Secure From Scratch

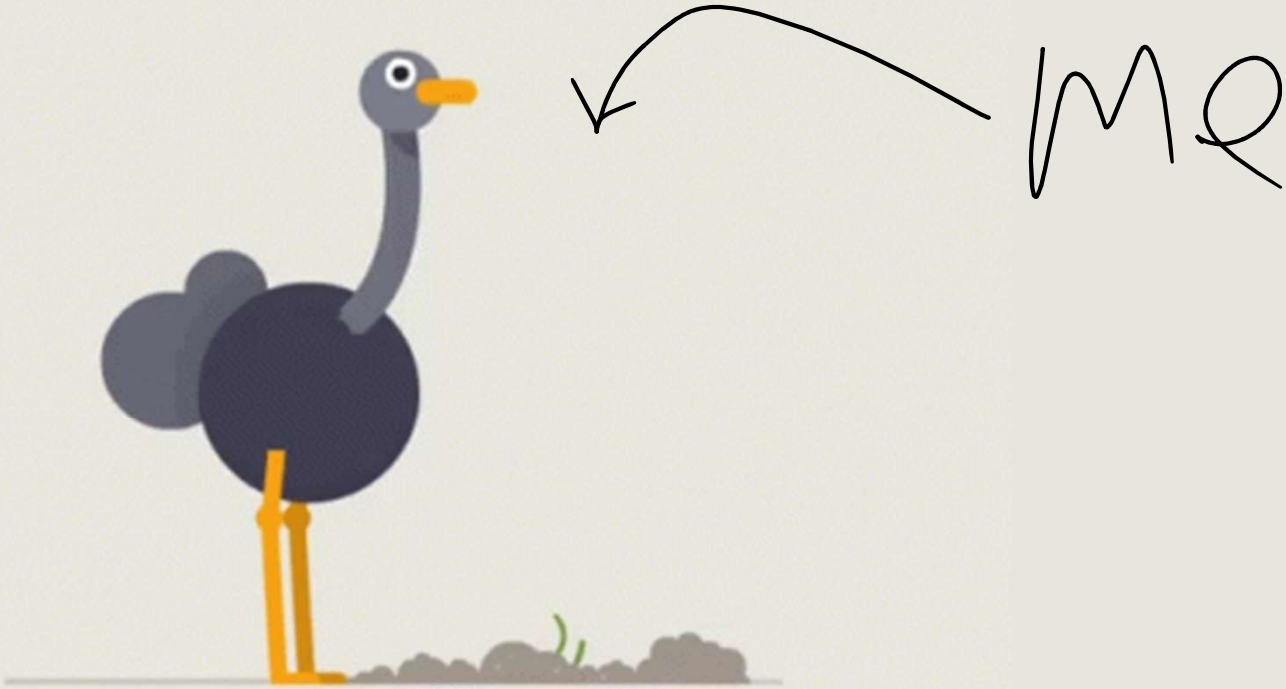
Programming with Security in Mind

After this workshop

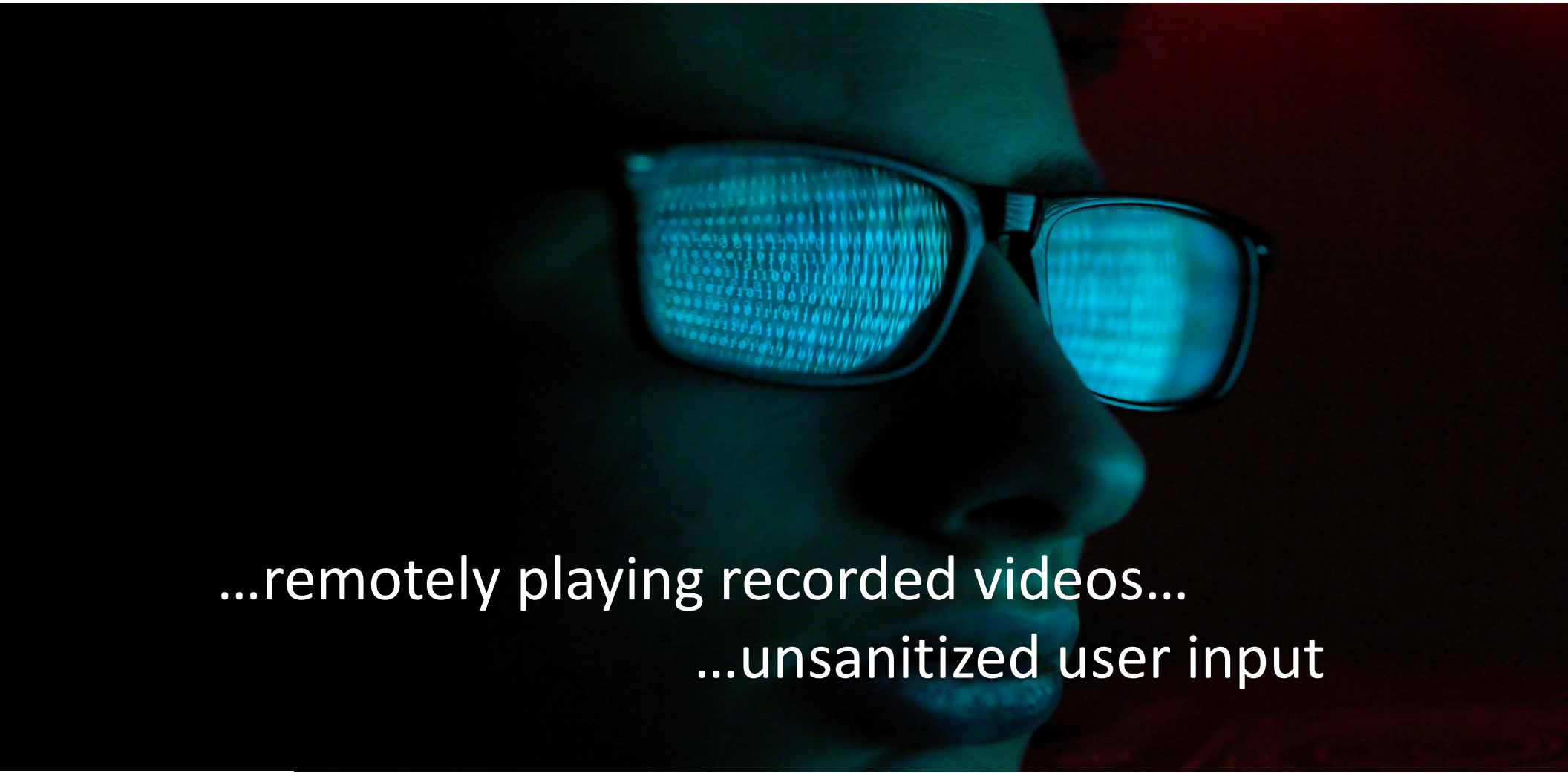
Write code in a way that prevents security vulnerabilities

(by the end of the lecture)

Secure From Scratch



Secure From Scratch



Secure From Scratch

Secure From Scratch

Demo

Secure From Scratch

Yariv Tal

University lecturer

Bootcamps mentor

Seasoned developer (WhyT software)

Application security researcher

Drug of choice: Travelling & Roller coasters

Secure From Scratch

Or Sahar

Senior security researcher

Secure code Instructor

Application security consultation and PT

A veteran developer

Drug of choice : CVEs & Snowy mountains

Getting to Know You

- Backend/Front end?
- Java/C#/Python/C++/NodeJS/Typescript/Other?
- Experience?
- Security knowledge?
- i++ or ++i?

Secure From Scratch

Unsanitized Input… Eh?!?

- Raise of hands, please!

What Is This Workshop About?

Secure From Scratch

This is a Secure Programming Workshop

It's a **mindset**

Can be applied to any programming language

Requires awareness

Programming, nothing more

- But more than just coding
- Reduces number of vulnerabilities – does not eliminate them

What is Secure Programming?

Programming that leads to ***less*** Security Bugs

Secure coding is … developing computer software in such a way that guards against the accidental introduction of *security vulnerabilities*.

Defects, bugs and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities.
… most vulnerabilities stem ***from a relatively small number of common software programming errors***.

Wikipedia

What is a Security Bug?

A bug that leads to a security vulnerability.

What is a Security Vulnerability?

So many (incomplete) definitions!

In OWASP We Trust:

A vulnerability is a hole or a weakness in the application .. that **allows an attacker to cause harm** to the stakeholders of an application.

Stakeholders include the application owner, application users, and other entities that rely on the application.

The Goal of Secure Programming

Less Security Bugs

→ Less Vulnerabilities

→ Less Exploitations

→ Less Harm

Secure From Scratch

The Goal of Secure From Scratch

Involve in the developers' careers ASAP

- Make Secure Programming the default
- Form good habits

Secure From Scratch

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exception

Neat code

Trust Boundaries

Secure From Scratch

Workshop Parts

1. The Need for Secure Coding
2. Easy to Use Safely
3. Building a Building Block

Part 1:
The Need

Cost of Vulnerabilities

A lot!

(Out of scope)

Part 2:
Easy to Use Safely

Secure From Scratch

Case In-Point – Committer App

A competitor to git/github

Save code version with a comment

Secure From Scratch

Case In-Point – Committer App

Created by Minnie

weclomeMsg.py
(version 1)

Version comment:
Show message to user

Modified by Mickey

weclomeMsg.py
(version 2)

Version comment:
Changed to upper case

Modified by Minnie

weclomeMsg.py
(version 3)

Version comment:
Added dot at end

Secure From Scratch

Case In-Point – Committer App

```
>committer.exe
Hello Yariv, please enter your commit message:
>>> Refactored with Secure From Scratch principles
Changes commited with message
Commit history:
initial commit
Refactored with Secure From Scratch principles
```

Stored in *shared* file *prevcommits.txt* (all developers see these messages)

Secure From Scratch

Case In-Point – Committer App

```
>committer.exe
Hello Yariv, please enter your commit message:
>>> This commit is bull shirt
Changes commited with message
Commit history:
initial commit
Refactored with Secure From Scratch principles
This commit is bull shirt
```

Inappropriate Word!

Secure From Scratch

LAB: Get to know the Committer app

15 min

1. Download the Committer App source code
2. Compile & Execute the Code.
3. Try *at least* these 4 messages: a, ab, abc, bull shirt
4. Try an empty message.
5. Look for the *prevcommits.txt* file and read it
6. Read the *commiter.log* file

https://github.com/SecureFromScratch/Summer2023/

Case In-Point – Committer App

<<<CODE OVERVIEW>>>

Handling Inappropriate Words

For simplicity:

Any 4 letter word is an inappropriate word.

Handling Inappropriate Words

Any 4 letter word is an inappropriate word.

Valid message example:

Got the XSO feature working, finally

Invalid message example:

Got the XSO bull shirt feature working

This sentence has two four letter words

Secure From Scratch

LAB: Block 4 letter words

15 min

1. Execute the Committer app
2. See that you can use 4 letter words
3. Rewrite the *IsValidCommitMsg* method to block 4 letter words
4. Test your code (for valid & invalid messages)

https://github.com/SecureFromScratch/Summer2023/

Committer App – Blocking 4 letter words

<<<CODE OVERVIEW>>>

Secure From Scratch

LAB: Learning How The Enemy Thinks

10 min

1. Find ways to bypass the protection.
   In other words: find ways to use 4 letter words
2. HINT: Think of what your code tests for
   (compared to what you want it to test for)

https://github.com/SecureFromScratch/Summer2023/

Committer App – Exploitation 1

Change the file using notepad

Not fair?

Is this your problem?

<<DISCUSSION>>

Committer App – Exploitation 2

Did you test for 4 *letter* words?

Well, hackers won't have to constrain themselves:

Bull.shirt

.FORK

F_O_R_K

Not fair?

Committer App – Exploitation 2

Can happen from normal speech:

Forking bad

Things are forked

Go fork!

It is a big fork, but it is all I can do

<<DISCUSS>>

Exploitation 2 – The Problem is Blocking

Alternative: Welcome Lists (aka Allow Lists)

Create rules for what is *allowed*, not what is *disallowed*

Exploitation 2 – Prevent with Welcome Lists

Create rules for what is *allowed*.

Examples:

Fixed chat dialog. Performance also improved.
No longer requires login for the home page.
Improved code readability.

<<DISCUSS>>

Exploitation 2 – Prevent with Welcome Lists

It’s a lot of work.

But:

P – Security is the first Priority

R

E – Easy to use *safely*

V

E

N

T

Secure From Scratch

What's so bad about blocking?

Almost all attempts to get it right are doomed.

There is *always* something you did not think about.

Committer App – Exploitation 3

Characters and Letters are not the same thing!

You might think that strings hold letters, but they don’t

(Re)introducing the ASCII table:

The ASCII Table

```
$ ascii -d
   0 NUL    16 DLE    32       48 0    64 @    80 P    96 `   112 p
   1 SOH    17 DC1    33 !     49 1    65 A    81 Q    97 a   113 q
   2 STX    18 DC2    34 "     50 2    66 B    82 R    98 b   114 r
   3 ETX    19 DC3    35 #     51 3    67 C    83 S    99 c   115 s
   4 EOT    20 DC4    36 $     52 4    68 D    84 T   100 d   116 t
   5 ENQ    21 NAK    37 %     53 5    69 E    85 U   101 e   117 u
   6 ACK    22 SYN    38 &     54 6    70 F    86 V   102 f   118 v
   7 BEL    23 ETB    39 '     55 7    71 G    87 W   103 g   119 w
   8 BS     24 CAN    40 (     56 8    72 H    88 X   104 h   120 x
   9 HT     25 EM     41 )     57 9    73 I    89 Y   105 i   121 y
  10 LF     26 SUB    42 *     58 :    74 J    90 Z   106 j   122 z
  11 VT     27 ESC    43 +     59 ;    75 K    91 [   107 k   123 {
  12 FF     28 FS     44 ,     60 <    76 L    92 \   108 l   124 |
  13 CR     29 GS     45 -     61 =    77 M    93 ]   109 m   125 }
  14 SO     30 RS     46 .     62 >    78 N    94 ^   110 n   126 ~
  15 SI     31 US     47 /     63 ?    79 O    95 _   111 o   127 DEL
```

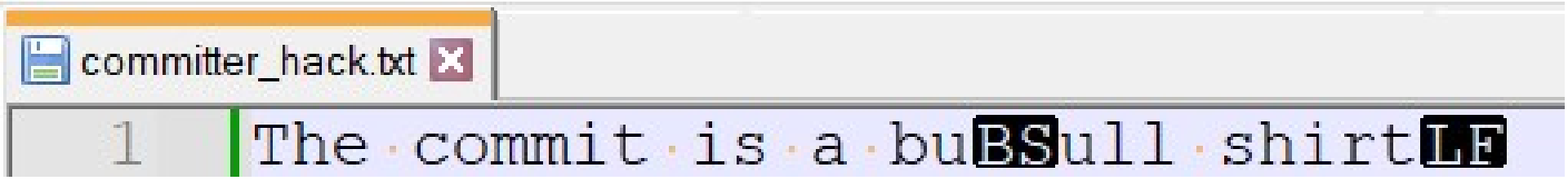
Committer App – Exploitation 3

How do I type Ctrl Characters?

Use a Hex Editor

Write a program that outputs them

…



Committer App – Exploitation 3

How do I input Ctrl Characters?

Use input redirection:

```
>committer.exe < \temp\committer_hack.txt
Hello Yariv, please enter your commit message:
>>> Changes commited with message
Commit history:
initial commit
Refactored w/ Secure frm Scratch principles
The commit is a bull shirt
```

LAB Conclusions

Input ➔ Bad

LAB Conclusions

Input ➔ Danger

Blocking leads to unexpected bypasses

Part 3:
Building Blocks

We No Longer Need Control Characters

They cause so many bugs

But, they are still supported by every language:
String

So, let’s get rid of them

How Would We Get Rid of Control Characters?

<<DISCUSS>>

Control Characters – The Solution

GOAL: Automatically Block Control Characters

SOLUTION: Replace class String

class TextLine:

- Will block any control character
- Validation: Notify with an error
- Sanitization: Erase control characters

Secure From Scratch

LAB: TextLine class (partial) implementation

15 min

1. Write a TextLine class
   - It should have a BlockCtrlChars(str) method
2. See that you block control characters
3. Create an application that uses TextLine class
4. Test your code

Secure From Scratch

Easy to Use Safely

Don’t rely on your memory

- Don’t require writing additional safety checks

Change *how* you code

- Safety should be inherent in your writing

~~String~~ → TextLine

Secure From Scratch

Summary

Coding errors => Security vulnerabilities

Change our mindset

Easy to Use Safely from PREVENT

Secure From Scratch

PREVENT - SFS Principles

Priority – Security is the first priority

Reporting & logging

Easy to use safely

Verify

Errors & exception

Neat code

Trust Boundaries

Call For Action

- Deepen your PREVENT
- Contribute PREVENT libraries & components
- Implement PREVENT @ work
- Share your experiences

Secure From Scratch

YouTube Channel

https://youtube.com/@SecureFromScratch

LinkedIn Page

https://www.linkedin.com/company/secure-from-scratch/

Open Source GIT

https://github.com/**SecureFromScratch/Summer2023**