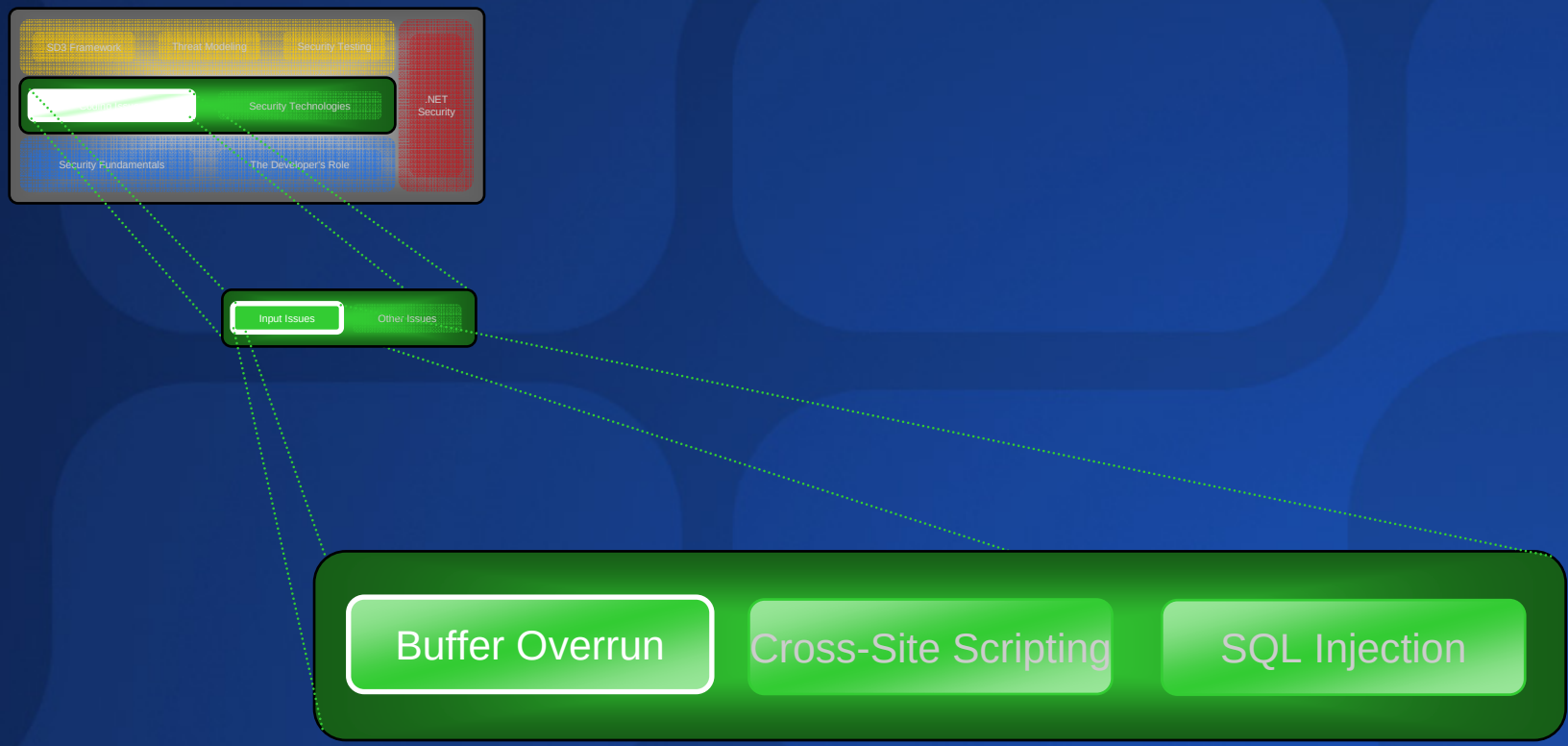
Coding Issues



Coding Issues



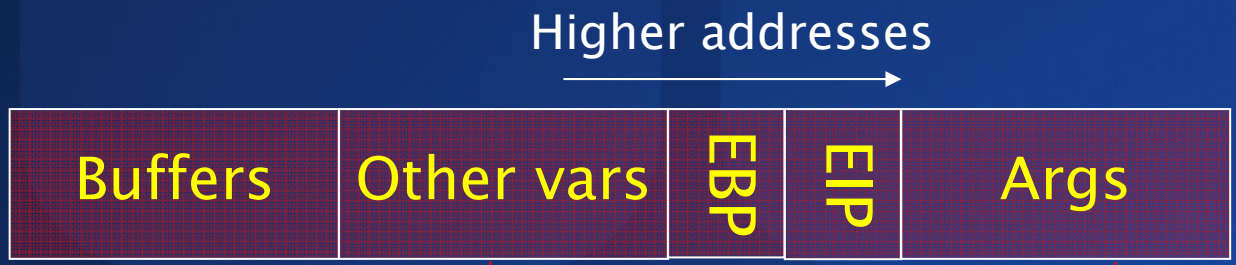
Buffer Overrun



demo

Buffer Overrun

Stack BOs at Work

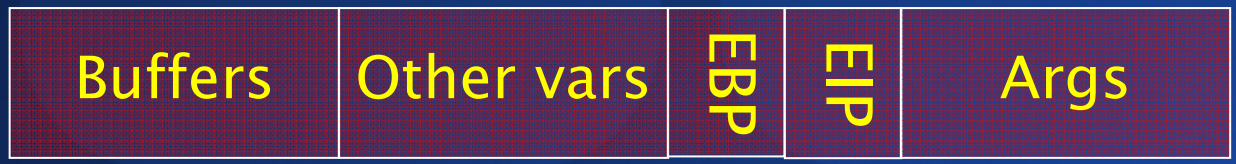



```
void foo(char *p, int i) {
  int j = 0;
  CFoo foo;
  int (*fp)(int) = &func;
  char b[16];
}
```

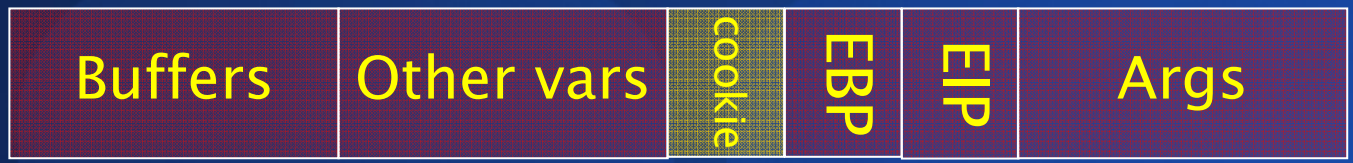
VC++ /GS at work

A normal stack

Higher addresses



VC++ 2002 stack (with /GS)



VC++ 2003 stack (with /GS and safe exceptions)

- EH #1: 0x77E74717
- EH #2: 0x77E8C2E7
- EH #3: 0x77E95FE7

In PE Header

View Safe Exception list with:
link -dump -loadconfig <executable>

Integer Overflow Attacks

- Integer overflow is a generic name for a set of common integer arithmetic mistakes that can lead to BOs
  - Overflow and underflow
  - Signed versus unsigned errors
  - Truncation
- They lead to BOs

Integer Overflow Attacks

```
int ConcatString(char *buf1, char *buf2,
                 size_t len1, size_t len2){
    char buf[255];

    if((len1 + len2) > 0xFF) return -1;

    memcpy(buf, buf1, len1);
    memcpy(buf + len1, buf2, len2);

    // do stuff with buf

    return 0;
}
```

```
      0x103  len1
+ 0xFFFFFFFC  len2
-------------
        0xFF
```

Both **memcpy** functions attempt to copy >255 bytes

Integer Arithmetic in C#
An Example ‘Authz’ Method

```
// 0-19 system data, 20-499 user data
const UInt32 USER_DATA_START = 20;
Object[] myData = new Object[500];

[PrincipalPermissionAttribute(SecurityAction.Demand,
   Role=@"BUILTIN\Administrators")]
public Object
GetSystemData(uint ItemNumber, WindowsPrincipal pr) {
    return myData[ItemNumber];
}

public Object GetUserData(UInt32 ItemNumber) {
    return myData[ItemNumber + USER_DATA_START ];
}
```

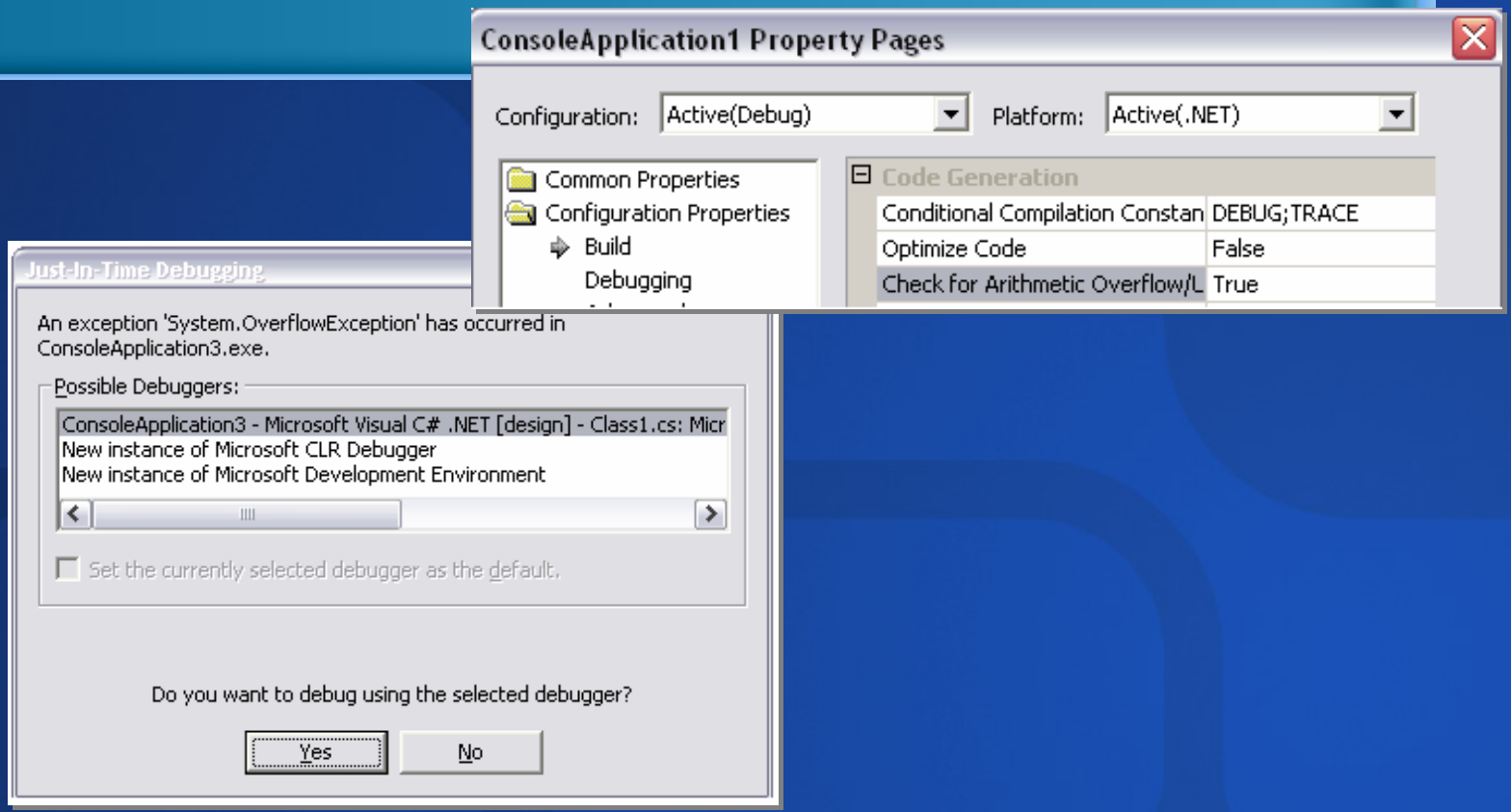
What if ItemNumber is close to Uint32.MaxValue?

Remedy: Integer Arithmetic

- Any calculation used to determine an array offset or memory allocation is suspect
- Use unsigned variables for array indexes and buffer sizes
- Watch out for:
  - C4018 & C4389 (signed/unsigned mismatch)
  - C4244 warnings (conversion from 'type1' to 'type2', possible loss of data)
  - #pragma and casts that shut the compiler up!

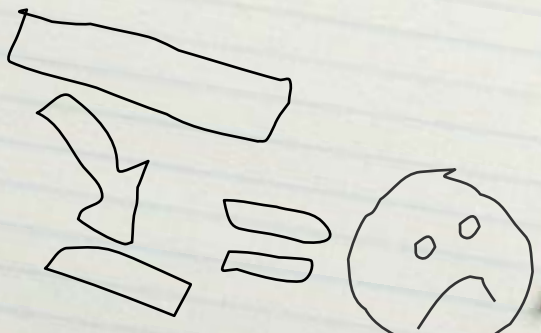
Remedy: Arithmetic Errors in C#

```
public Object GetUserData(UInt32 ItemNumber) {
    checked {
        return myData[ItemNumber+USER_DATA_START];
    }
}
```

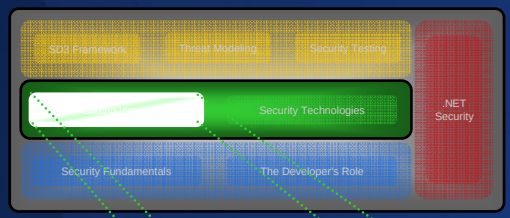


Buffer Overrun Checklist

- [x] Just fix 'em!
- [x] Build defensive code, and add defensive layers (/GS etc)
- [x] Use the latest version of VC++ /GS
- [x] User safer string libraries
- [x] All arithmetic used to calculate memory allocations are probably wrong!



Cross-Site Scripting



Buffer Overrun

Cross-Site Scripting

SQL Injection

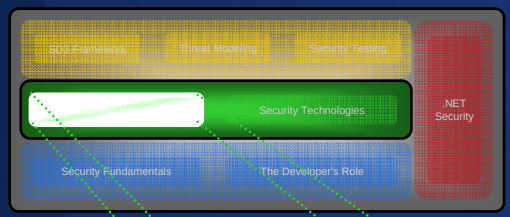
demo

Cross-Site Scripting

Anatomy of Cross-Site Scripting

- Web based applications
  - Redirect info via <*form*>
  - E-Mail platforms & discussion boards
- Allows hackers to:
  - Execute script in client's browser
  - <*script*>, <*object*>, <*applet*>, <*form*>, <*embed*>
- Arising threats
  - Steal session / AuthN cookies
  - Access to client computer

SQL Injection



demo

SQL Injection

Anatomy of SQL Injections (2/2)

- Problem: string concatenation

```
strSql = "SELECT * FROM titles " & _
         "WHERE id LIKE '" & textName.Text & "'"
Dim cmd As New SqlCommand(strSql, "server=...")
myReader = cmd.ExecuteReader()
```

Good Guy

ID: 1001

Not so Good Guy

Really Bad Guy

Downright Evil Guy

ID: 1001'; exec xp_cmdshell('fdisk.exe') --

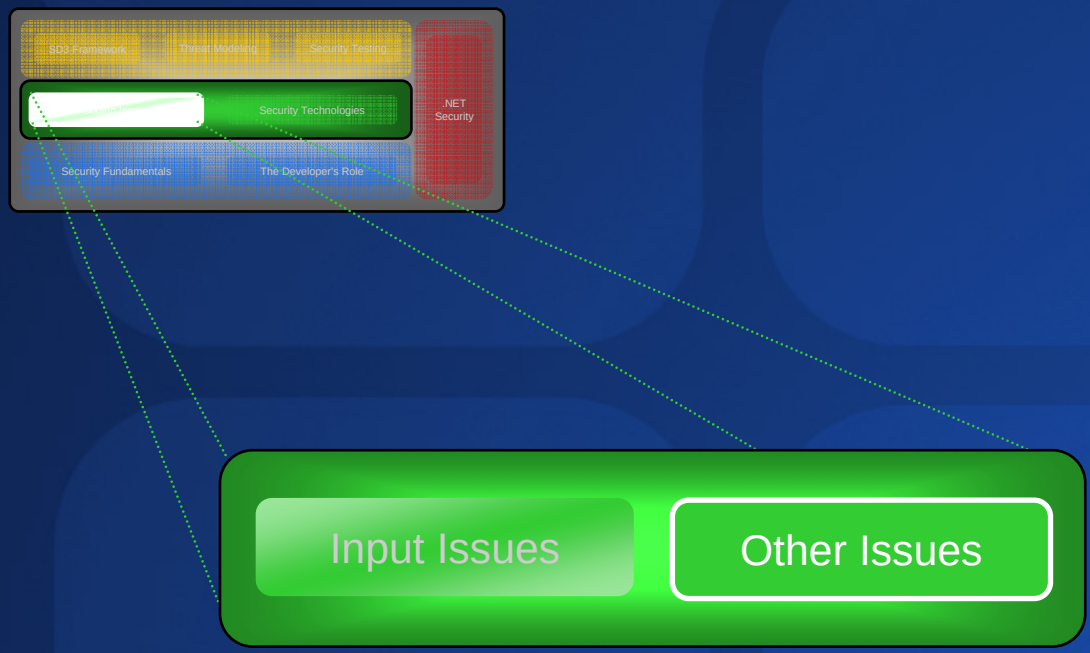
```
SELECT *
FROM titles
WHERE id='1001'; exec xp_cmdshell('fdisk.exe') --
```

```
ders -- '
```

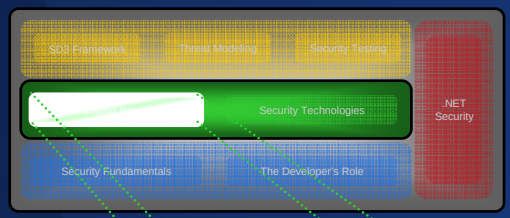
demo

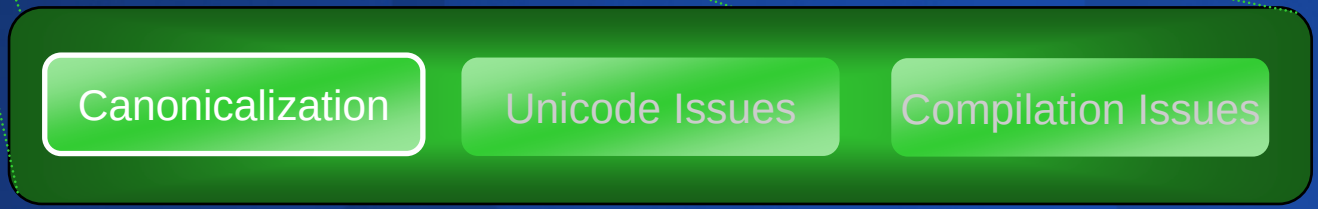
Preventing SQL Injection

Other Issues



Canonicalization Issues





Canonicalization Issues

- Never make a security decision based on the name of something
  - Chances are good that you'll get it wrong
  - Often, there is more than one way to name something

```
if (username == @"DOMAIN\User" &&
      filename == "MyLongFile.txt")
      // go away
```

File Name Issues

1. MyLongFile.txt
2. MyLongFile.txt.
3. MyLong~1.txt
4. MyLongFile.txt::$DATA

Character Representation

http://www.microsoft.com/technet/security

Is the same as -

http://www%2emicrosoft%2ecom%2ftechnet%2fsecurity

http://www.microsoft.com%c0%aftechnet%c0%afsecurity

http://www%25%32%65microsoft.com/technet/security

http://172.43.122.12 = http://2888530444

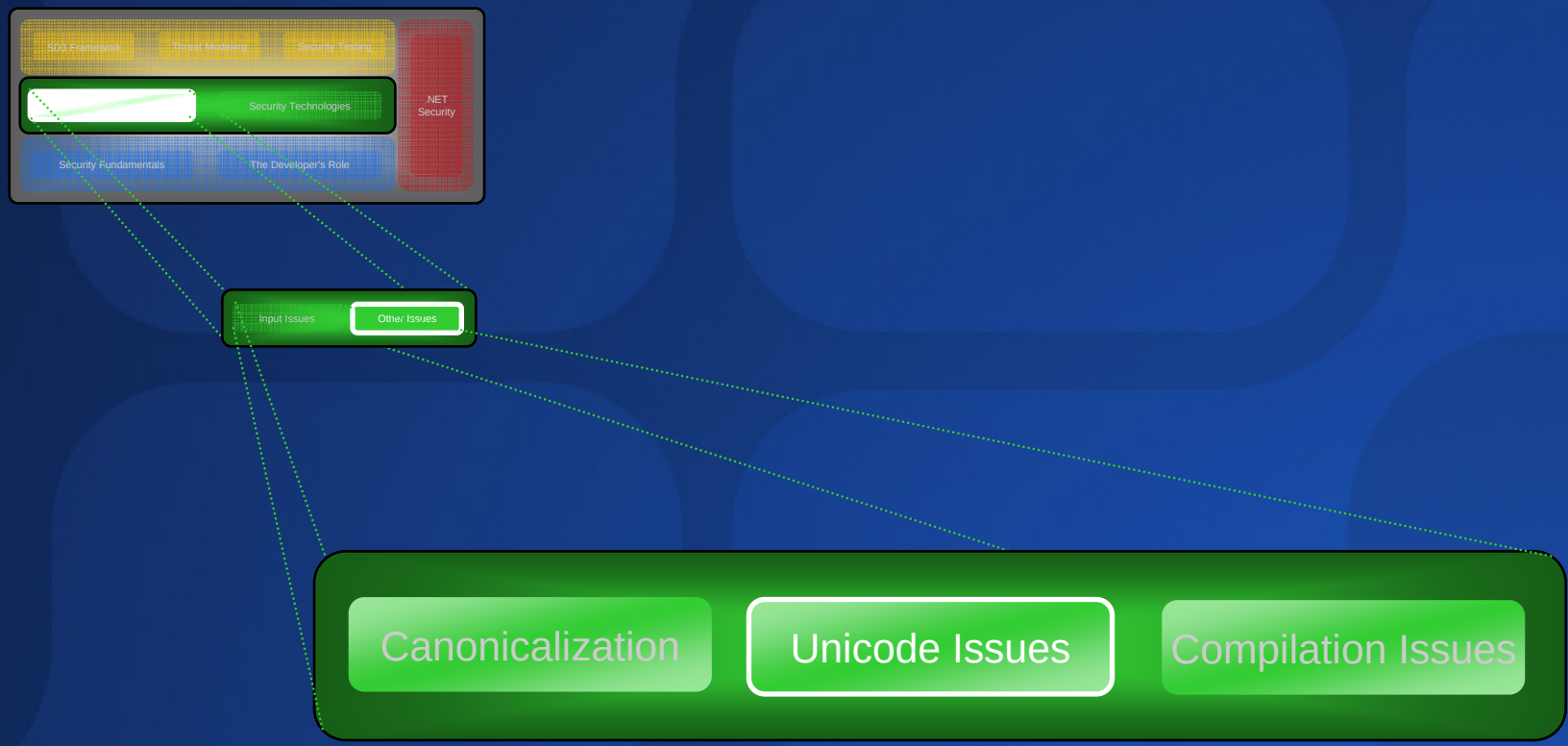
Canonicalization Issues

http://www.foo.com/prn/default.asp

http://www.foo.com/aux.asp

http://www.cnn.com&story=breaking_news@18.69.0.44/evarady/www/top_story.htm

Unicode Issues



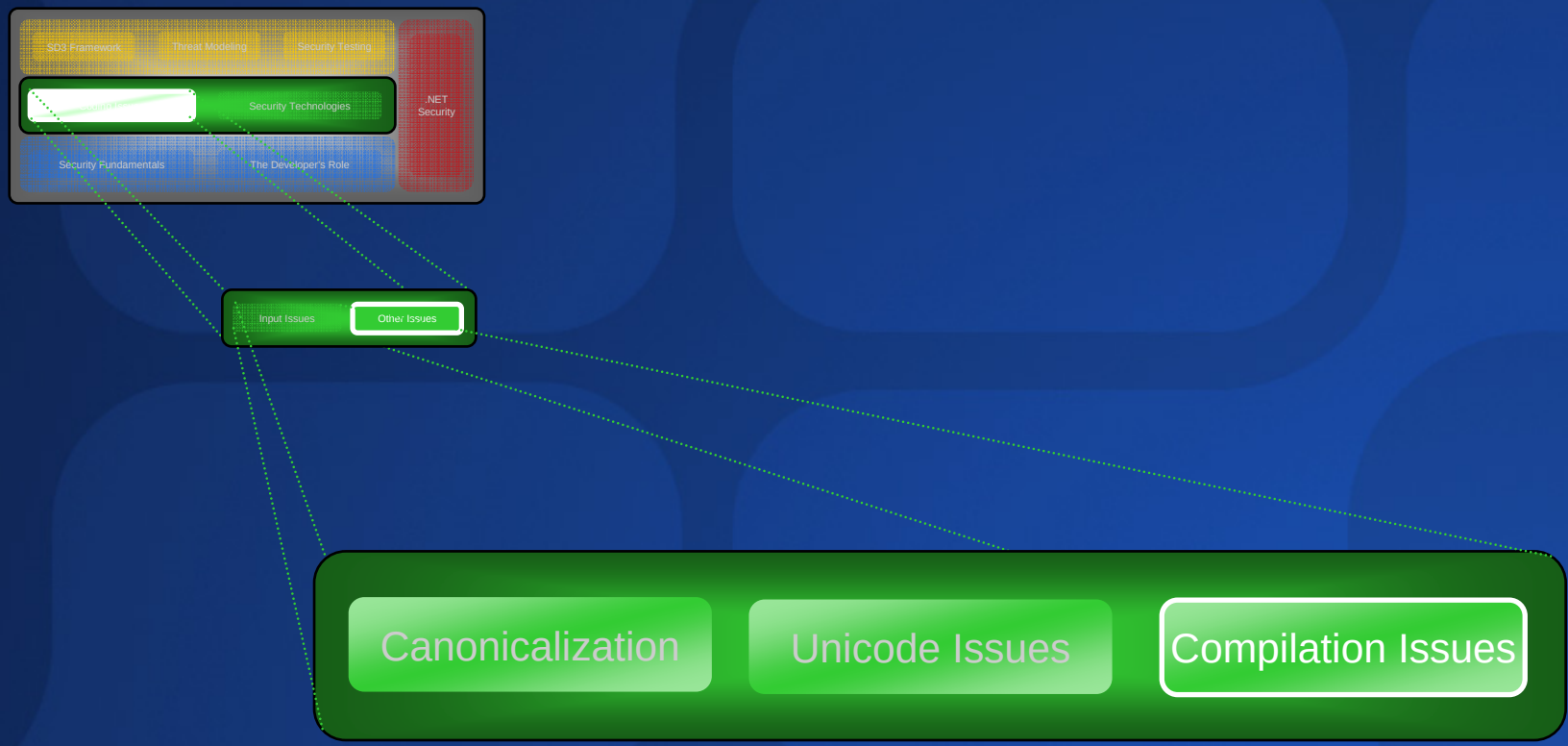
The Turkish-I problem (Applies also to Azerbaijan!)

- Turkish has four letter ‘I’s
  - i (U+0069) ı (U+0131) İ (U+0130) I (U+0049)
- In Turkish locale `UC("file")==FİLE`

```
// Do not allow "FILE://" URLs
if(url.ToUpper().Left(5) == "FILE:")
  return ERROR;
getStuff(url);
```

```
// Only allow "HTTP://" URLs
if(url.ToUpper(CULTURE_INVARIANT).Left(5) == "HTTP:")
  getStuff(url);
else
  return ERROR;
```

Canonicalization Issues



Scrubbing Secrets in Memory

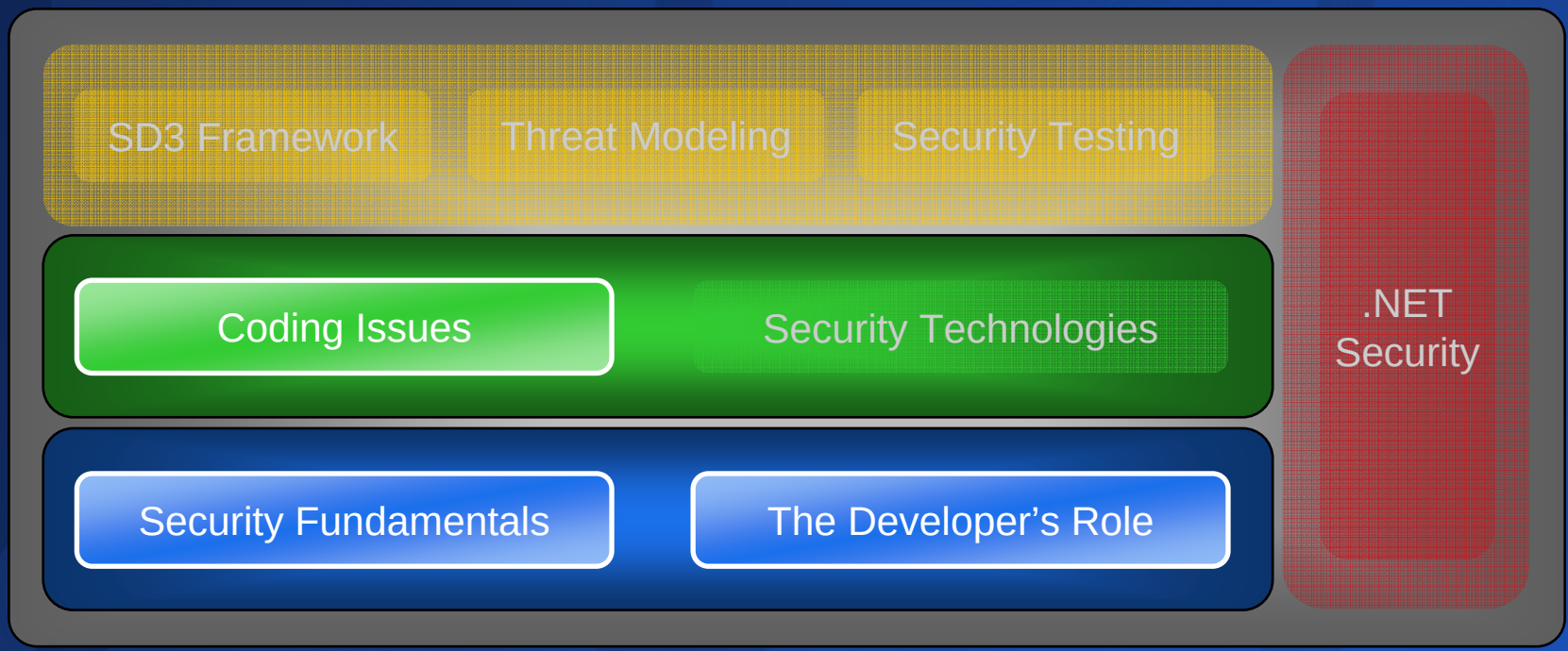
What's wrong with this code?

```
void Function() {
        char pwd[32];
        GetPwdFromUser(pwd,32);
        UsePwd(pwd,32);
        memset(pwd,0,32);
}
```

Victim of
"dead store removal"
by optimizing compilers

```
void Function() {
        char pwd[32];
        GetPwdFromUser(pwd,32);
        UsePwd(pwd,32);
        SecureZeroMemory(pwd,32);
}
```

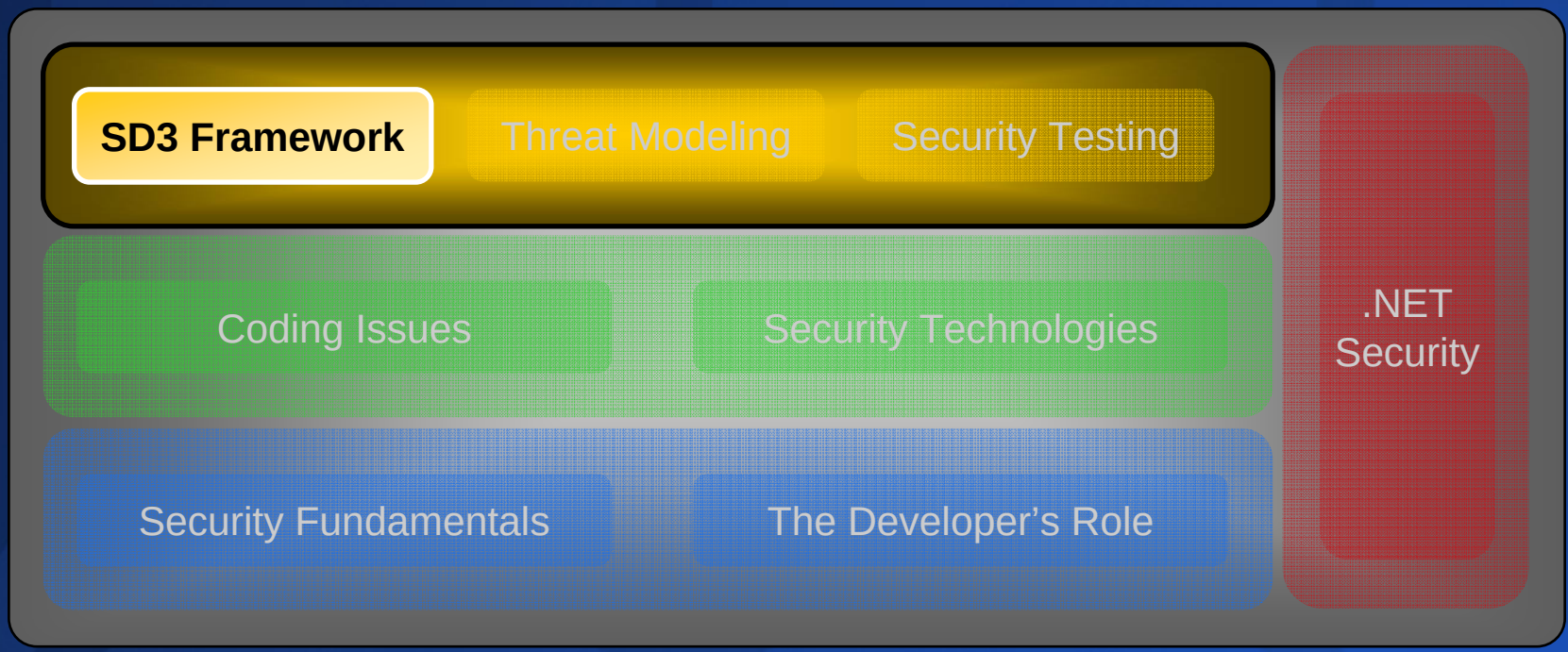
Security for Developer



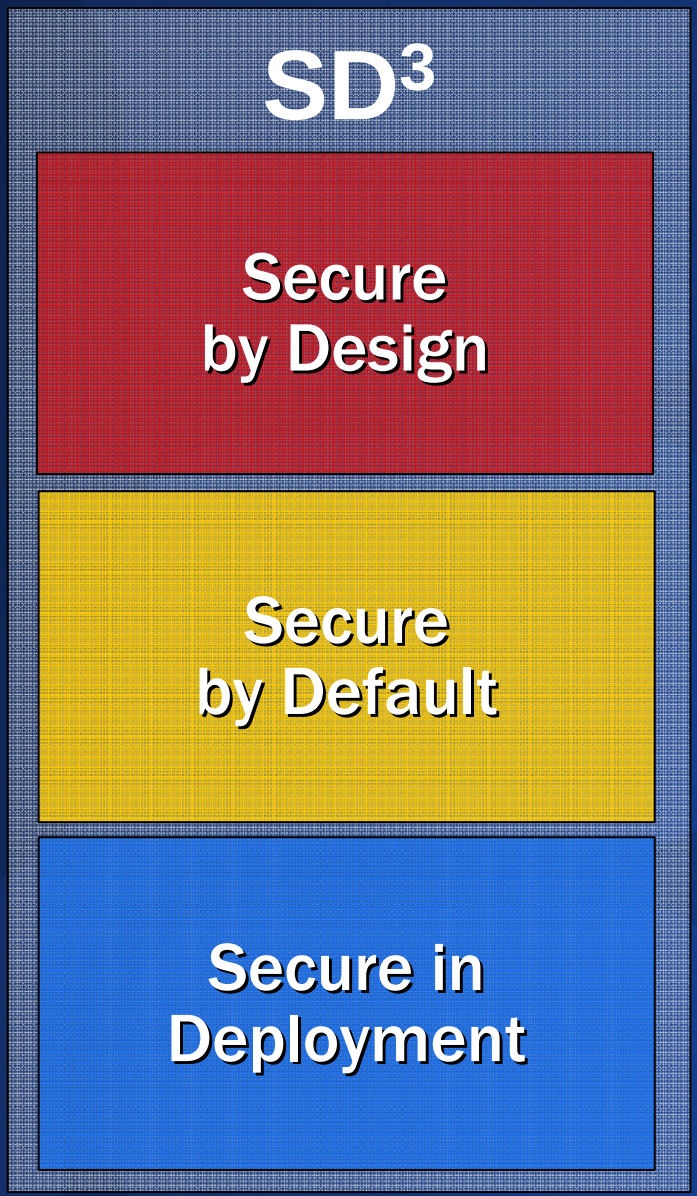
Note…

Security is only as good
as its weakest link

Security for Developer



The SD³ Security Framework



- Threat analysis
- Secure architecture & code
- Vulnerability reduction

- Reduced attack surface
- Unused features → off by default
- Least privilege

- Detection, defense, recovery, management
- Architecture guides, how-to
- Ongoing education

Defense in Depth (MS03-007)

Windows Server 2003 Unaffected

The underlying DLL (NTDLL.DLL) not vulnerable	Code made more conservative during Security Push
Even if it was vulnerable	IIS 6.0 not running by default on Windows Server 2003
Even if it was running	IIS 6.0 doesn't have WebDAV enabled by default
Even if it did have WebDAV enabled	Maximum URL length in IIS 6.0 is 16kb by default (>64kb needed)
Even if the buffer was large enough	Process halts rather than executes malicious code, due to buffer-overrun detection code (-GS)
Even if it there was an exploitable buffer overrun	Would have occurred in *w3wp.exe* which is now running as 'network service'

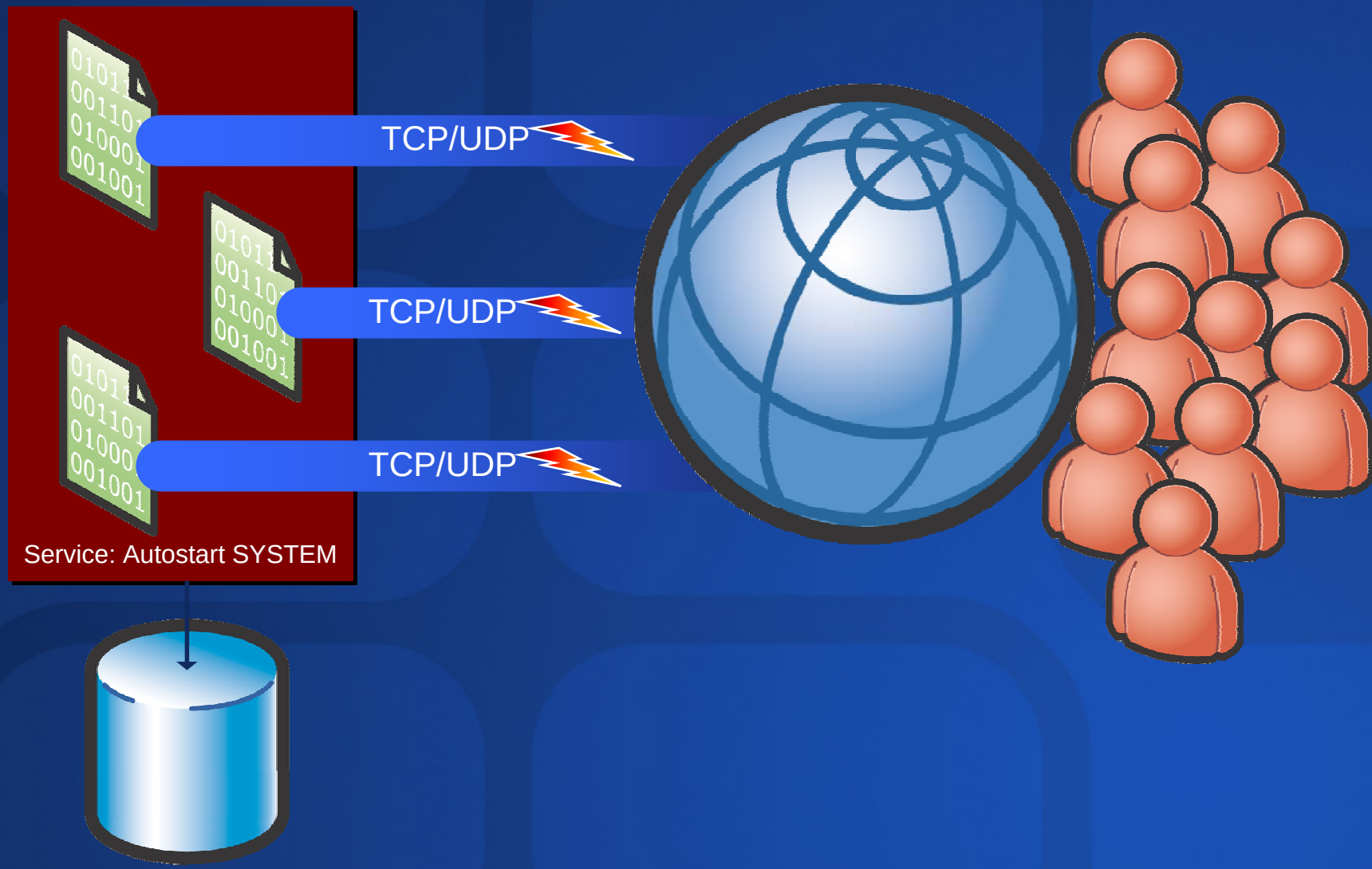
Secure Defaults

- Less code running by default = less stuff to attack by default
- Slammer & CodeRed would not have happened if the features were not enabled by default
- Reduces the urgency to deploy security fixes
  - A ‘critical’ may be rated ‘important’
- Defense in depth removes single points of failure
- Reduces the need for customers to ‘harden’ the product
- Reduces your testing workload
- Reduce your attack surface early!

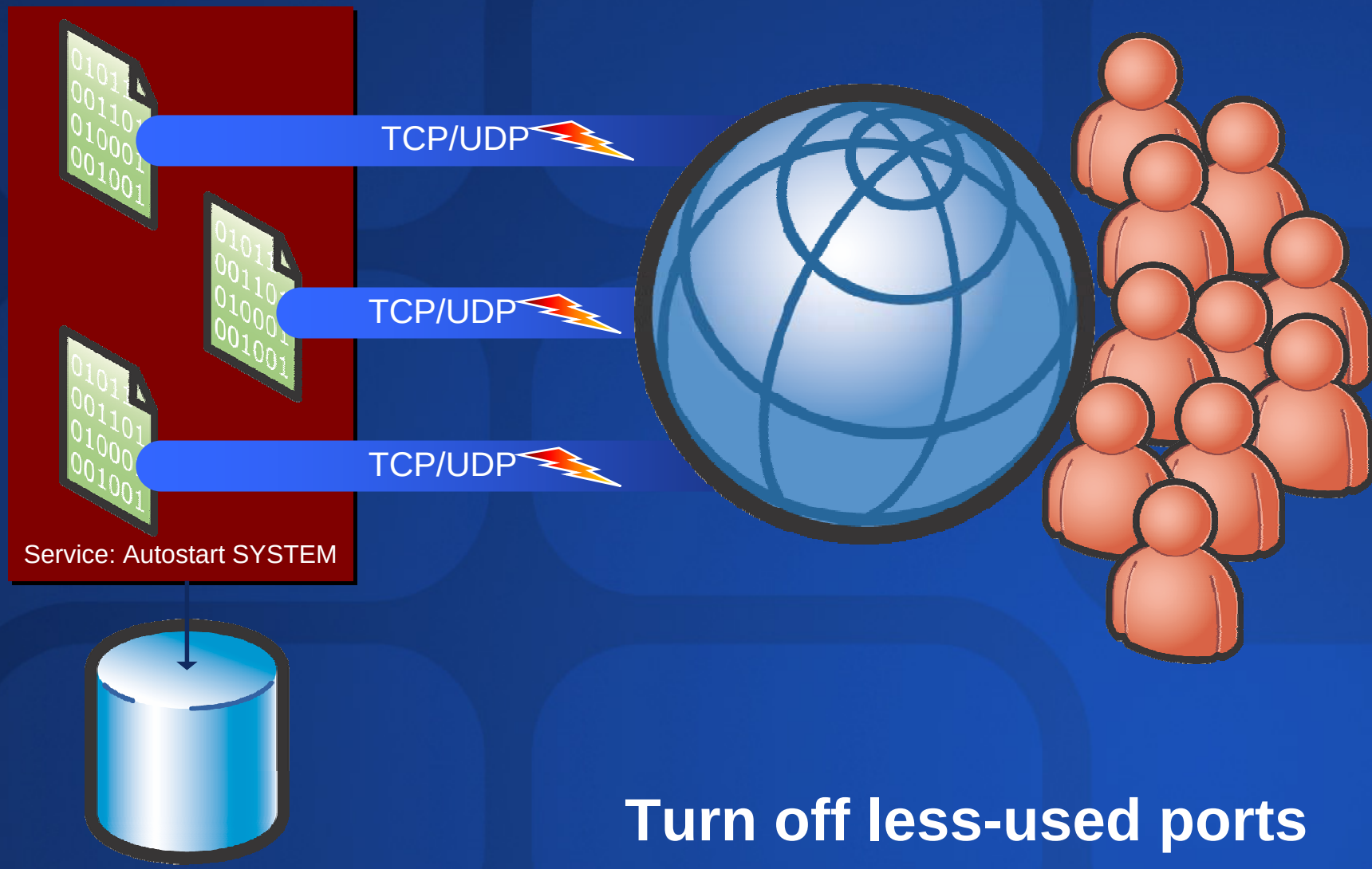
Increased Attack Surface means Increased Security Scrutiny…

… From the good guys, and the bad guys!

Attack Surface Reduction Ideas

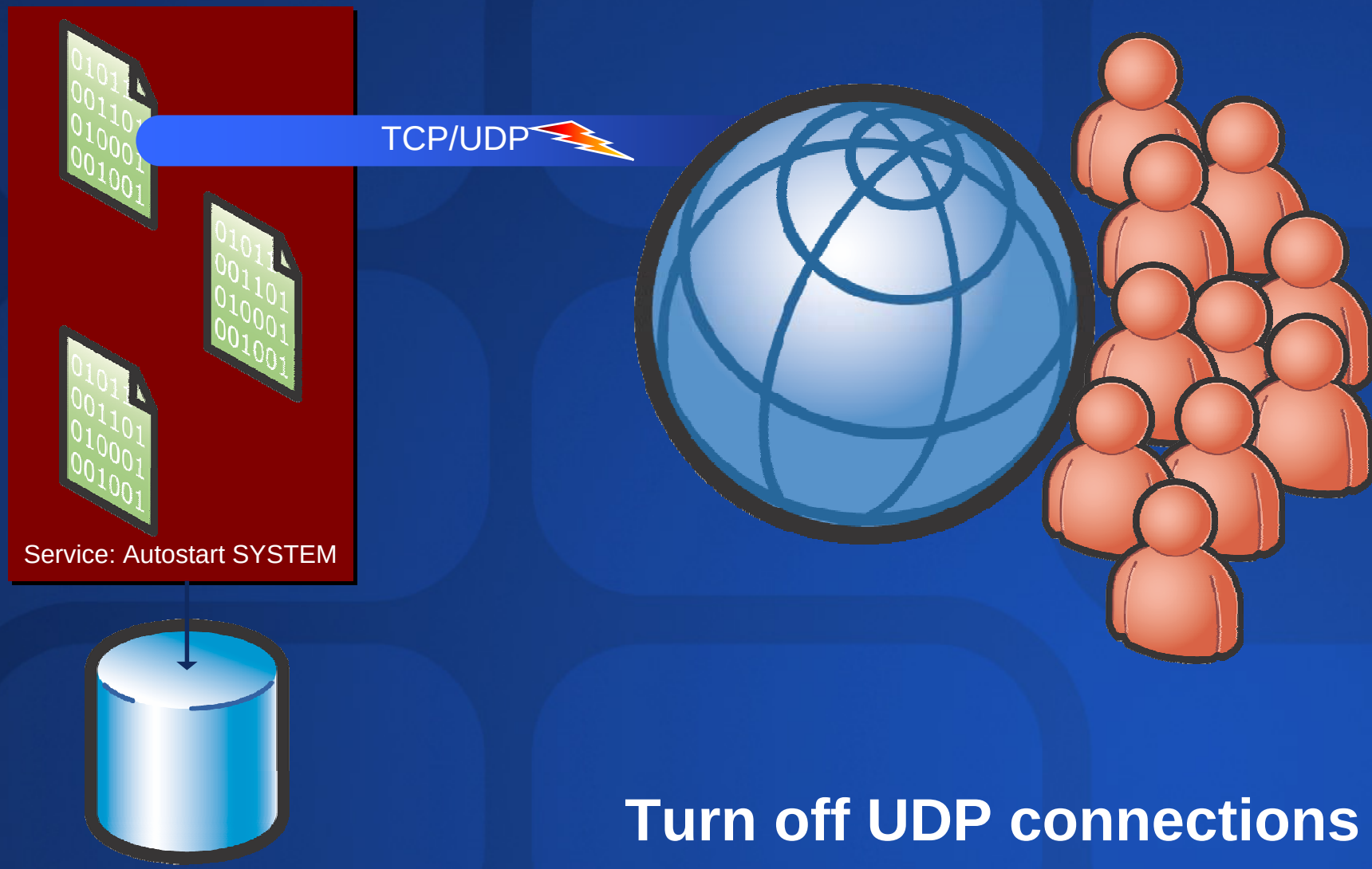


Attack Surface Reduction Ideas



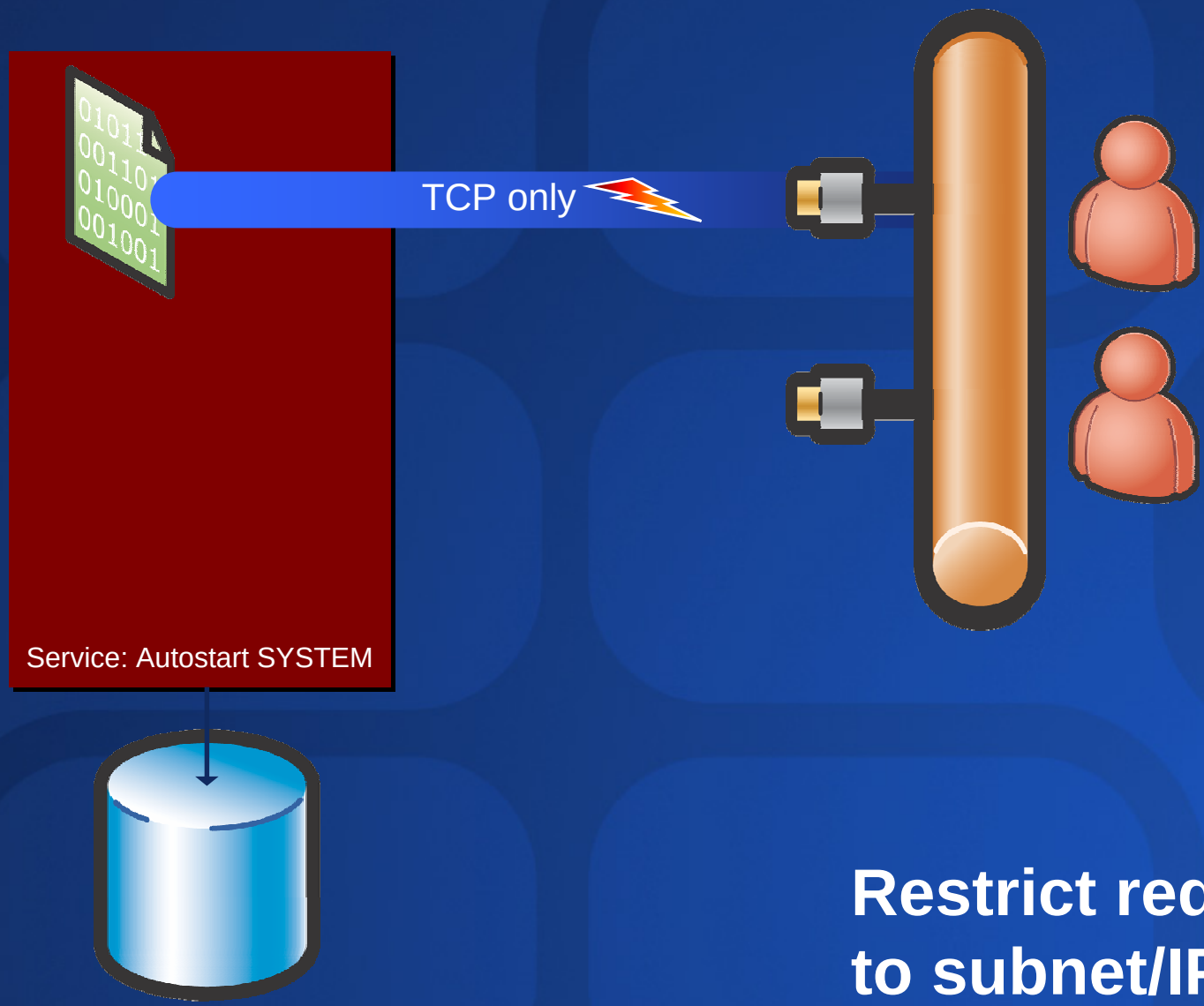
Turn off less-used ports

Attack Surface Reduction Ideas



Turn off UDP connections

Attack Surface Reduction Ideas

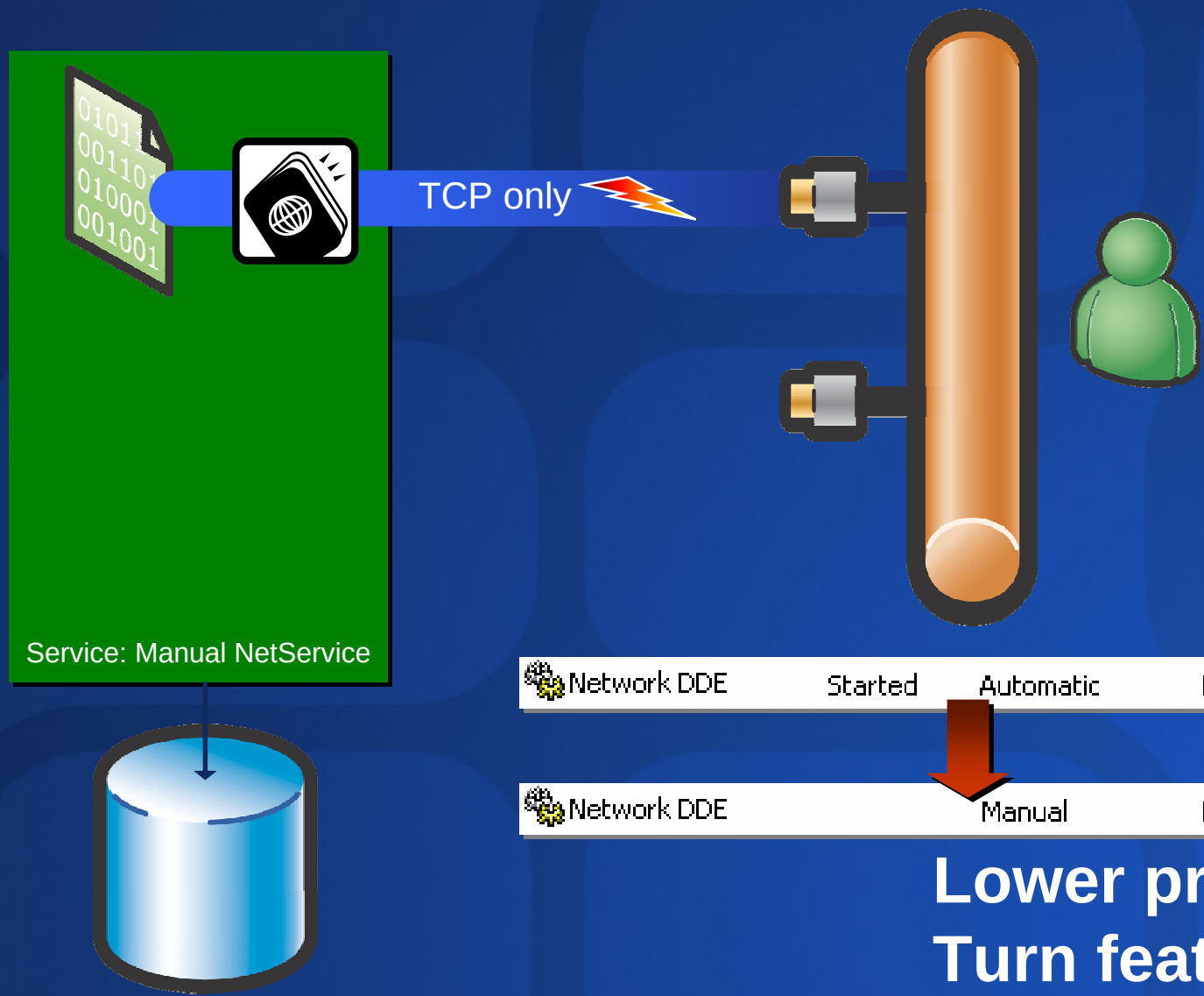


Restrict requests to subnet/IP range

Attack Surface Reduction Ideas

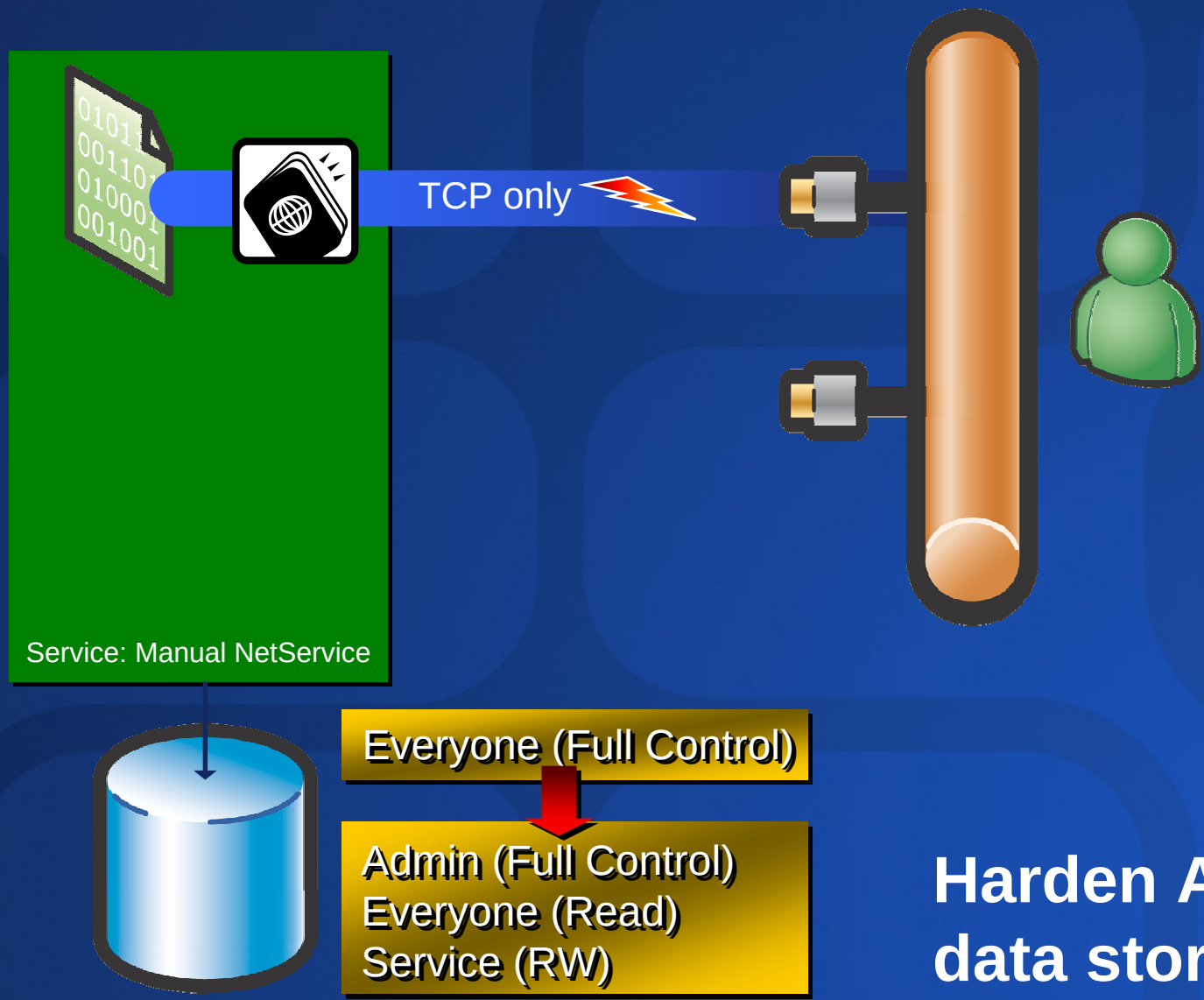
Authenticate connections

Attack Surface Reduction Ideas

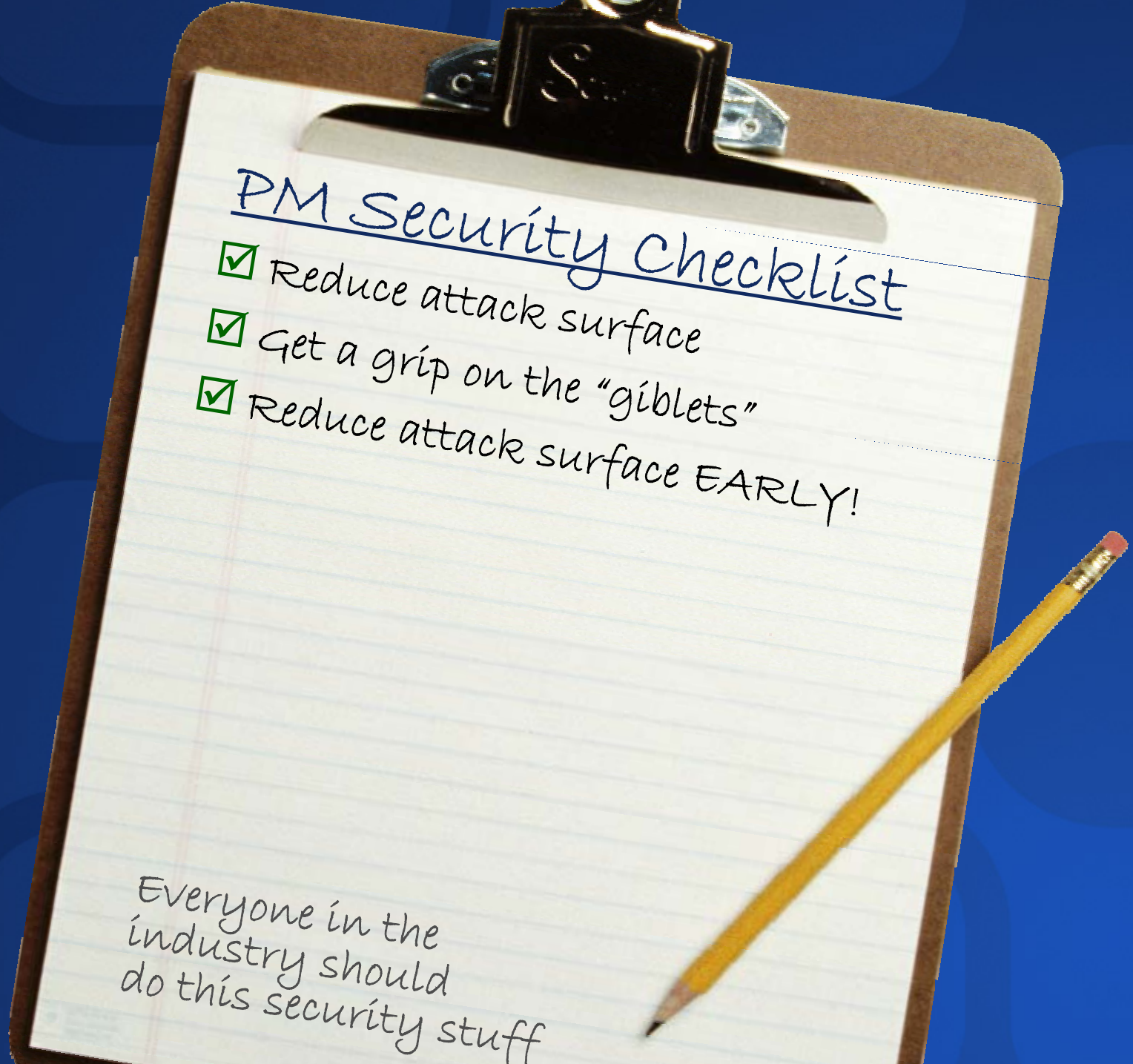


Lower privilege
Turn feature off

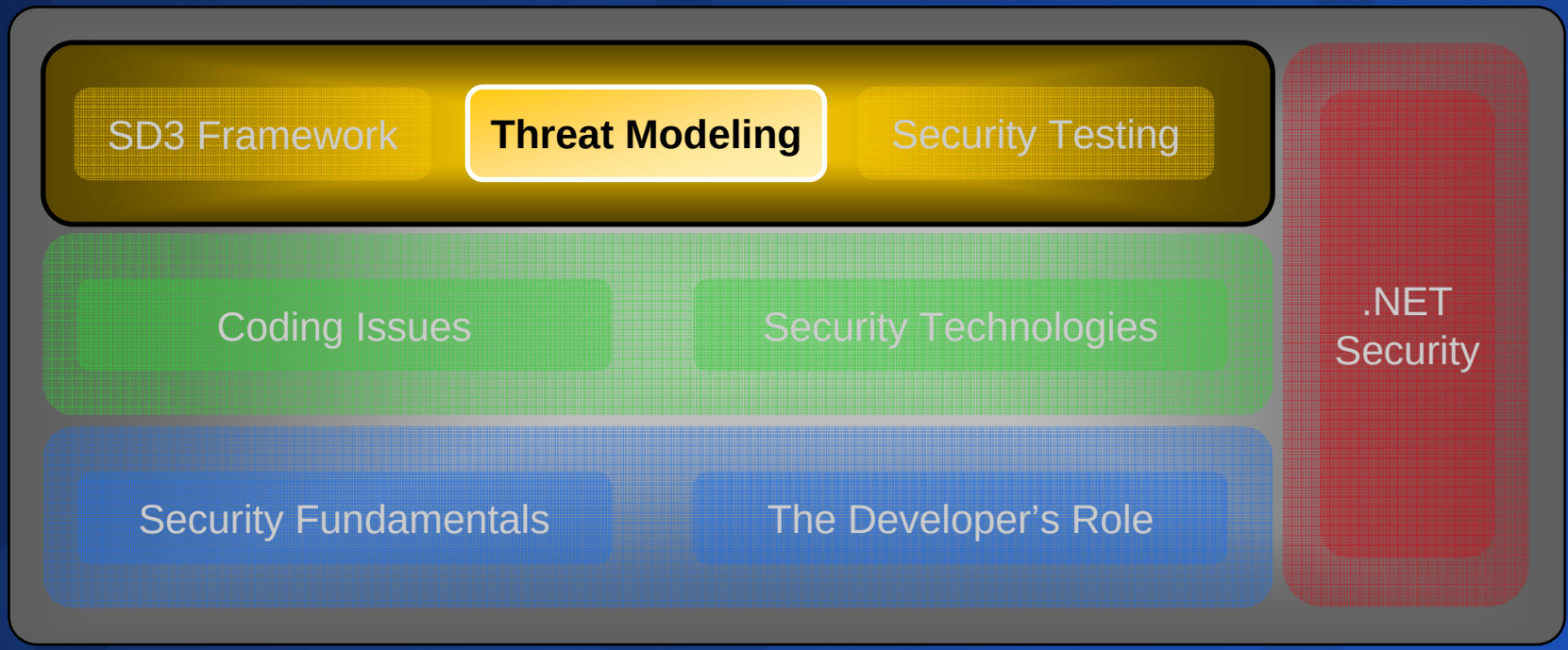
Attack Surface Reduction Ideas



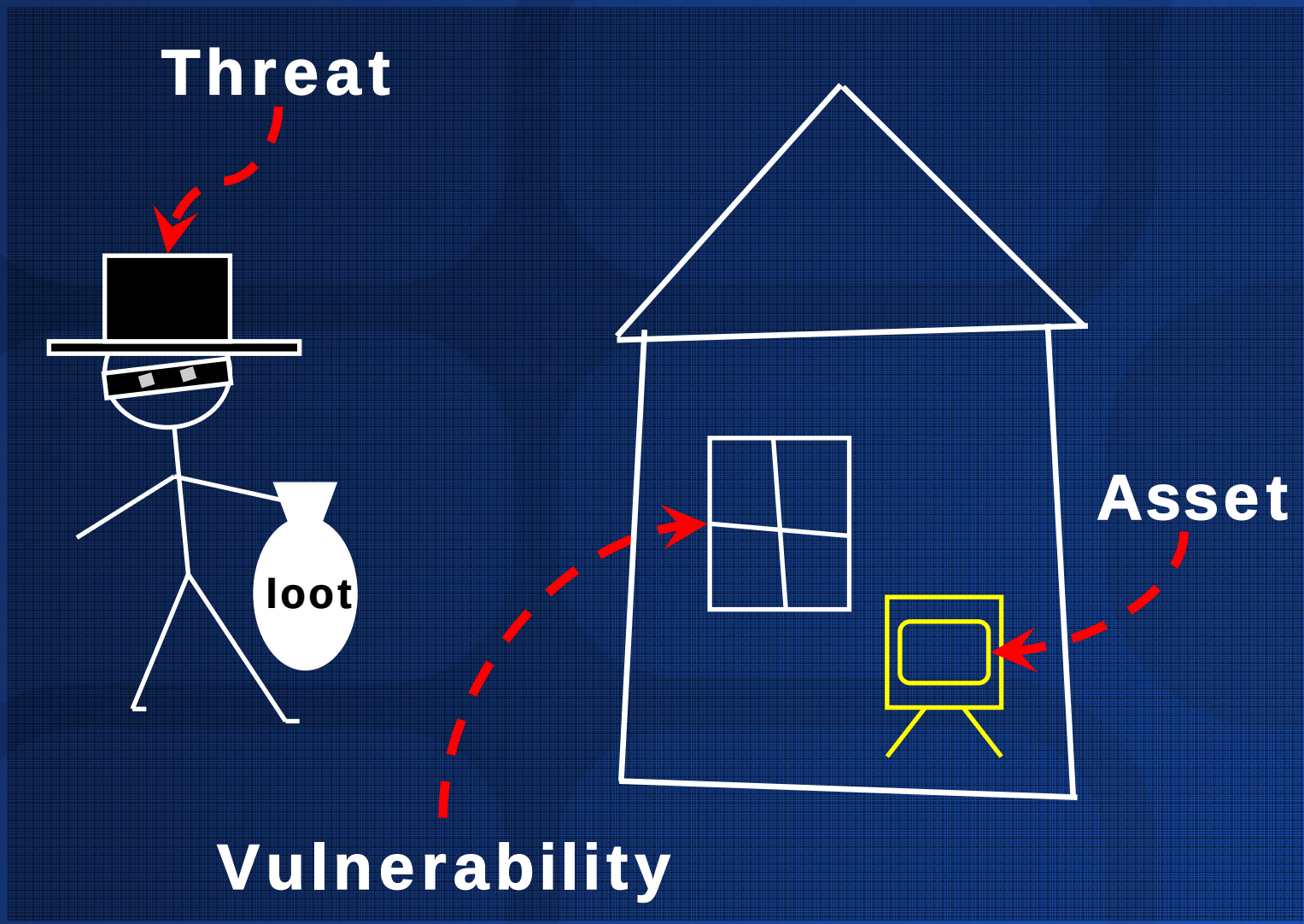
Harden ACLs on data store


PM Security Checklist
Reduce attack surface
Get a grip on the "giblets"
Reduce attack surface EARLY!
Everyone in the industry should do this security stuff

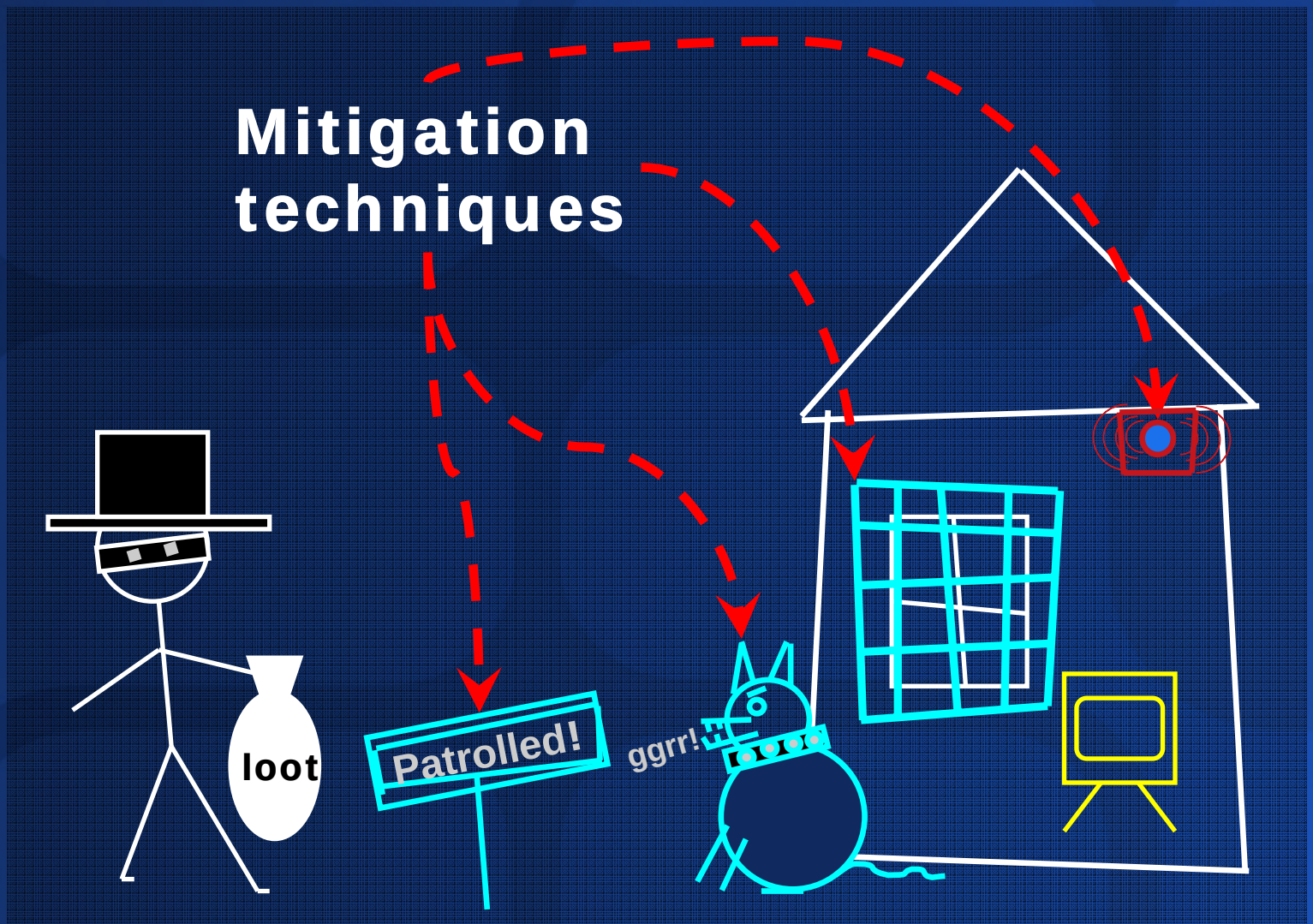
Security for Developer



Threats and Defense



Threats and Defense

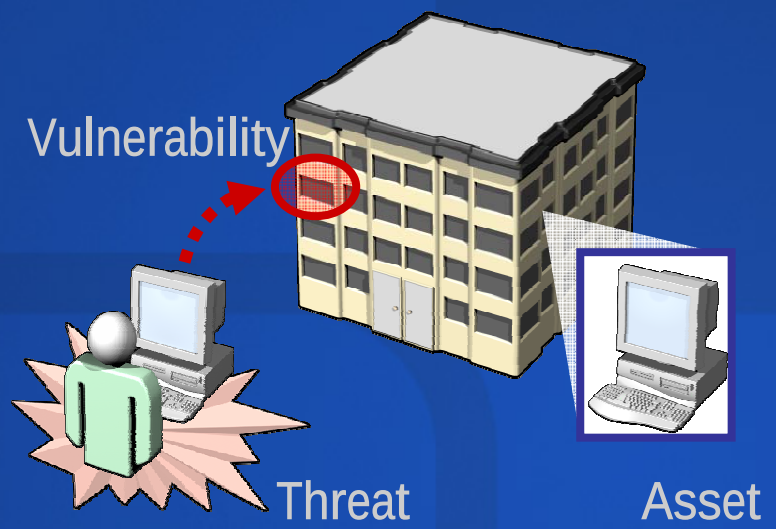


Threat Analysis

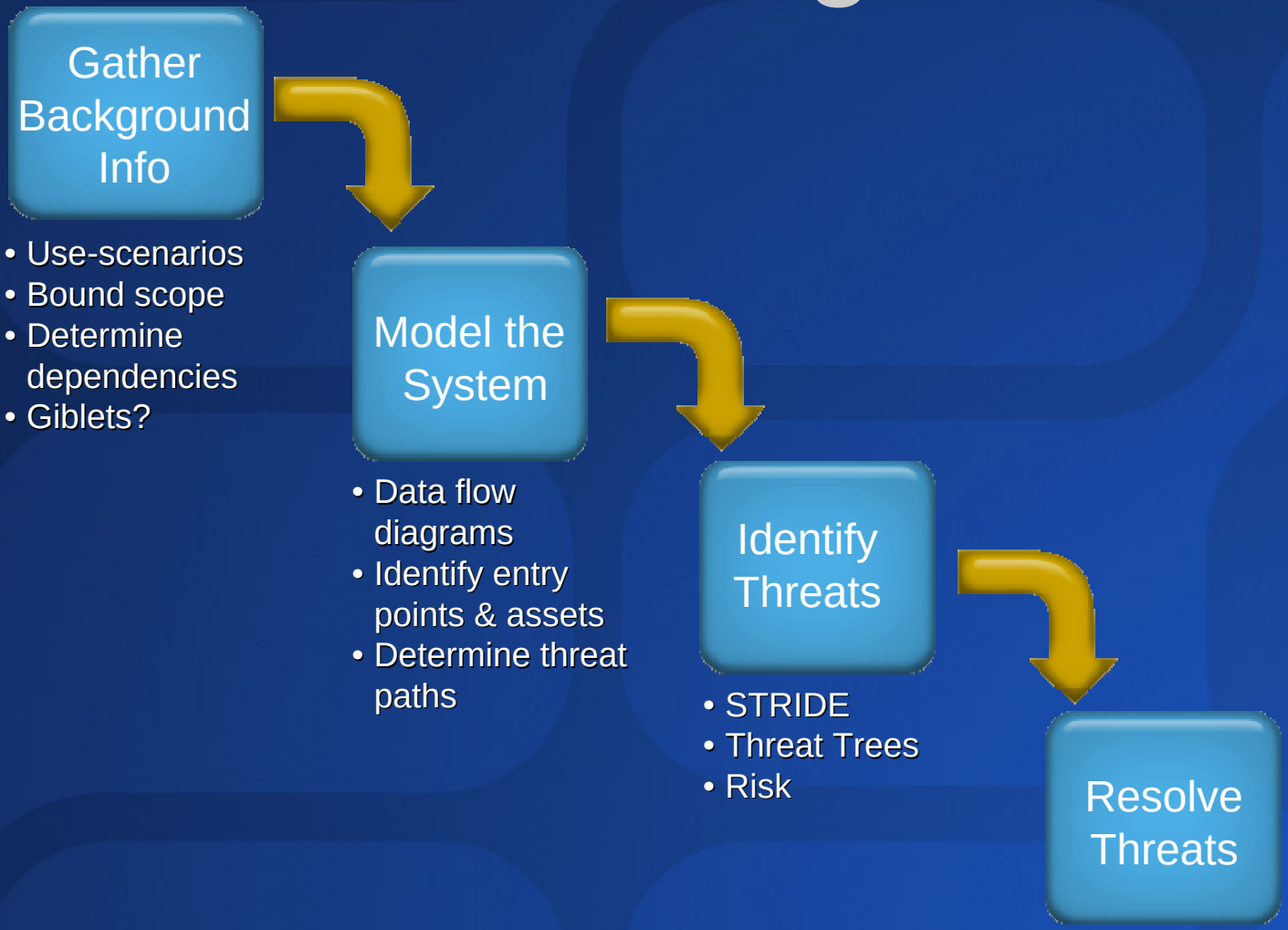
- Secure software starts with understanding the threats
- Threats are not vulnerabilities
- Threats live forever
- How will attackers attempt to compromise the system?

Benefits of Threat Modeling

- Better application understanding
- Helps finding bugs
- Identify complex design bugs
- Helps integrate new team members
- Drives well-designed security test plans

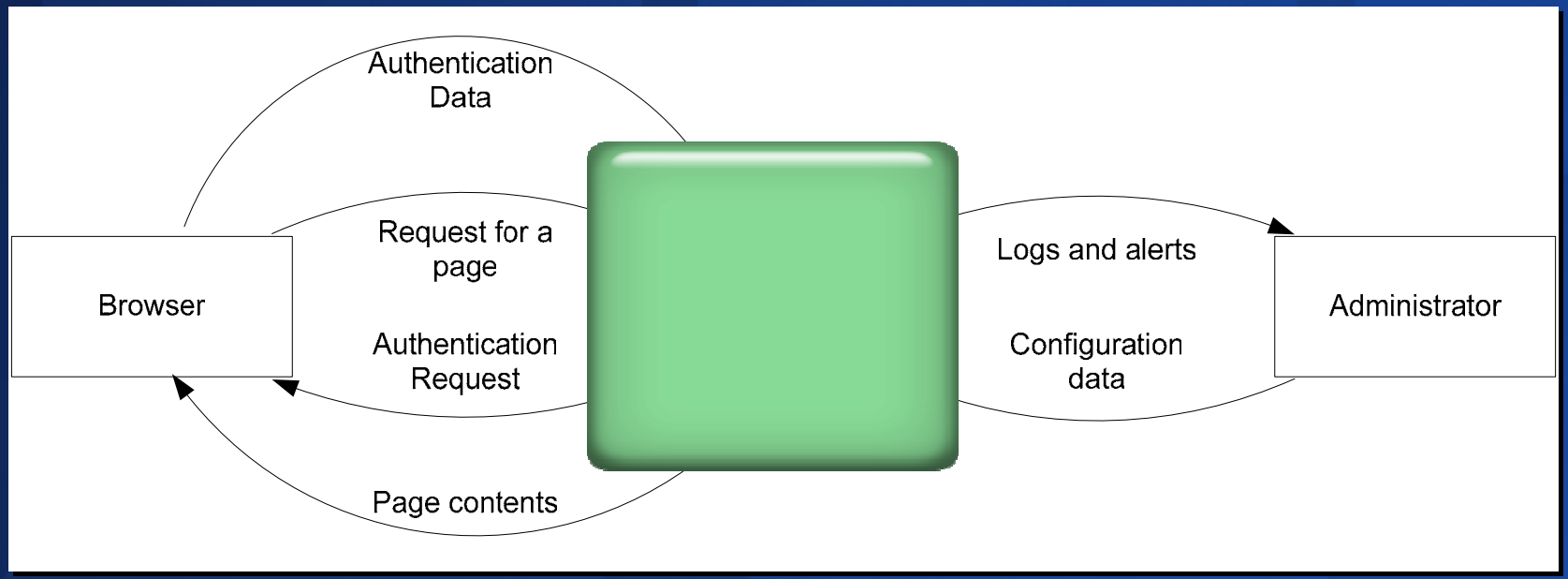


A Threat Modeling Process



“Threat Modeling”
Frank Swiderski

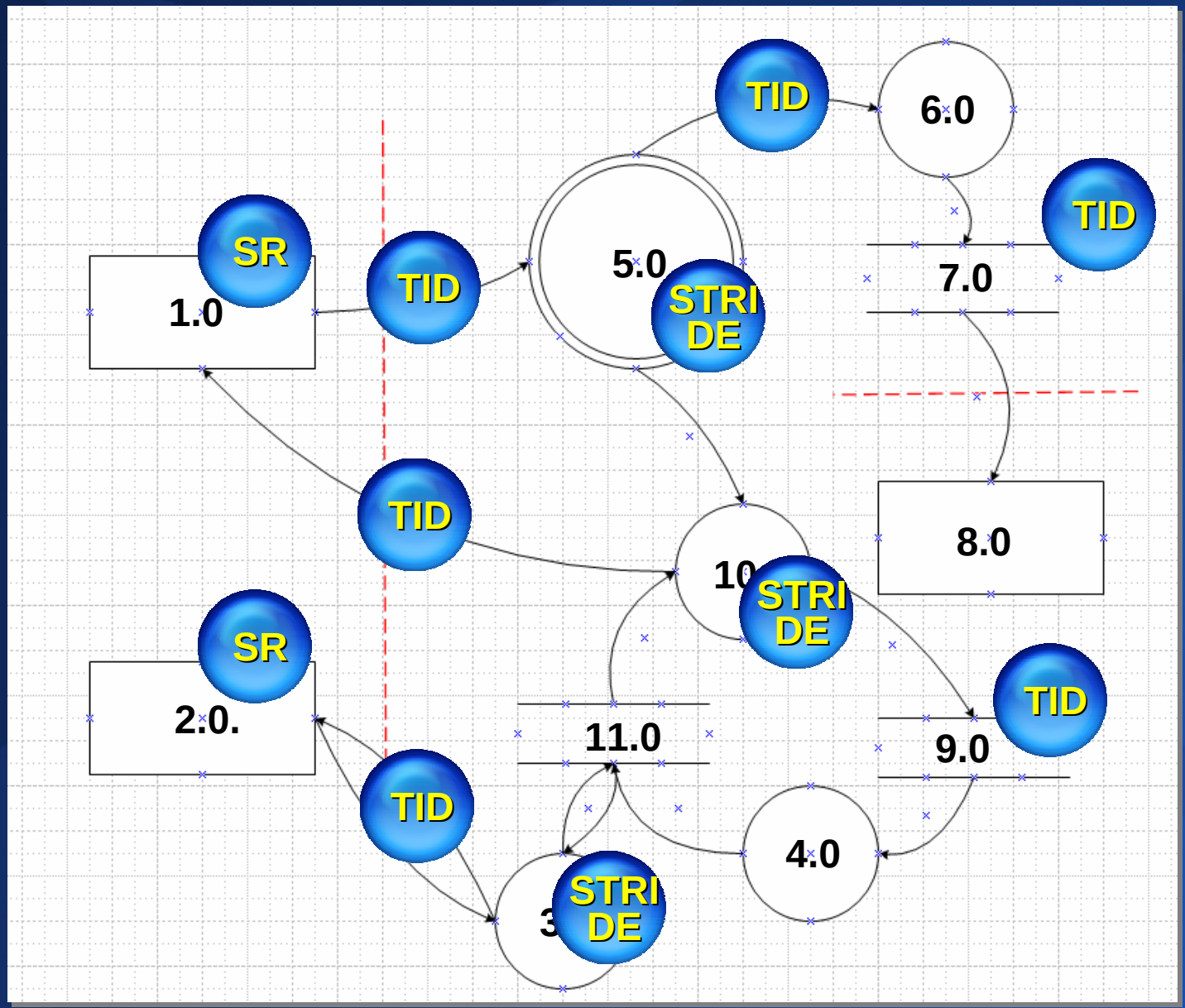
IIS6 Context Diagram



Identify the Threats

Types of threats	Examples
Spoofing	• Forging e-mail messages • Replaying authentication packets
Tampering	• Altering data during transmission • Changing data in files
Repudiation	• Deleting a critical file and deny it • Purchasing a product and deny it
Information disclosure	• Exposing information in error messages • Exposing code on Web sites
Denial of service	• Flooding a network with SYN packets • Flooding a network with forged ICMP packets
Elevation of privilege	• Exploiting buffer overruns to gain system privileges • Obtaining administrator privileges illegitimately

Determining Threat Types



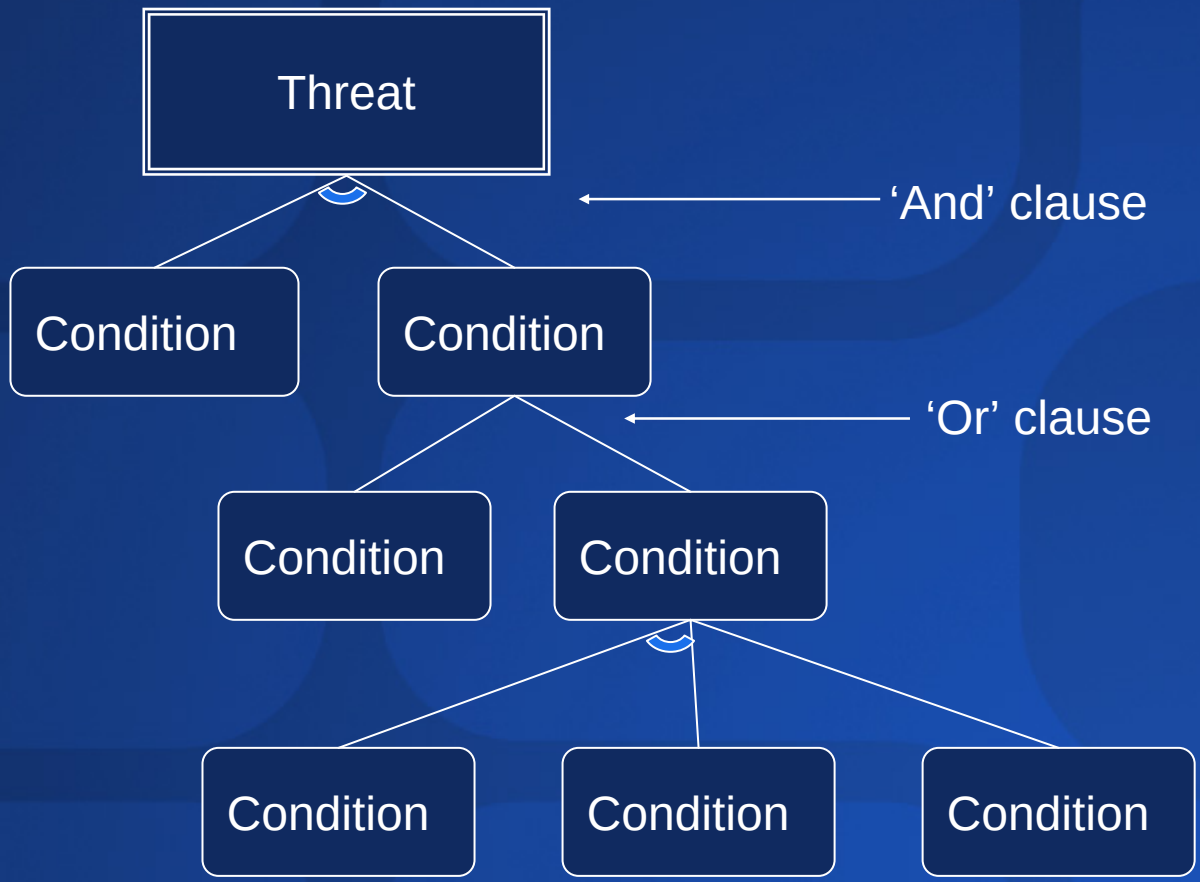
Each element in the DFD is susceptible to one or more threat types

DFD Elements are Threat Targets

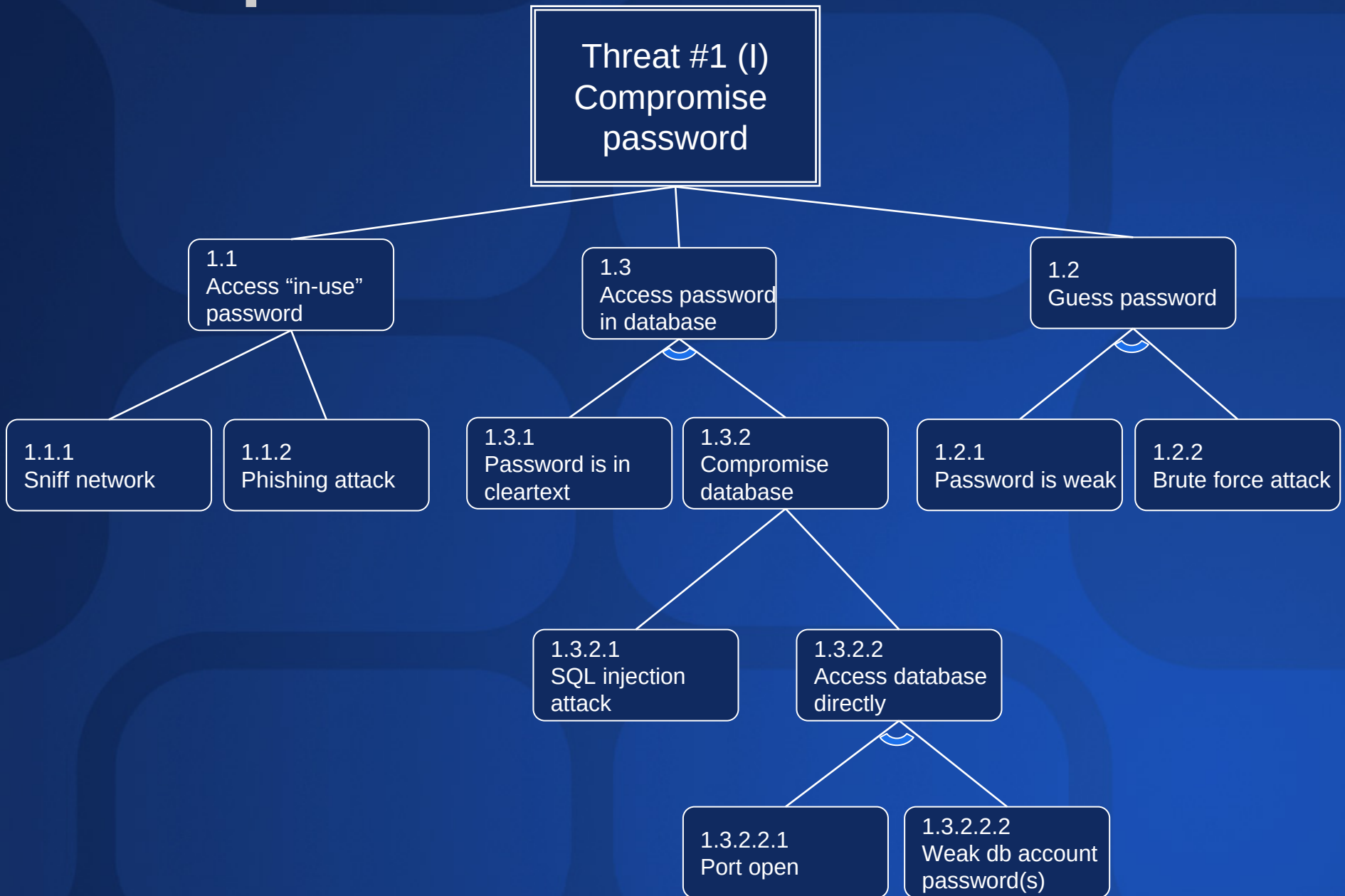
Data Flow	S	T	R	I	D	E
1 → 5		✓		✓	✓	
5 → 6		✓		✓	✓	
6 → 7		✓		✓	✓	
7 → 8		✓		✓	✓	
Data Store						
7		✓		✓	✓	
9		✓		✓	✓	
11		✓		✓	✓	
Interactor						
1	✓		✓			
2	✓		✓			
8	✓		✓			
Process						
3	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓

Each threat is governed by the conditions which make the threat possible

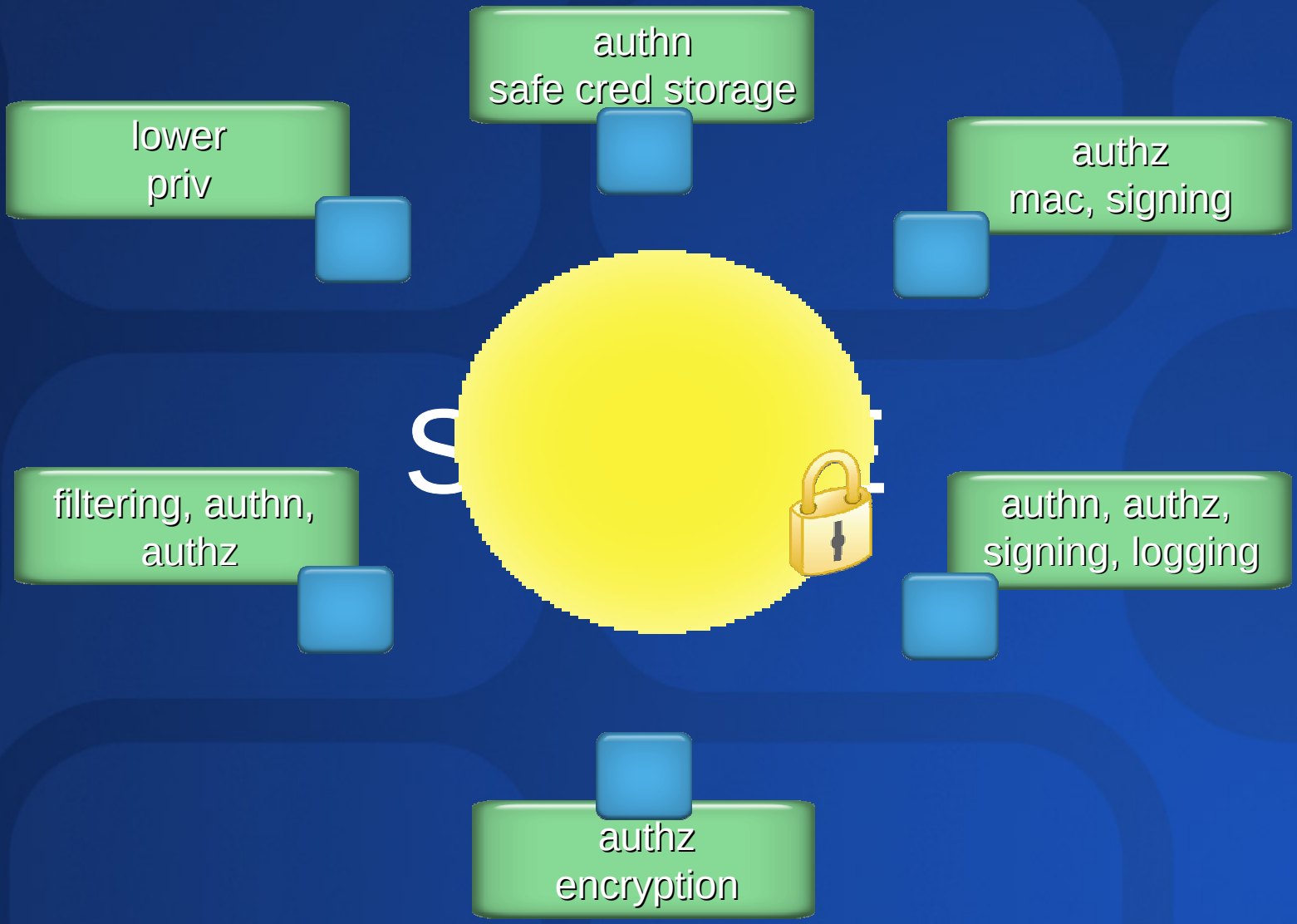
Threat Tree Format



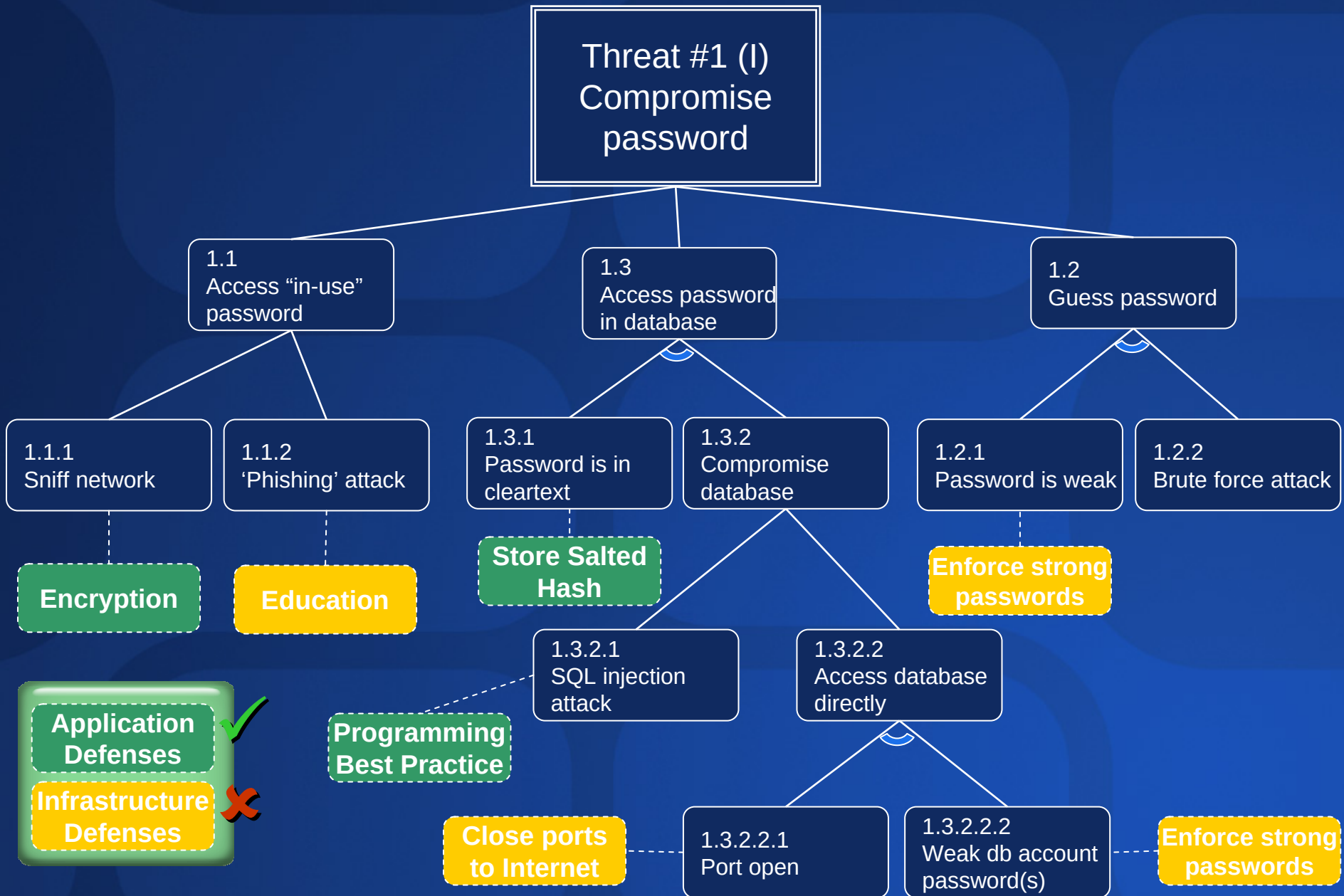
Sample Info Disclosure Threat



Designing Mitigations



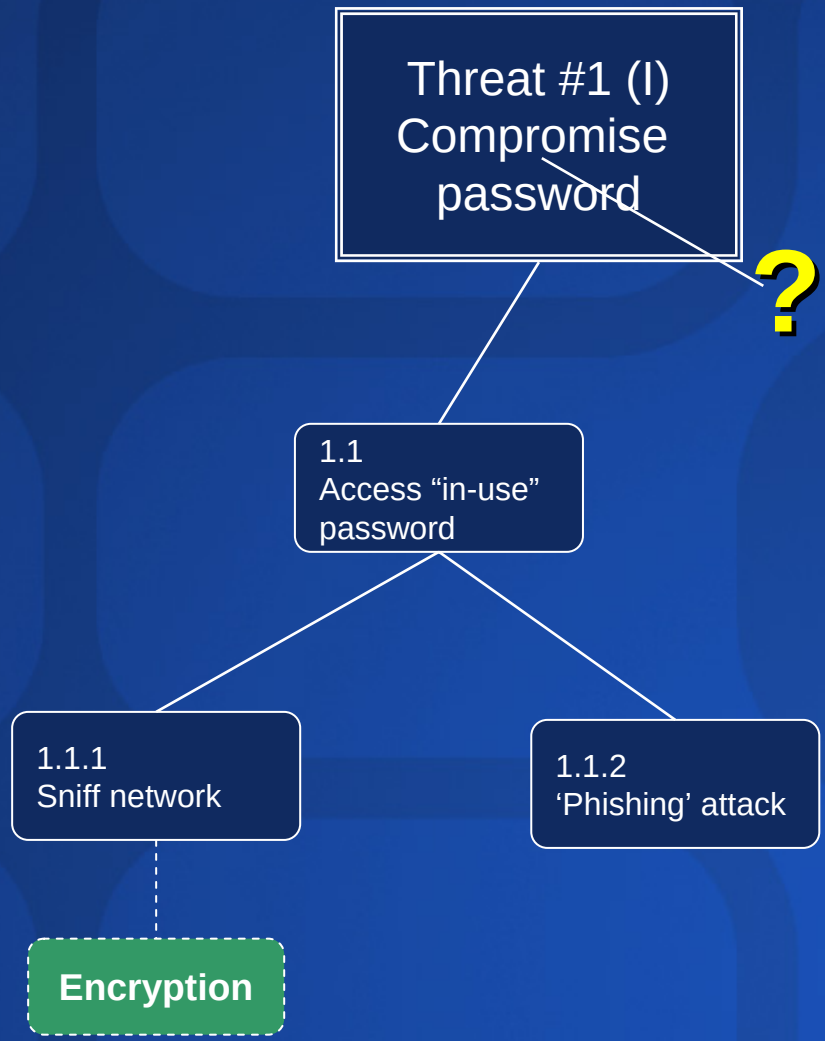
Sample I Threat Mitigations



Testing a Mitigation

- Functionally, is the traffic adequately protected?

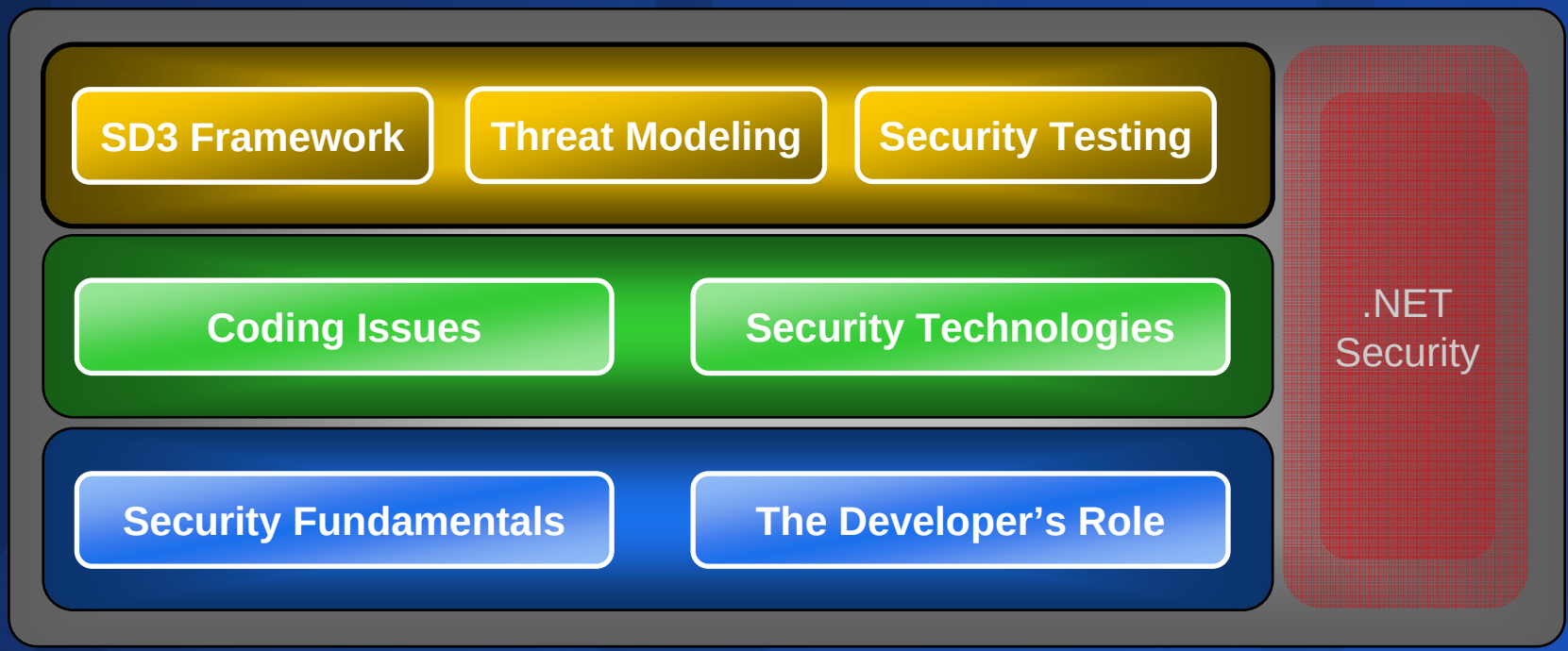
- Can you force unprotected traffic?
- Is the crypto weak?
- Where are keys stored?
- How are they exchanged?
- Defense in depth methods ok?
- Other conditions?



Threat Model Checklist

- [x] No design is complete without a threat model!
- [x] Follow anonymous data paths
- [x] Every threat needs a security test plan
- [x] Check all information disclosure threats – are they privacy issues?
- [x] Be wary of elevated processes

Security for Developer



Microsoft®

Your potential. Our passion.™