The User Supply Chain Attacks and Defenses

2nd KTH Workshop on the Software Supply Chain 2023

Christian Collberg
University of Arizona

End User Supply Chain Attacks and Defenses

2nd KTH Workshop on the Software Supply Chain 2023

Christian Collberg
University of Arizona

Supply Chain Integrity

Developer

User

```
main(){
}
```

GitHub

Build

gcc

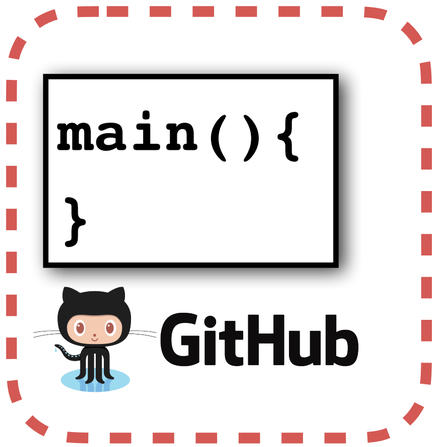
Package

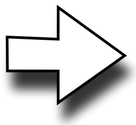
Use

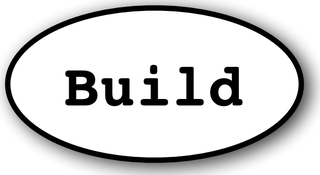
Supply Chain Integrity

Developer

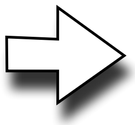
User

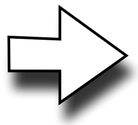


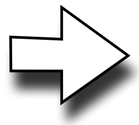


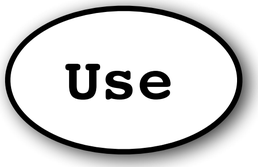


gcc





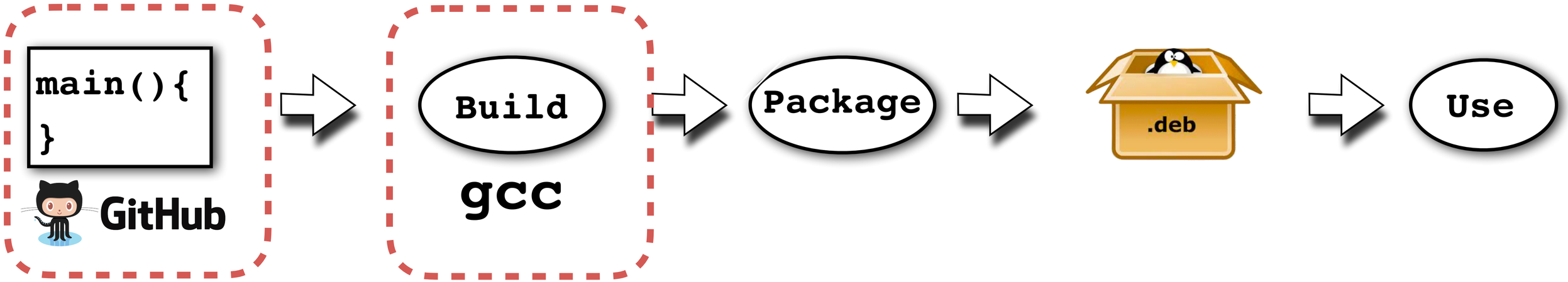




Supply Chain Integrity

Developer

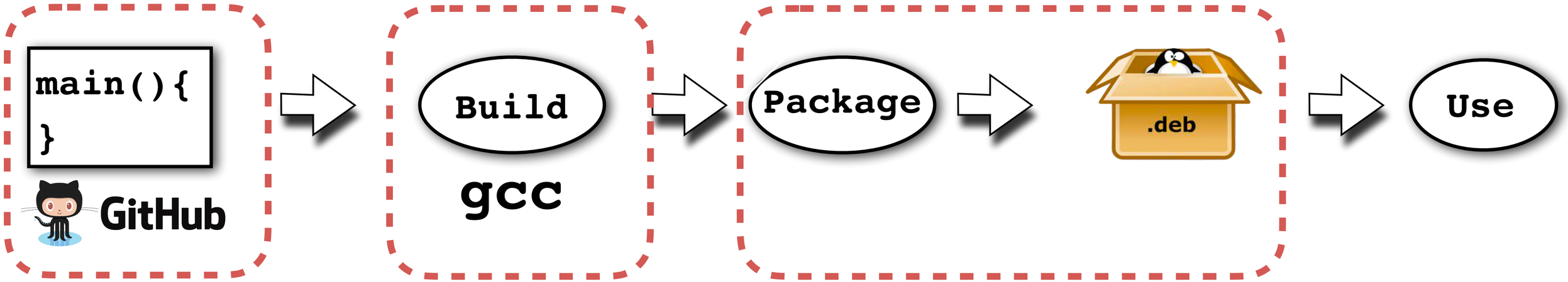
User



Supply Chain Integrity

Developer

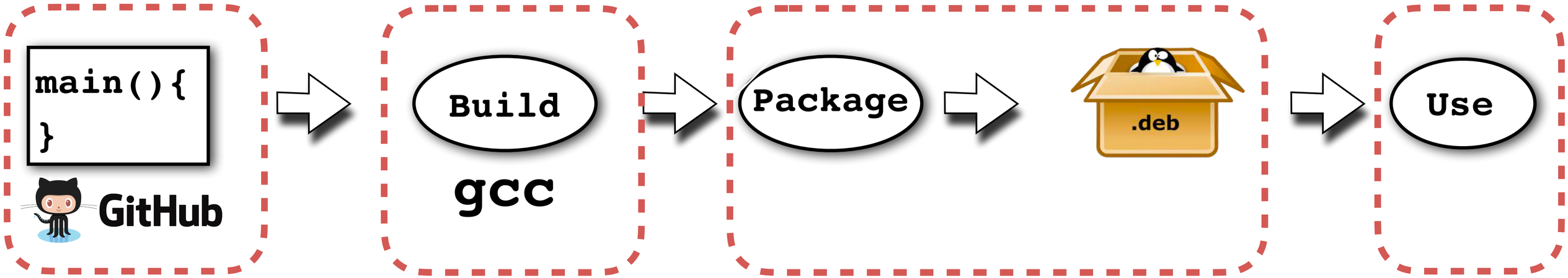
User



Supply Chain Integrity

Developer

User

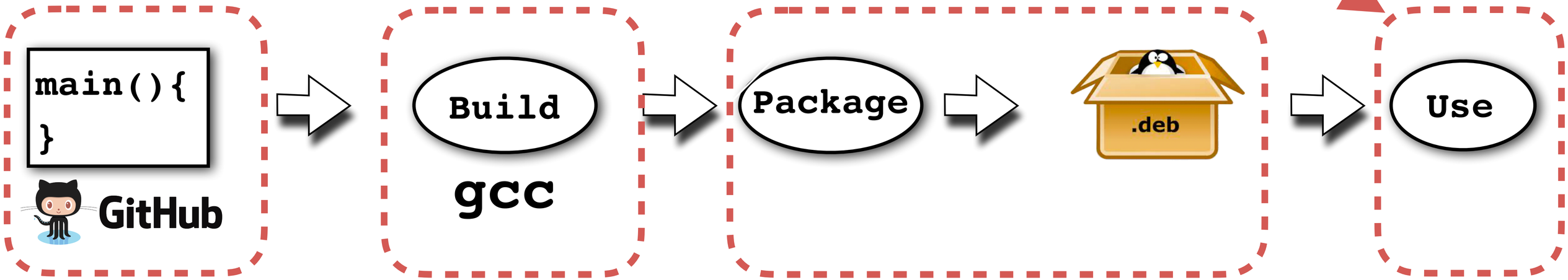


Supply Chain Integrity

Developer

All users should execute the same bits as the developer built and tested

User



Supply Chain Integrity

All users should execute the same bits as the developer built and tested

Developer

User

```
main(){
}
```

GitHub

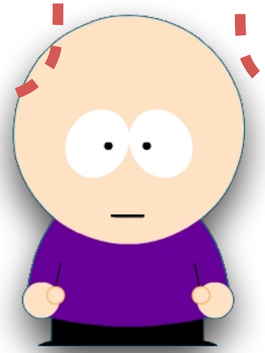
Build

gcc

Package

Use

My definition!



Supply Chain Integrity

Developer

All users should execute the same bits as the developer built and tested

```
main(){
}
```

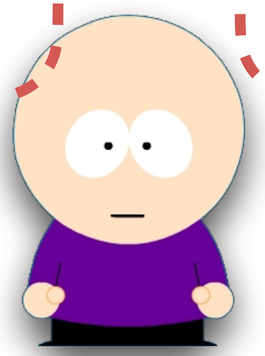
GitHub

Build

gcc

Package

My definition!



Supply Chain Integrity

All users should execute the same bits as the developer built and tested

Developer

User

```
main(){
}
```

GitHub

Build

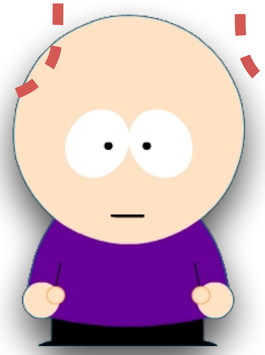
gcc

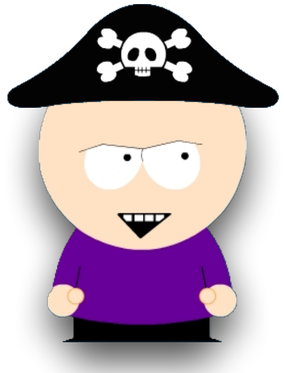
Package

.deb

Use

My definition!

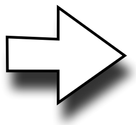


Supply Chain  Attacks

Developer

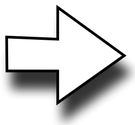
```
main(){
}
```

GitHub

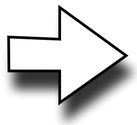


Build

gcc



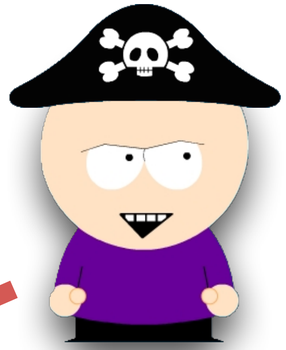
Package



Use

Supply Chain Attacks

Developer



```
main(){
}
```

GitHub

Submit Bad Code

Build

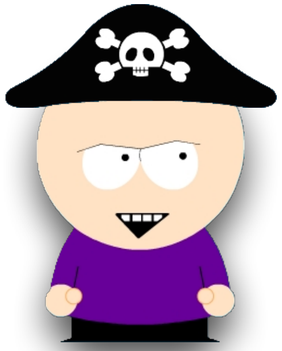
gcc

Package

Use

Supply Chain Attacks

Developer



```
main(){
}
```

GitHub

Build

gcc

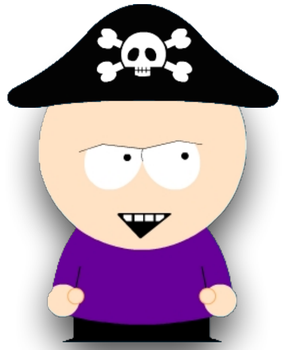
Compromise Build Tools

Package

Use

Supply Chain Attacks

Developer



```
main(){
}
```

GitHub

Build

gcc

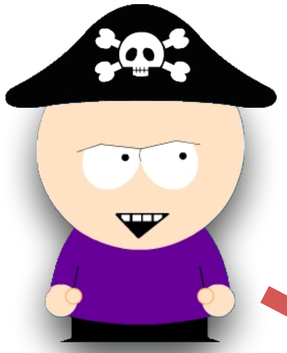
Package

Use

Compromise
Package Repository

Supply Chain Attacks

Developer



```
main(){
}
```

GitHub

Build

gcc

Package

Use

Compromise Installed Applications

Attack: Submit Bad Code

Developer

```
buggy_package(){
    char buffer[10];
    buffer = gets();
}
```

```
main(){
}
```

GitHub

Attack: Submit Bad Code

Developer

Oops, if you use my package you get my bug!

Open Source Developer

GitHub

```
buggy_package(){
    char buffer[10];
    buffer = gets();
}
```

```
main(){
}
```

GitHub

Attack: Submit Bad Code

Developer

Oops, if you use my package you get my bug!

Open Source Developer

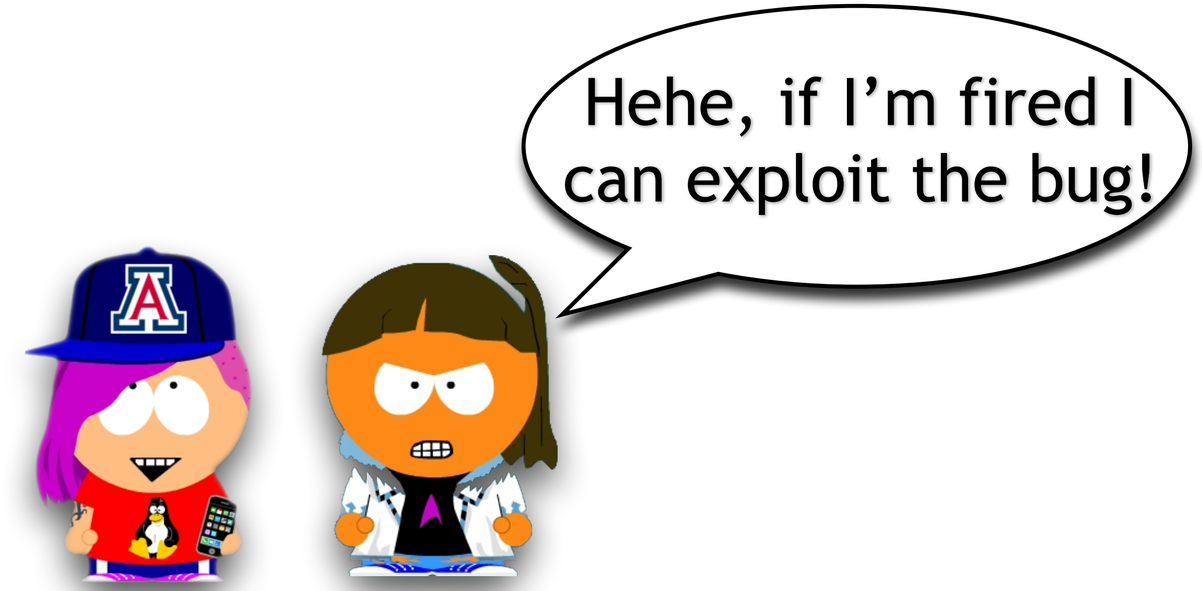
GitHub

```
main(){
}
```

GitHub

```
buggy_package(){
    char buffer[10];
    buffer = gets();
}
```

Attack: Submit Bad Code



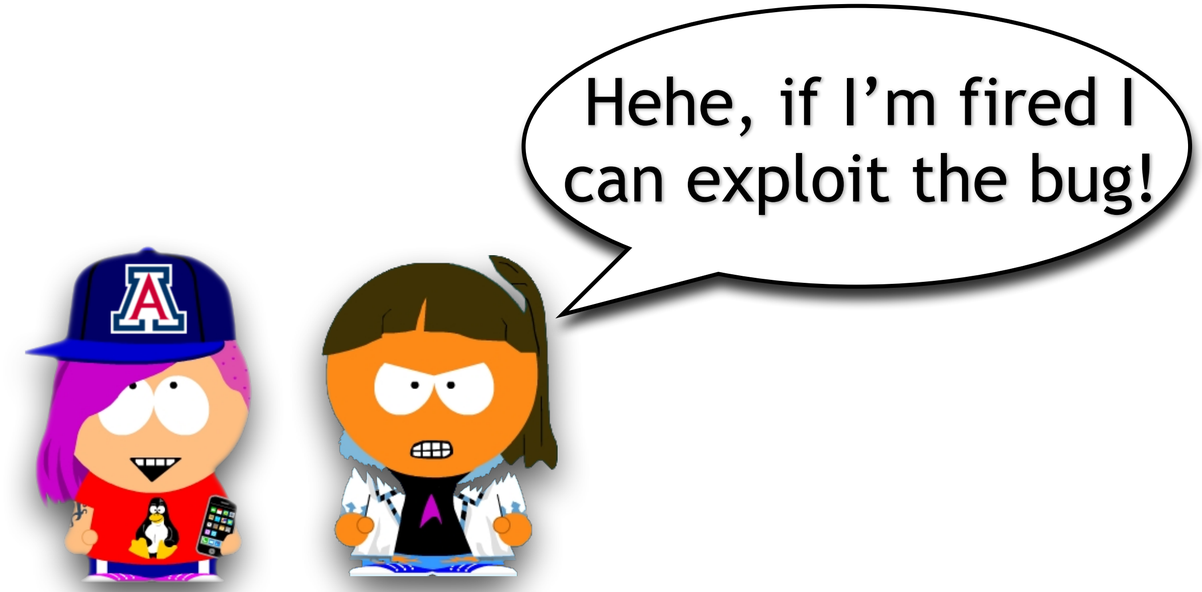
Developer Insider

```
main(){
}
```

GitHub

```
buggy_package(){
    char buffer[10];
    buffer = gets();
}
```

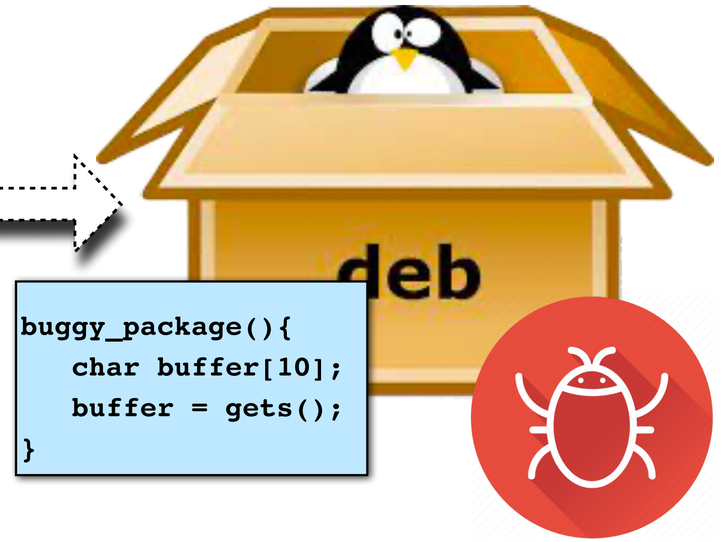
Attack: Submit Bad Code



Developer Insider

```
main(){
}
```

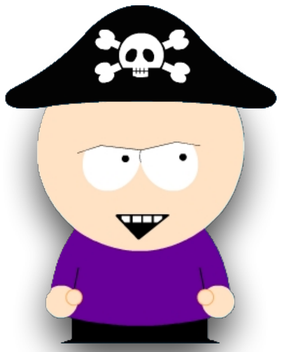
GitHub



Attack: Compromise Build Tools

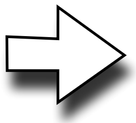
Developer

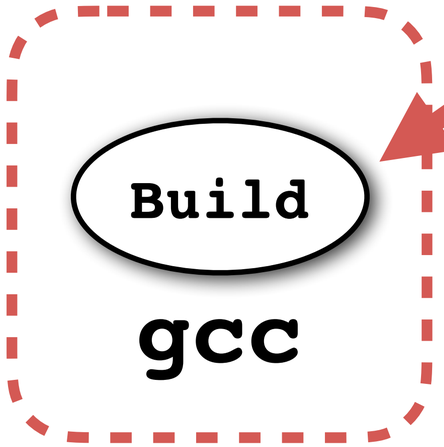
```
gcc(){

      insert_backdoor()

}
```



Compiler Developer

```
main(){
}
```

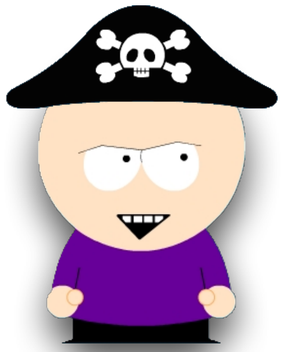




Attack: Compromise Build Tools

Developer

```
gcc(){

      insert_backdoor()

}
```

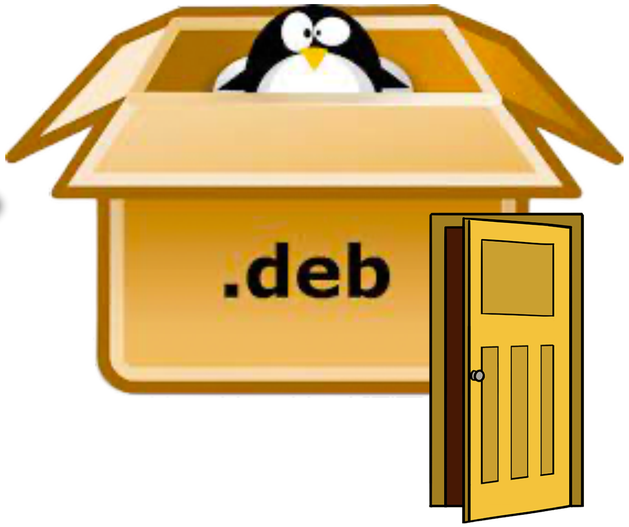


Compiler Developer

```
main(){
}
```

Build

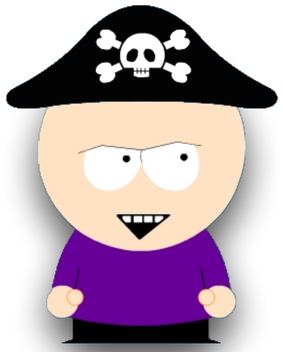
gcc



Attack: Compromise Build Tools

Developer

```
gcc(){
    if (compiling “login.c”){
        insert_backdoor()
    }
}
```



Compiler Developer

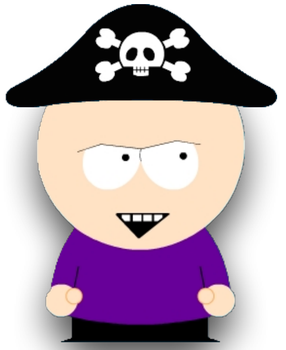
```
main(){
}
```

Build

gcc

Attack: Compromise Installed Apps

Developer



```
main(){
}
```

Build

Package

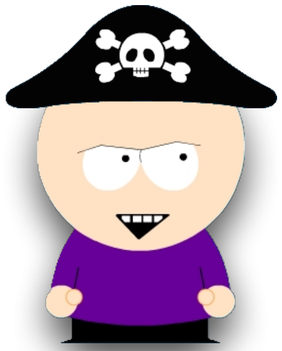
Use

Attack: Compromise Installed Apps

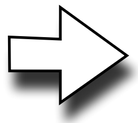
Security Checks

```
main(){
  if (!check())
    abort()
}
```

Developer

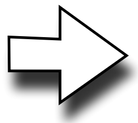


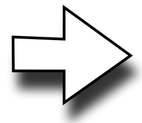
```
main(){
}
```



Build

Package





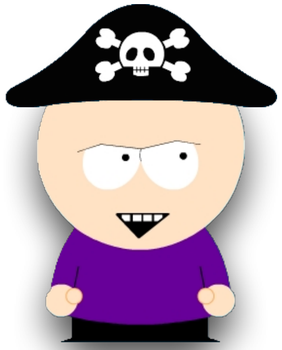
Use

Attack: Compromise Installed Apps

Security Checks

```
main(){
  if (                )
     abort()
}
```

Developer



```
main(){
}
```

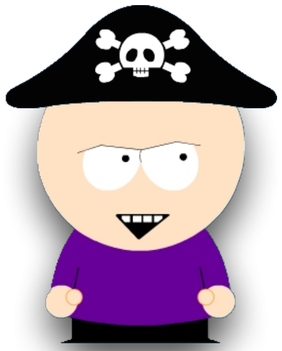
Build → Package → .deb → Use

Attack: Compromise Installed Apps

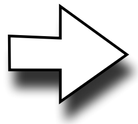
Developer

Confidential Data

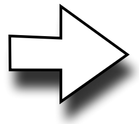
```
main(){
  key=0x14AF2;


}
```



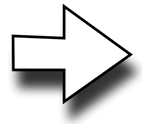
```
main(){
}
```



Build



Package



Use

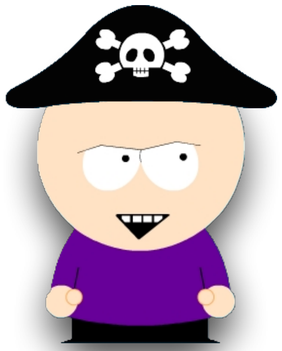
Attack: Compromise Installed Apps

Confidential Data

```
main(){
  key=0x14AF2;


}
```

0x14AF2;

Developer



```
main(){
}
```

Build

Package

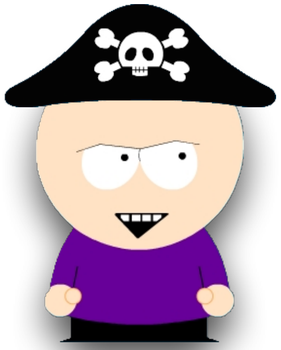
Use

Attack: Compromise Installed Apps

Intellectual Property

Developer

```
main(){
  secret_alg();


}
```



```
main(){
}
```

Build

Package

Use

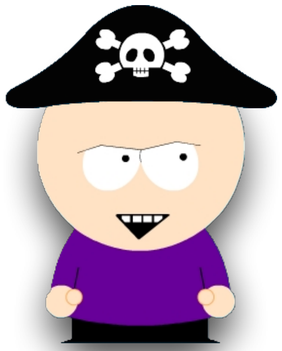
Attack: Compromise Installed Apps

Intellectual Property

Developer

```
main(){
  secret_alg();


}
```

secret_alg



```
main(){
}
```

Build

Package

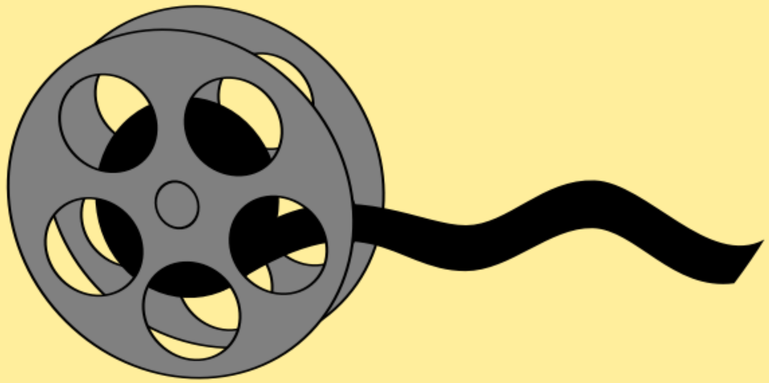
Use

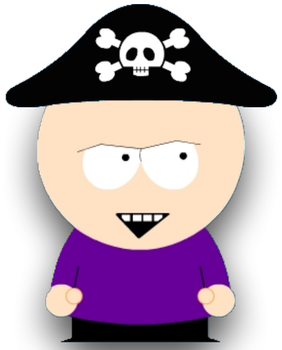
Attack: Compromise Installed Apps

Developer

Proprietary Media

```
main(){
}
```

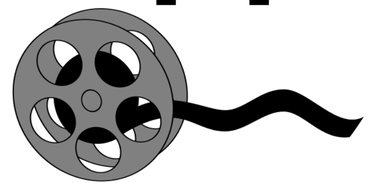




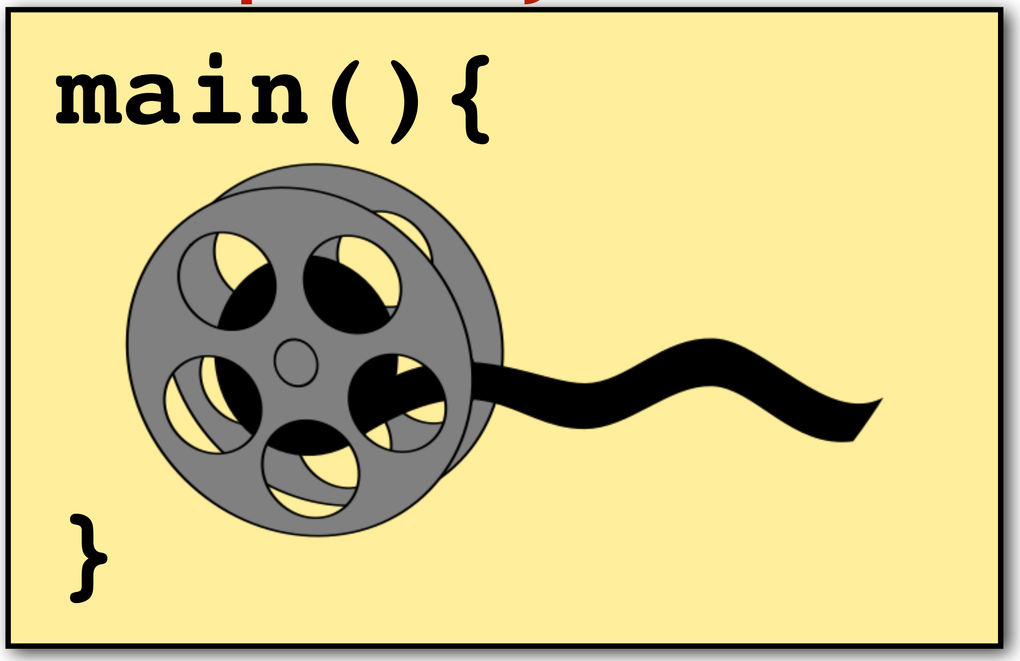
```
main(){
}
```

Build → Package → .deb → Use

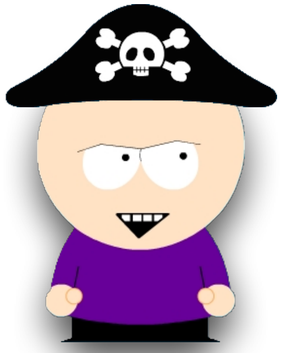
Attack: Compromise Installed Apps

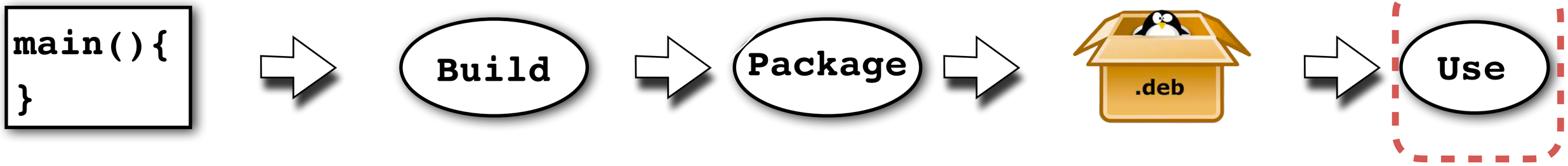


Proprietary Media



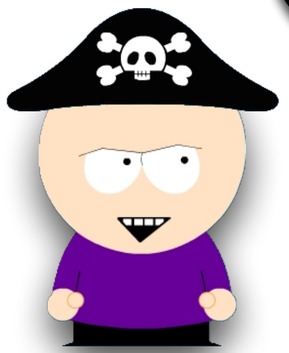
Developer





Attack: Illegal Application Reuse

Developer



Hehe, I can repackage the app and sell it as my own!

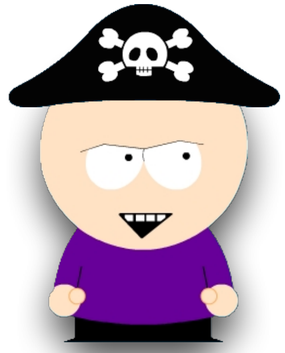
```
main(){
}
```


```
main(){
}
```

Today: End User Compromise

Developer

```
main(){
   if (!check())…
                key="0x1AF2…"
   secret_alg();

}
```

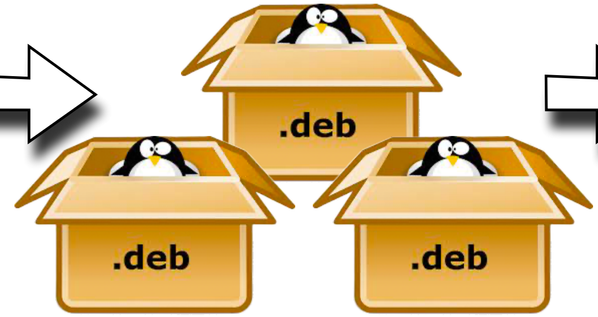


```
main(){
}
```

Build

gcc

Package

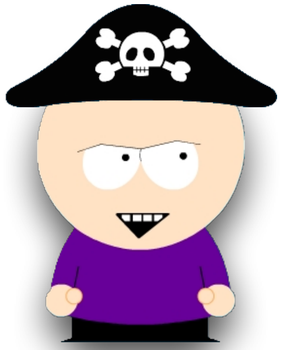


Use

Today: End User Compromise

Developer

```
main(){

                key="0x1AF2…"
  secret_alg();

}
```

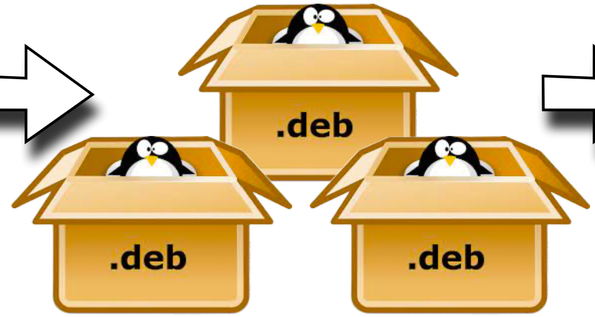


```
main(){
}
```

Build

gcc

Package

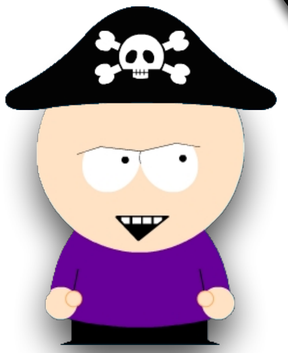


Use

Today: Illegal Application Reuse

Developer

Oh no you can't! Because I'm tracking you! Hah!



Hehe, I can repackage the app and sell it as my own!

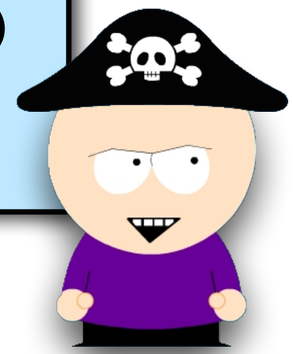
```
main(){
}
```


```
main(){
}
```

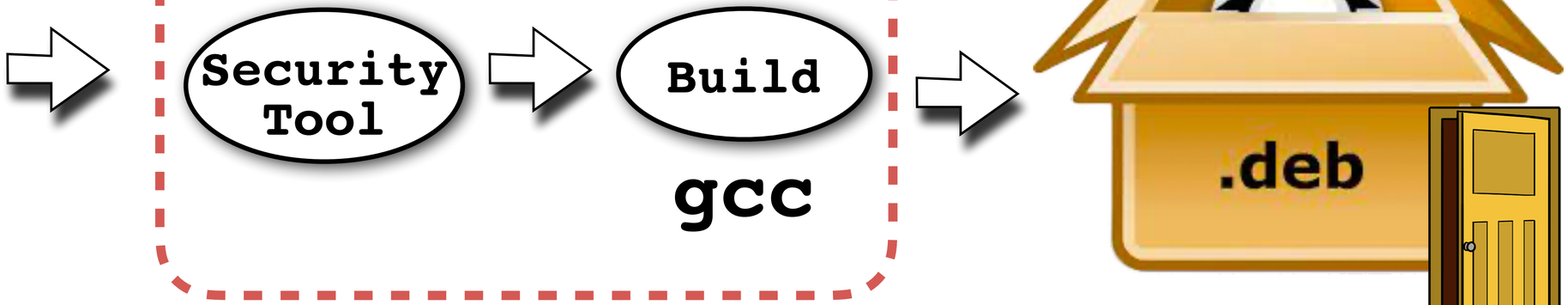
Today: Compromise Security Tool

Developer

```
protect(){
    hide(insert_backdoor())
}
```

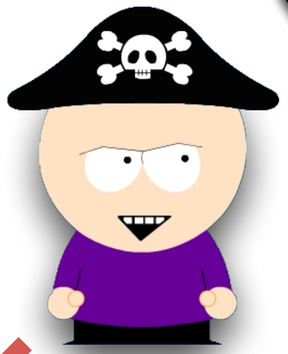


```
main(){
}
```



Today: End User Protection Tools

Developer



Hahaha, software protection tools are so sweet to p0wn!

```
main(){
}
```

Tigress

MATE Protect

Build

gcc

```
main(){


}
```

End User

Sketch

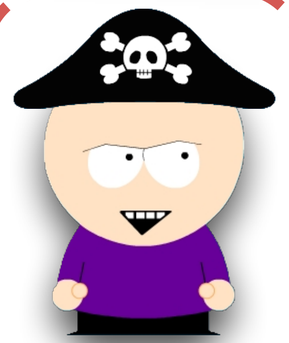
End User Attacks

Man-At-The-End (MATE) attacks occur in any setting where an adversary has physical access to a device and compromises it by inspecting, reverse engineering, or tampering with its hardware or software.

```
main(){
}
```

Build

Package



Use

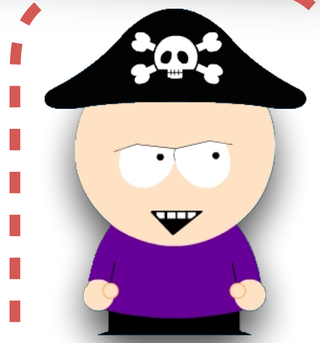
Attacks on

- **Confidentiality**: steal secrets from our code
- **Integrity**: modify our code
- **Availability**: gain control of the distribution of our code

```
main(){
}
```

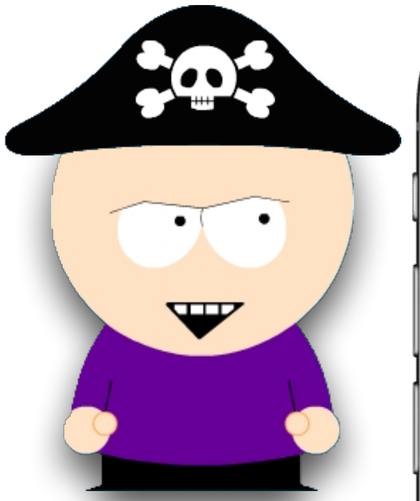
Build

Package

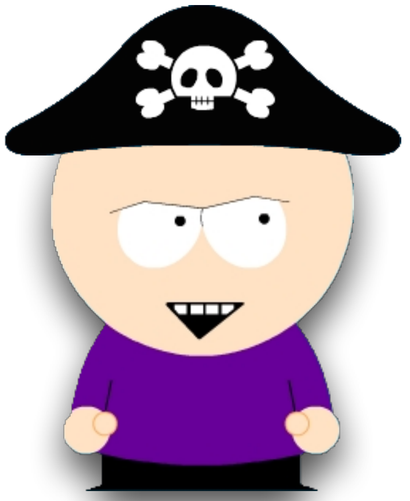


Use

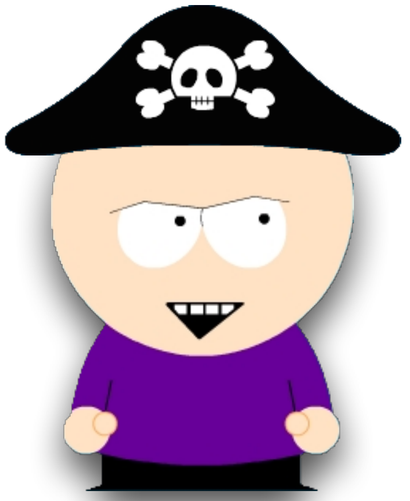

9:41
Tuesday, September 12

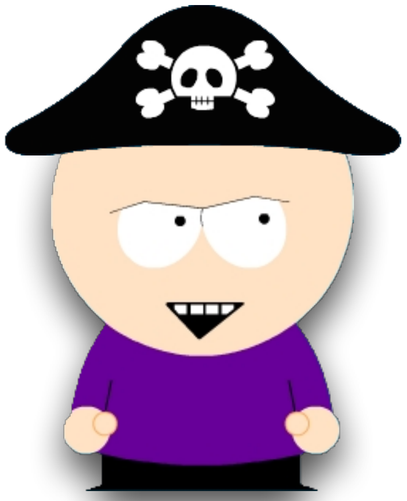



9:41
Tuesday, September 12

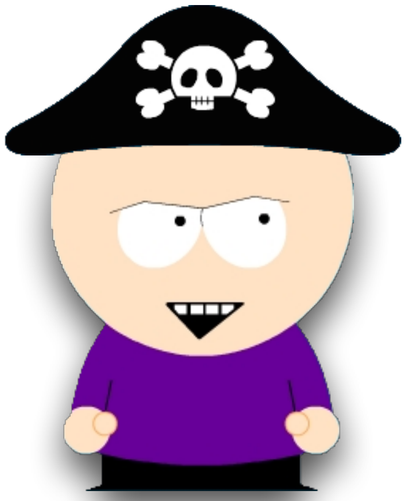


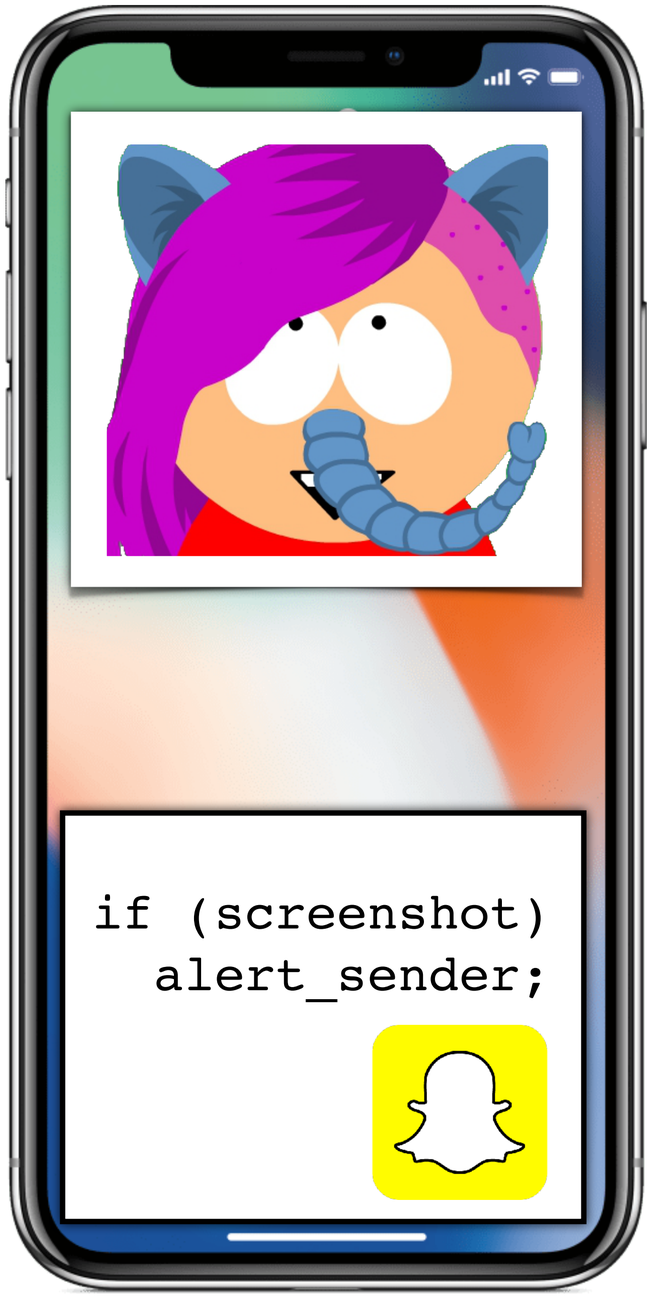

9:41
Tuesday, September 12



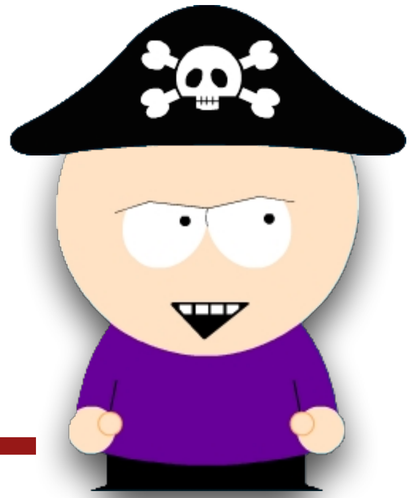


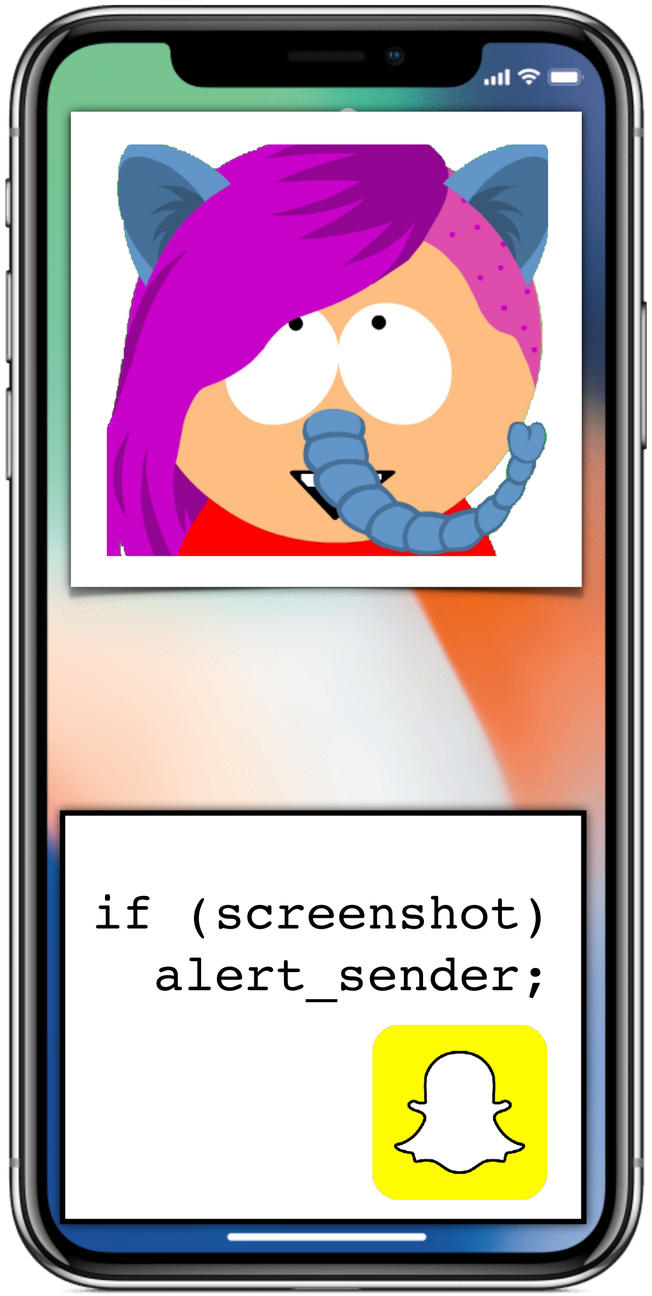

snapchat
Bob took a screenshot!



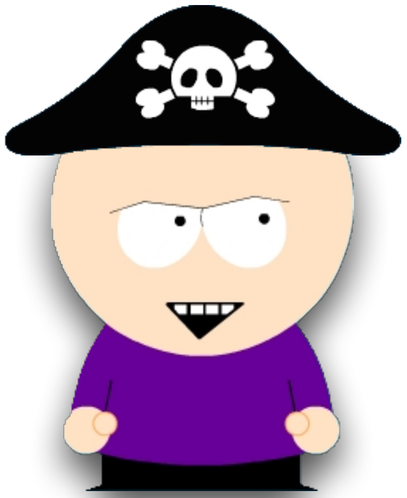

if (screenshot)
alert_sender;

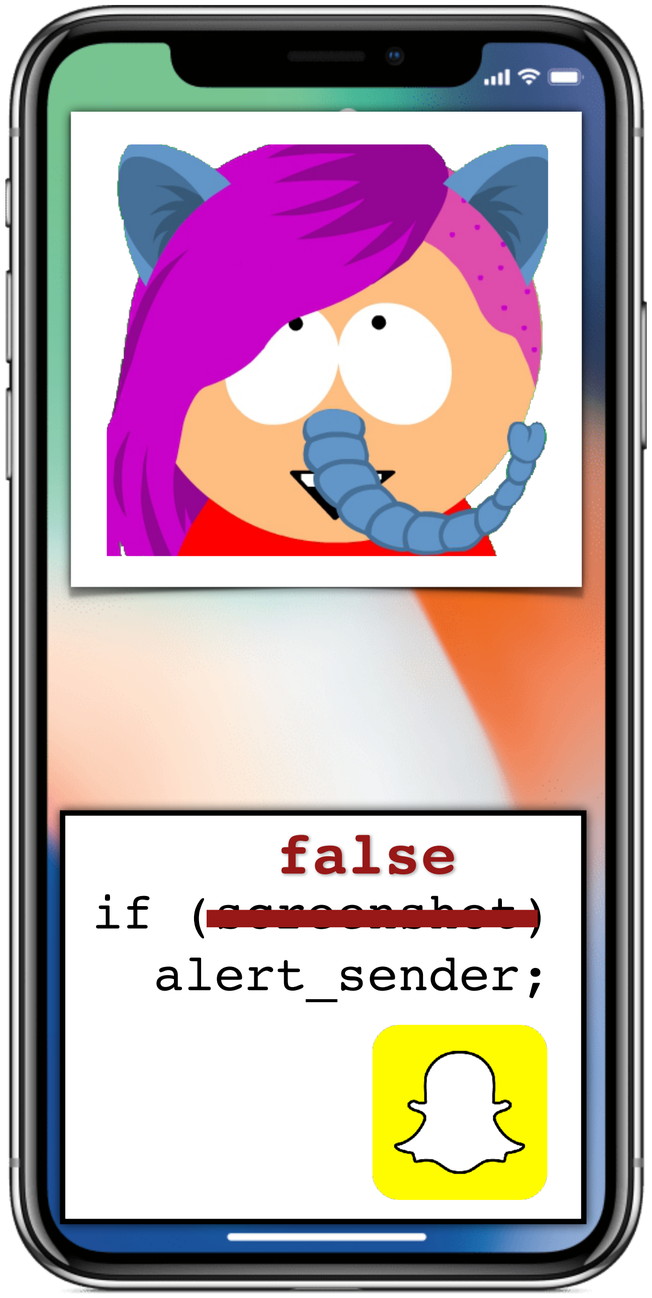
false

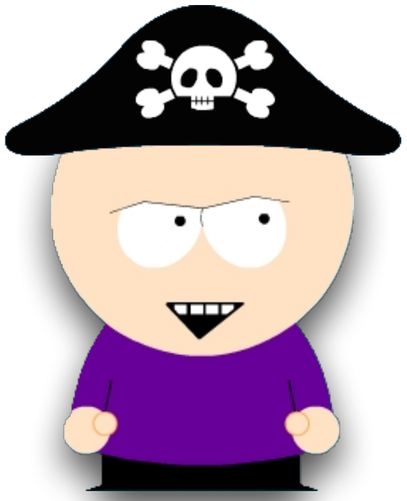


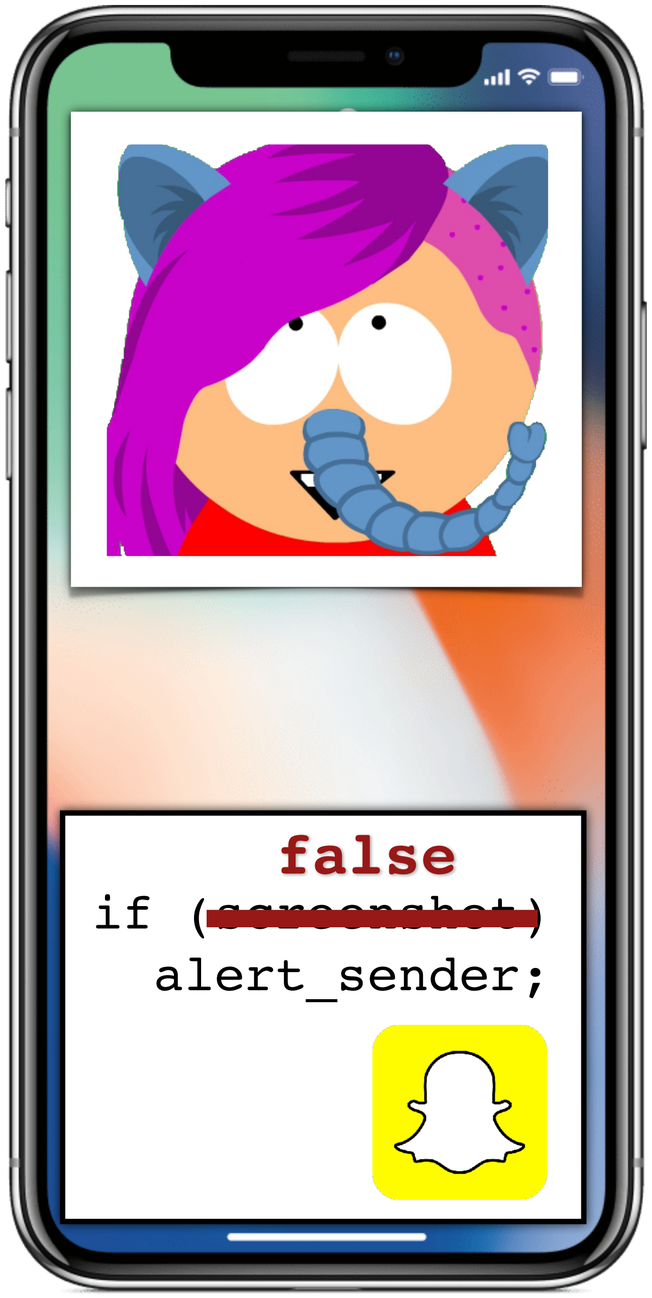



snapchat
Bob took a screenshot!




false
if (screenshot)
alert_sender;



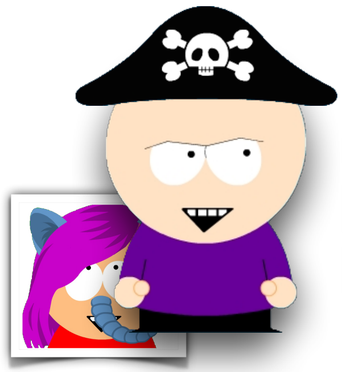


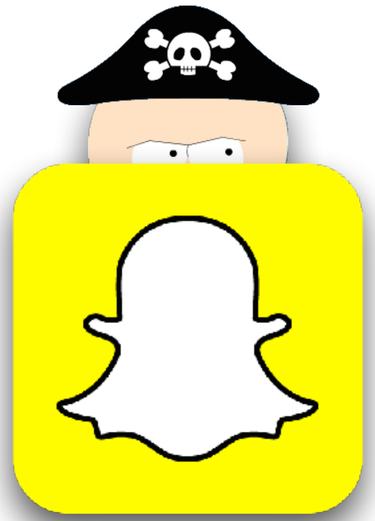
Integrity Violation

```
void snapchat() {
 if (screenshot)
  alert_sender;
}
```

Build

Package



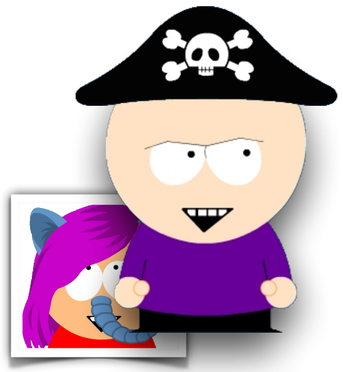


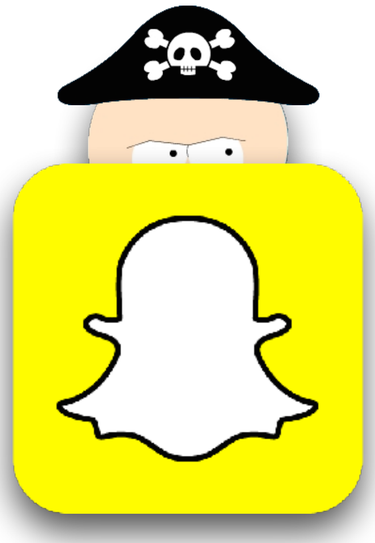
```
void snapchat() {
  if (screenshot)
    alert_sender;
}
```

Apache LOG4J CVE-2021-44228 Scanner™

Build

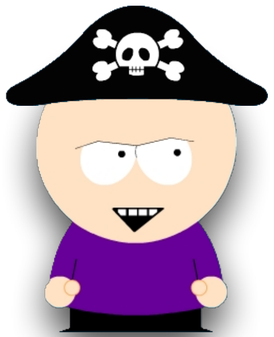
Package



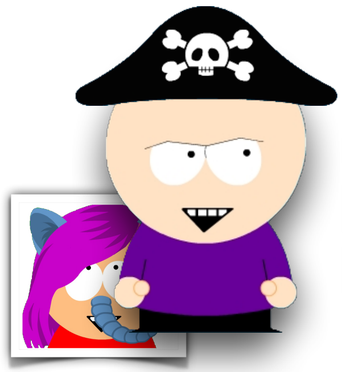


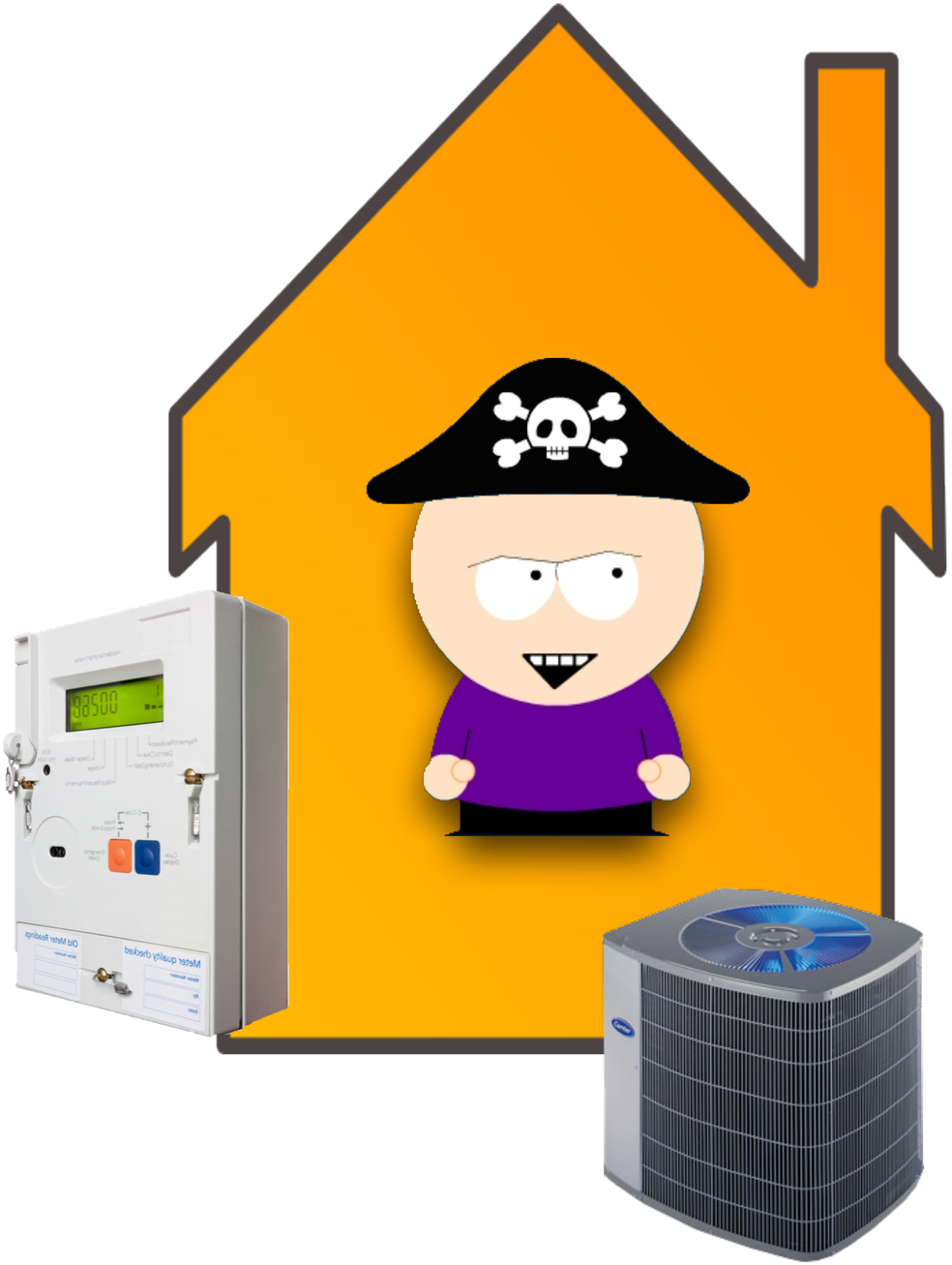
```
void snapchat() {
  if (screenshot)
    alert_sender;
}
```

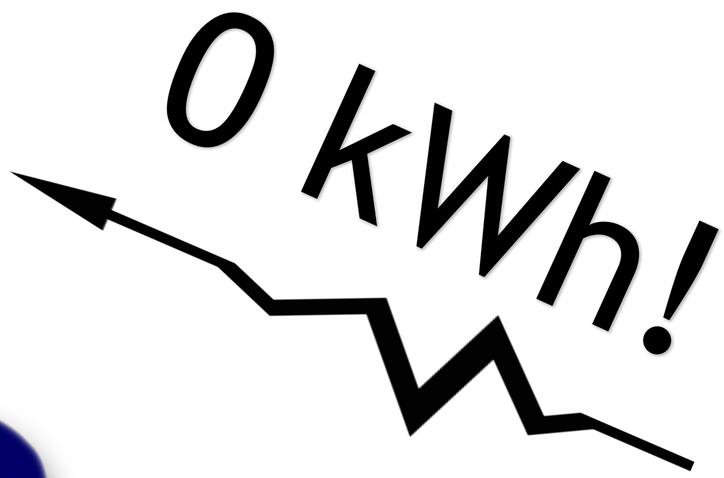
Apache LOG4J CVE-2021-44228 Scanner

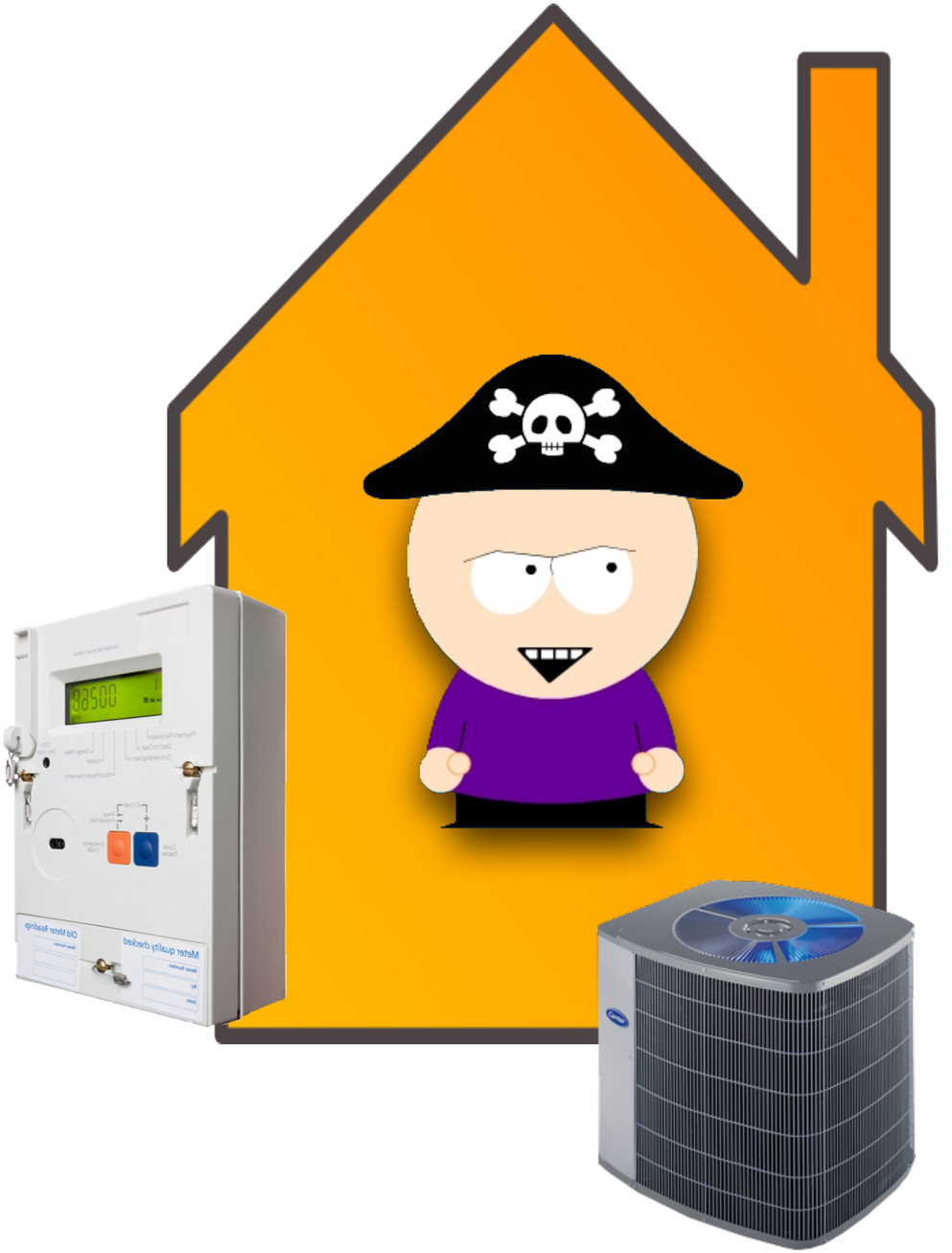


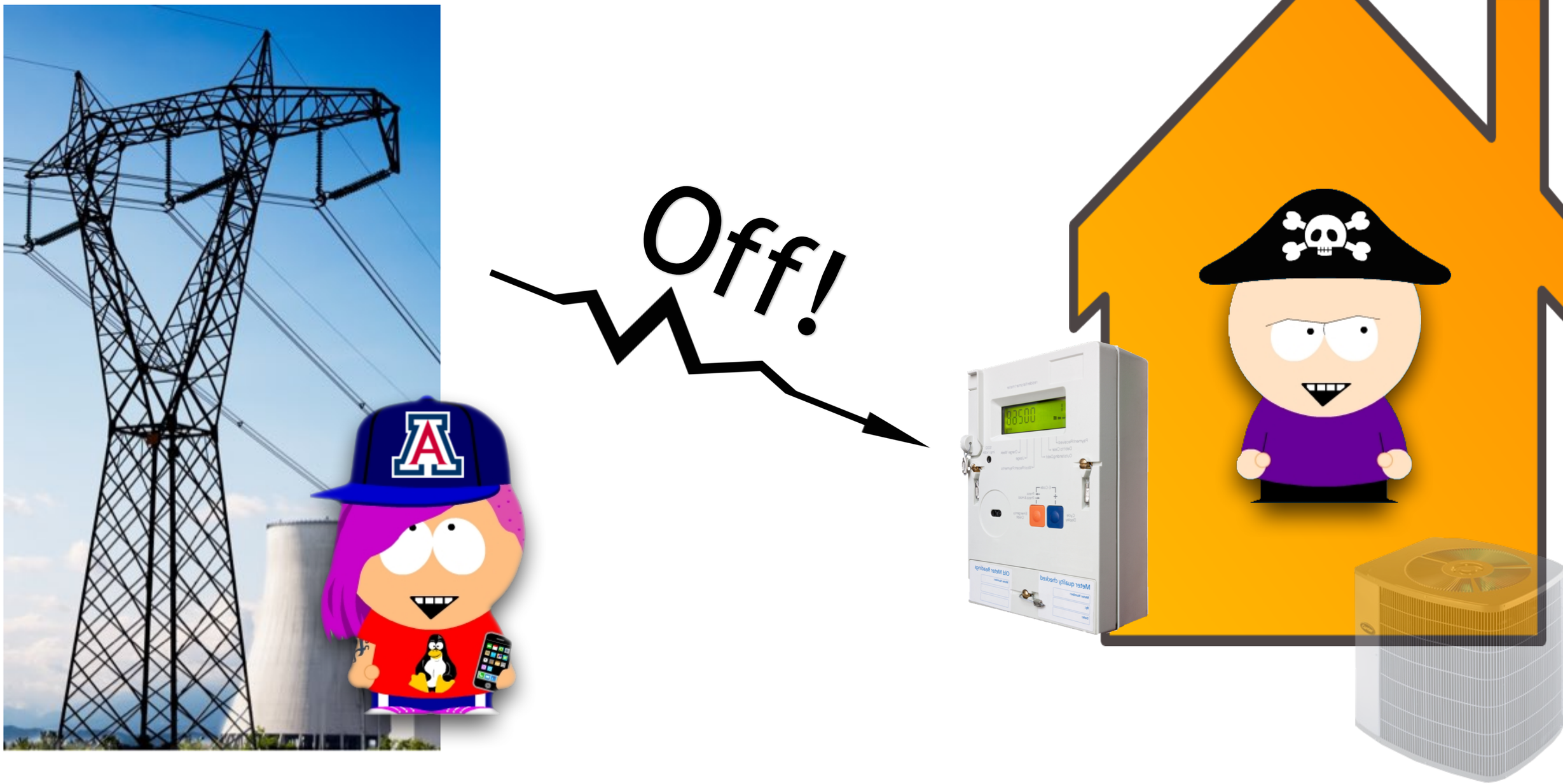
Package



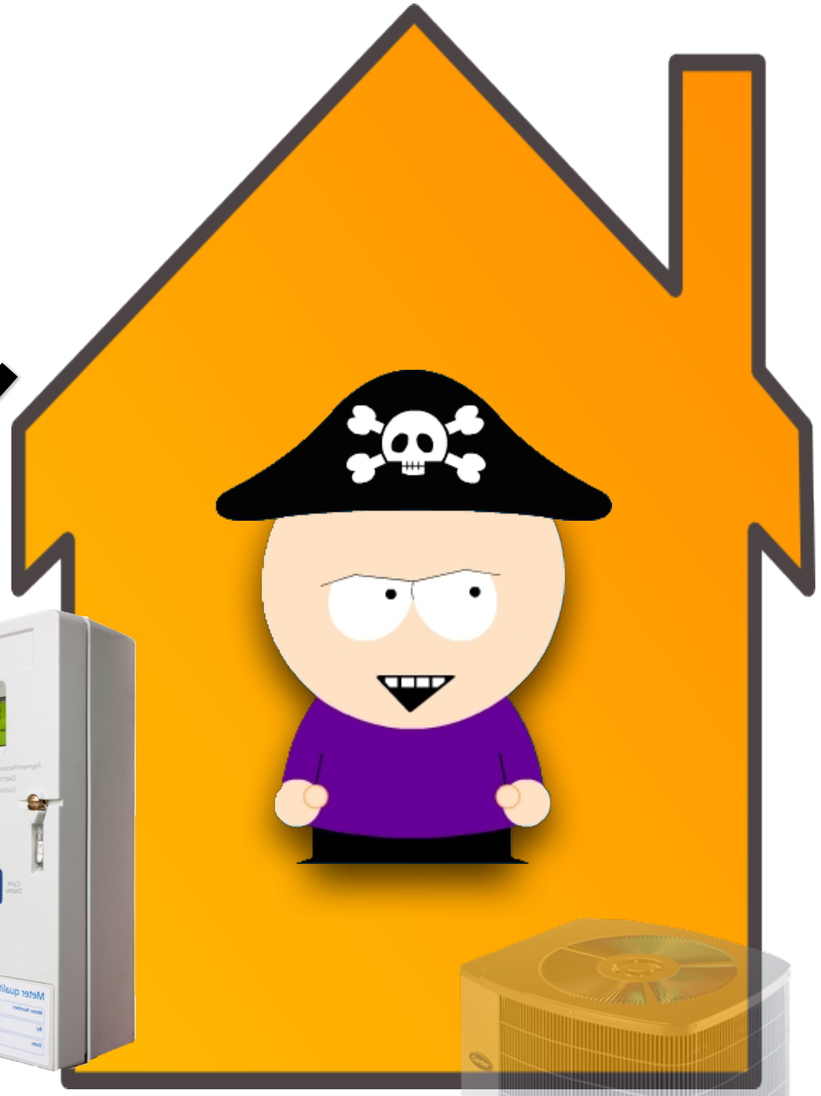



0 kWh!



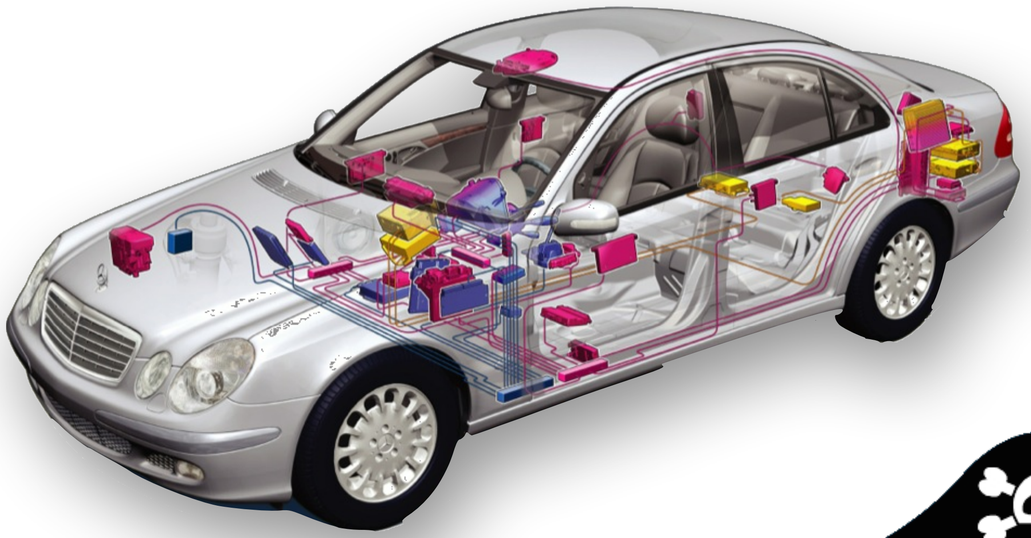

Off!

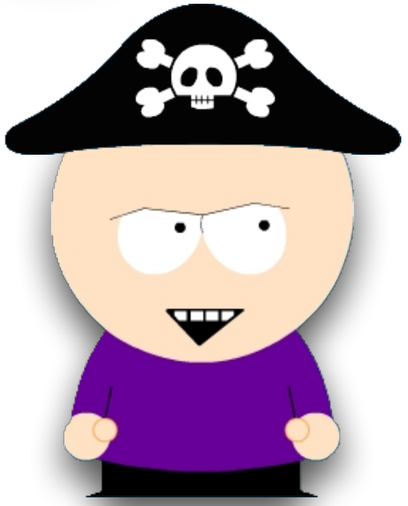
Off!

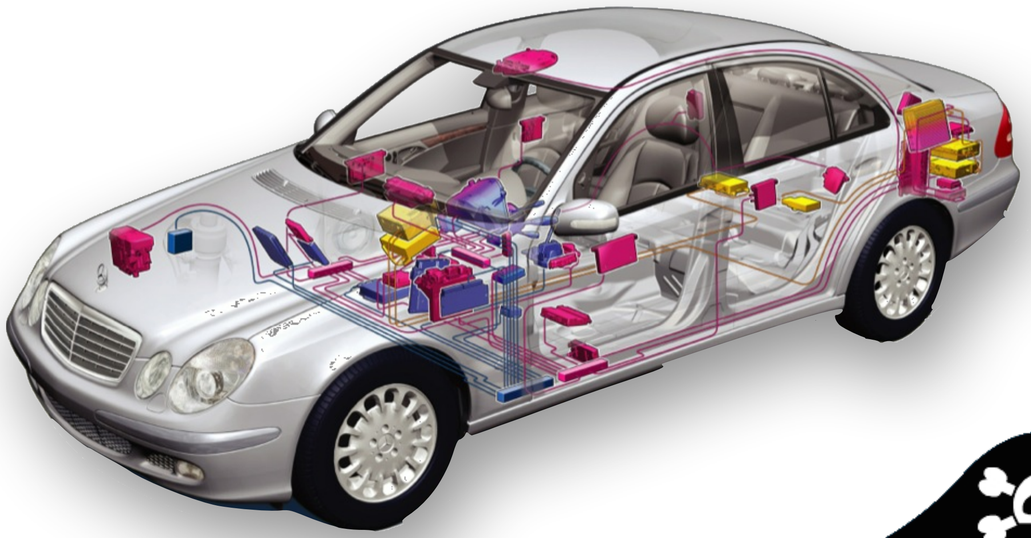


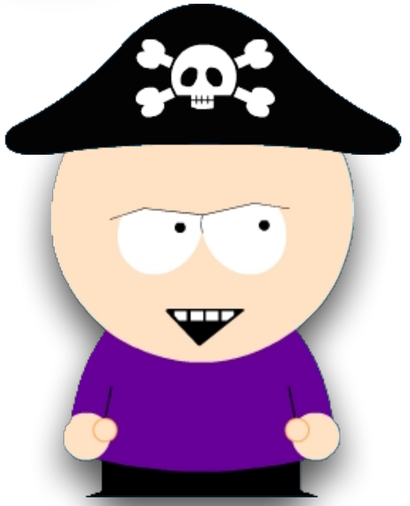
Off!

Integrity Violation

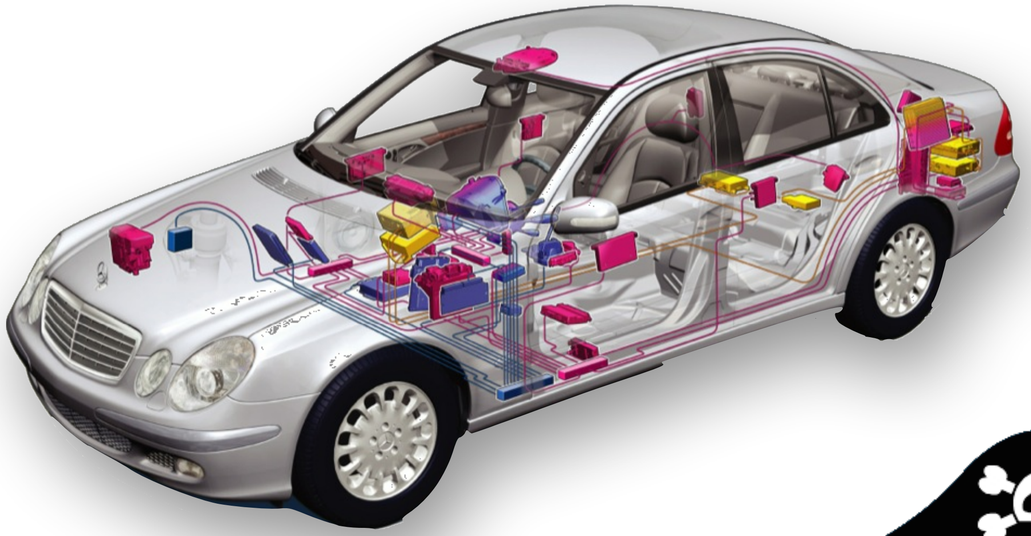


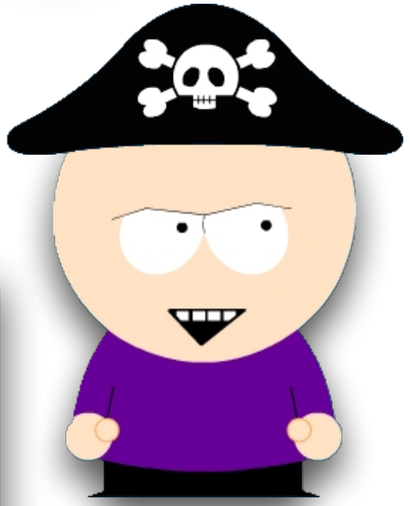






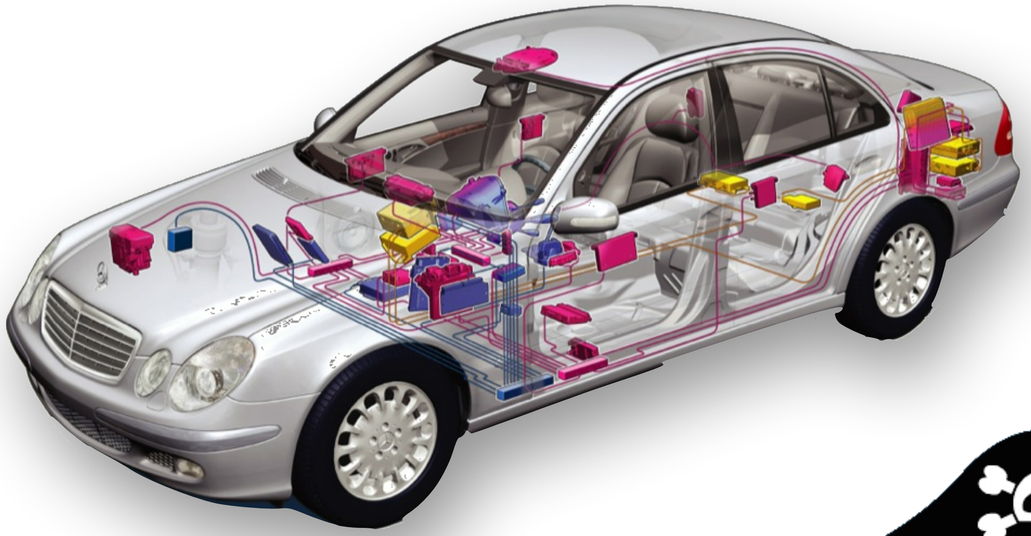
5000U

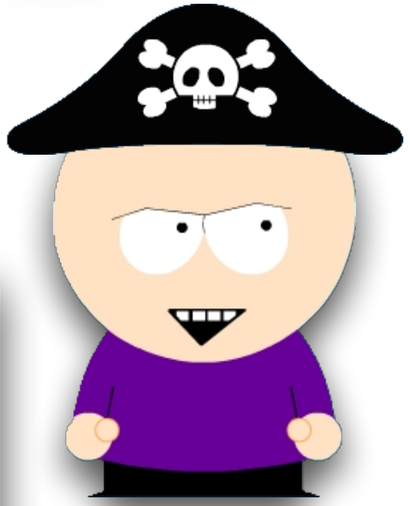




true

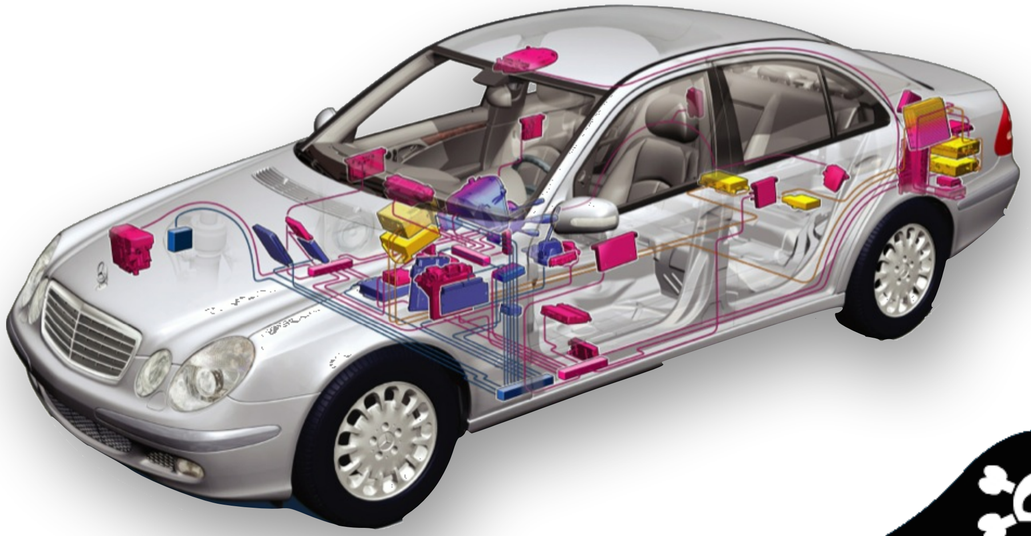
```
if (has_paid_for_upgrade())
    allow_ridiculous_mode = 1;
```

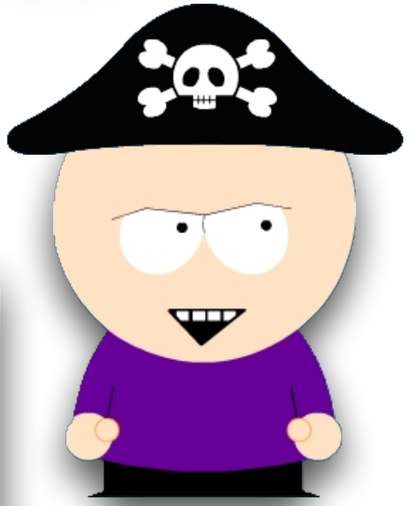




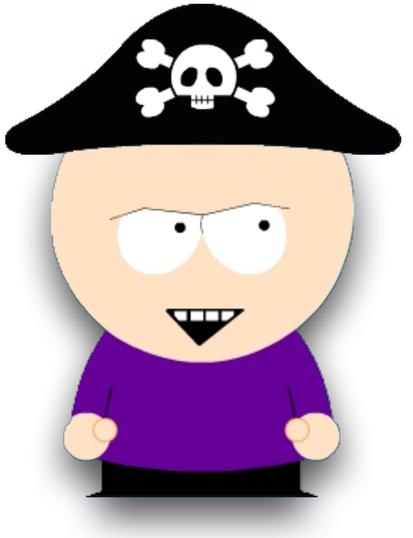
false

```
if (reported_stolen())
    send_GPS_coords();
```

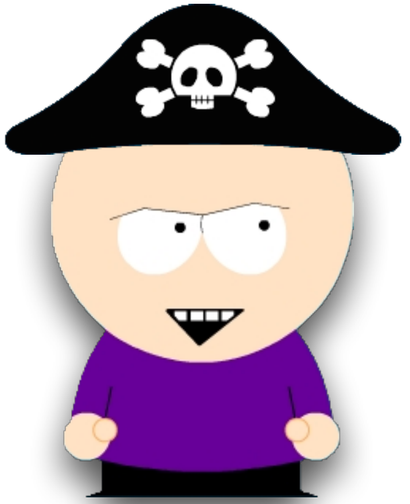


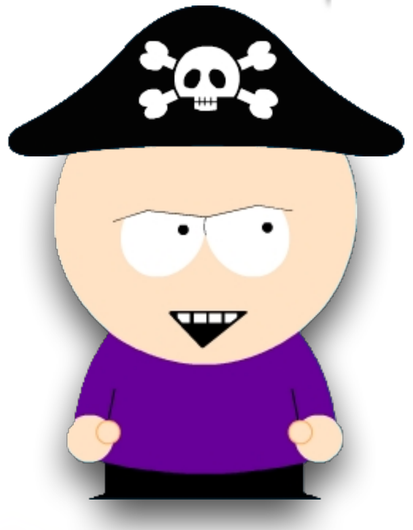


```
battery_management() {
    valuable trade secrets
}
```

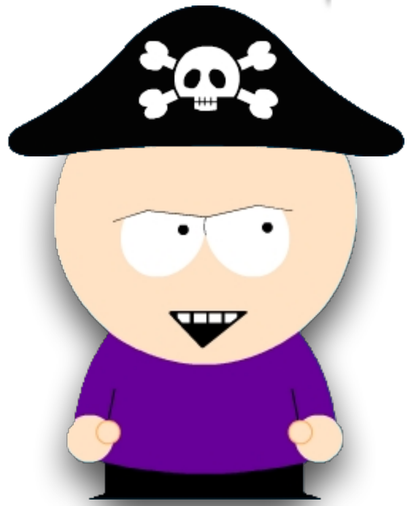


Confidentiality and Integrity Violation

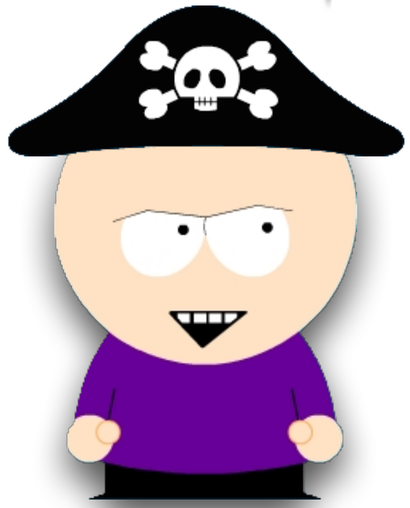





MENU
OK
LAST
MUTE
DELETE
ENTER
ADB

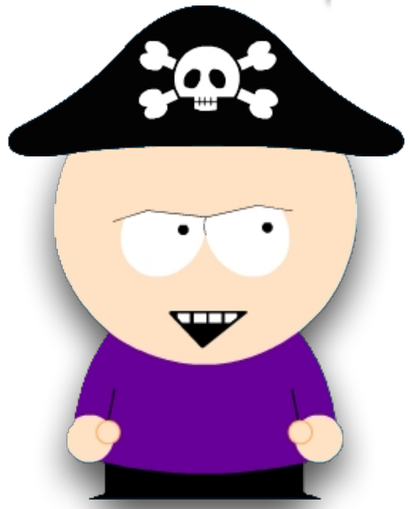



MENU
OK
GUIDE
LAST
MUTE
DELETE
ENTER
ADB

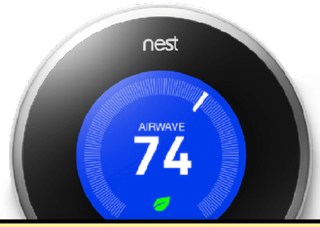



MENU
OK
LAST
MUTE
DELETE
ENTER
ADB


nest
AIRWAVE
74




MENU
OK
LAST
MUTE
DELETE
ENTER
ADB



Confidentiality and Integrity Violation

End User Defenses

End User Defenses

Protect Against End User Attack

Developer

A **Software Protection Tool** transforms a program into one that is harder to analyze, tamper with, and reverse engineer

Source-to-Source Transformer

https://tigress.wtf

the tigress c obfuscator

linux/darwin/android, intel/arm/webassembly, 32/64, gcc/clang/emcc

Download

Source-to-Source Transformer

foo.c

```
main(){
}
```

Source-to-Source Transformer

foo.c

```
main(){
}
```

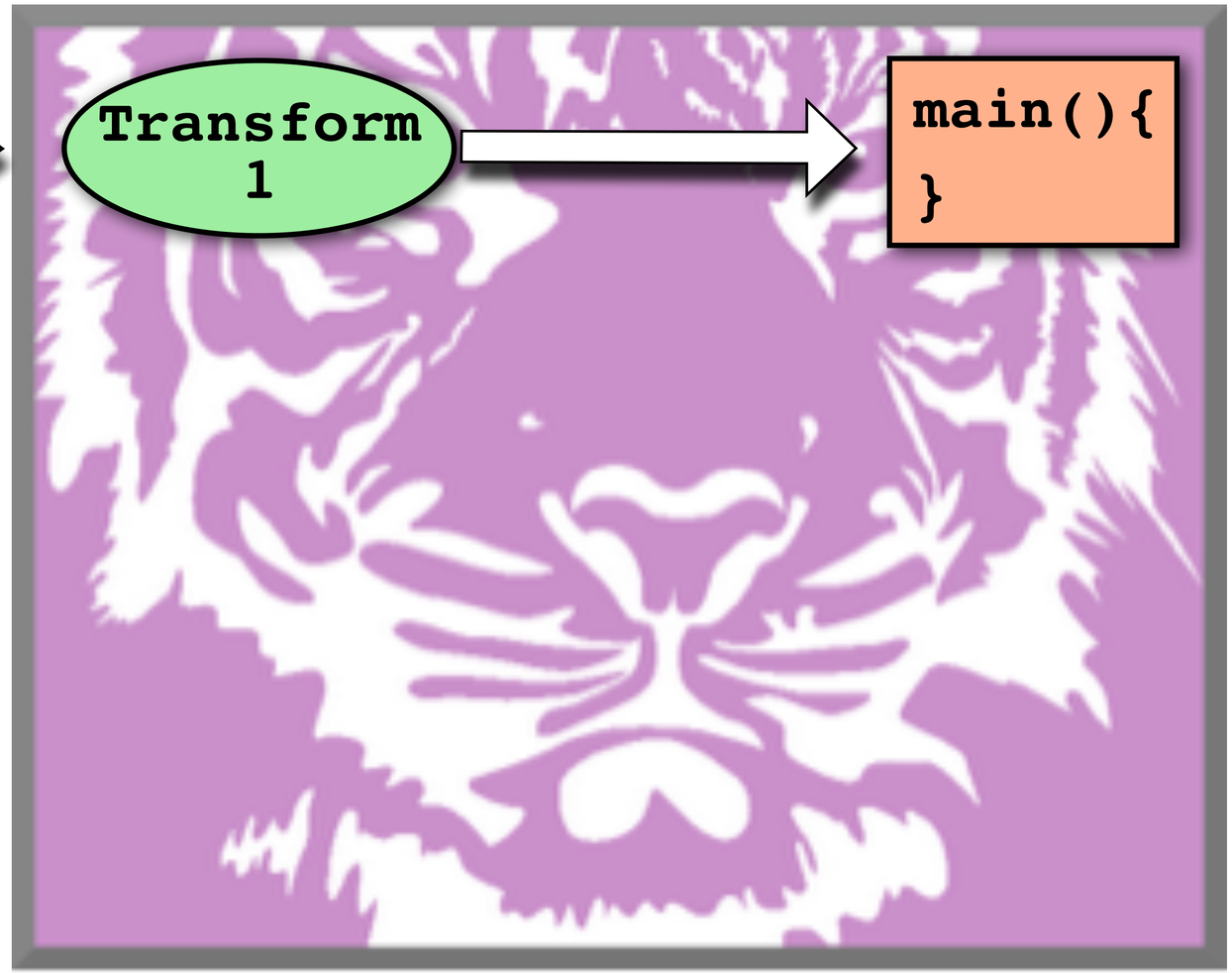
foo’.c

```
main(){
}
```

Source-to-Source Transformer

foo.c

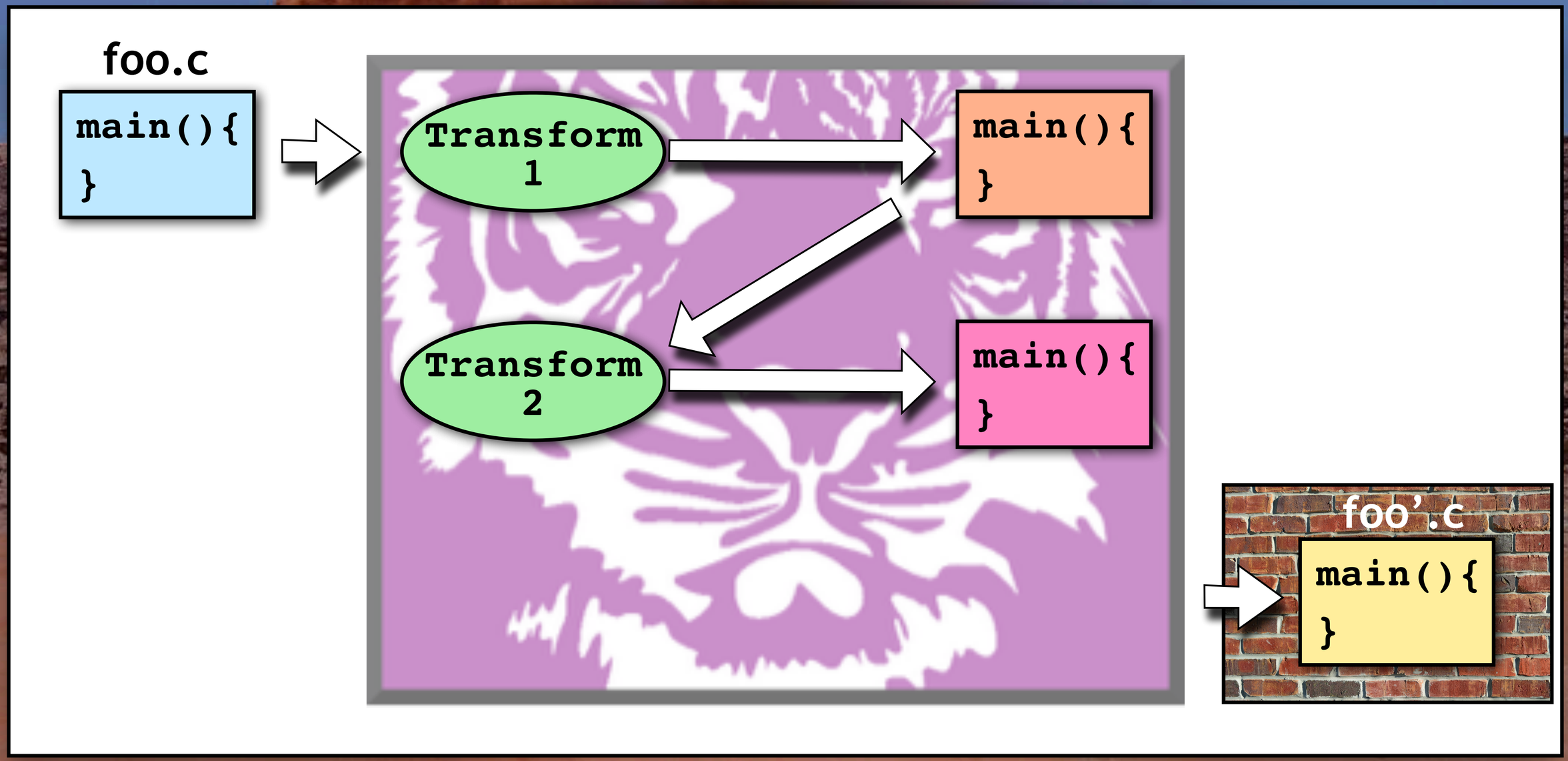
```
main(){
}
```



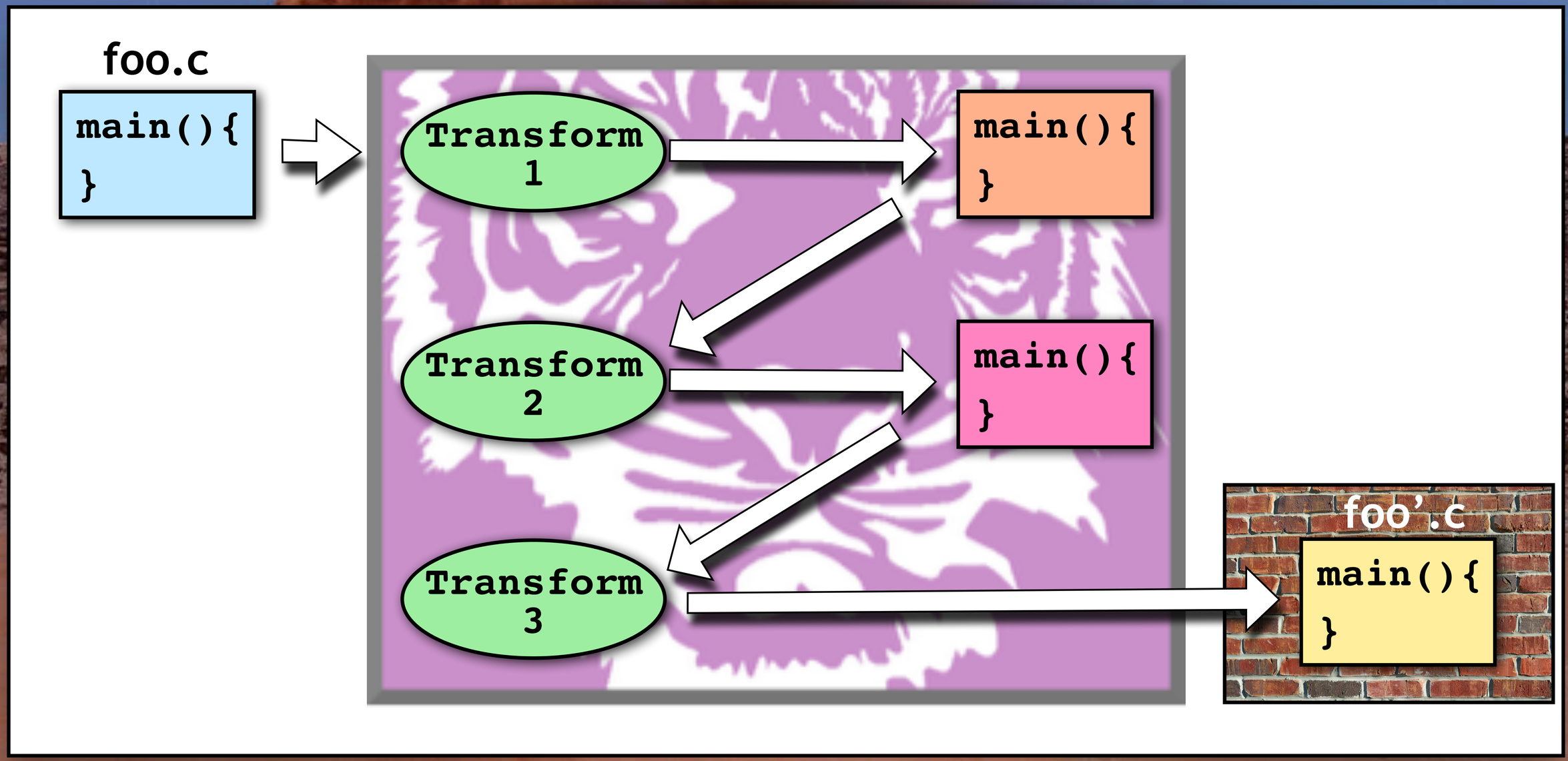
foo'.c

```
main(){
}
```

Source-to-Source Transformer



Source-to-Source Transformer



Source-to-Source Transformer

- Code obfuscation
- Tamperproofing
- Watermarking and
- Diversification

A **code obfuscation** tool transforms a program into one that is

- harder for a human to read and understand and
- harder for a program to automatically analyze.

Protects against **confidentiality** violations.

Code Obfuscation

```
int foobar(int x, int z, int s){
   char* next=&&cell0;
   int retVal = 0;
   cell0: {next=(s==1)?&&cell1:&&cell2;
           goto *next;}
   cell1: {retVal=(x*37)%51; goto end;}
   cell2: {next=(s==2)?&&cell3:&&end;
           goto *next;}
   cell3: {next=(x==z)?&&cell4:&&end;
           goto *next;}
   cell4: {
      int x2=x*x%51,x3=x2*x%51;
      int x4=x2*x2%51,x8=x4*x4%51;
      int x11=x8*x3 % 51;
      printf("%i \n",x11); goto end;
   }
   end: return retVal;
}
int main() {
   int y = 6;
   y = foobar(y,99,1);
   foobar(y,42,2);
}
```

Code Obfuscation

```
int foo(int x) {
    return x*7;
}
void bar(int x,int z){
    if (x==z)
      printf(“%i\n",x);
}
int main() {
    int y = 6;
    y = foo(y);
    bar(y,42);
}
```

```
int foobar(int x, int z, int s){
    char* next=&&cell0;
    int retVal = 0;
    cell0: {next=(s==1)?&&cell1:&&cell2;
            goto *next;}
    cell1: {retVal=(x*37)%51; goto end;}
    cell2: {next=(s==2)?&&cell3:&&end;
            goto *next;}
    cell3: {next=(x==z)?&&cell4:&&end;
            goto *next;}
    cell4: {
        int x2=x*x%51,x3=x2*x%51;
        int x4=x2*x2%51,x8=x4*x4%51;
        int x11=x8*x3 % 51;
        printf("%i \n",x11); goto end;
    }
    end: return retVal;
}
int main() {
    int y = 6;
    y = foobar(y,99,1);
    foobar(y,42,2);
}
```

Code Tamperproofing

A **code tamperproofing** tool transforms a program into one that

1. **detects** if the program has been modified, and
2. **reacts**.

Protects against **integrity** violations.

Code Tamperproofing

```
int foo() {
    … … …;
}
int main() {
    foo();
}
```

Code Tamperproofing

```
int foo() {
    … … …;
}
int main() {
    foo();
}
```

```
int foo() {
    … … …;
}
int main() {
    if (foo has changed) {
        repair foo;
    }
    foo();
}
```

Code Tamperproofing

```
int foo() {
    … … …;
}
int main() {
    foo();
}
```

```
int foo() {
    … … …;
}
int main() {
    if (foo has changed) {
        repair foo;
    }
    foo();
}
```

Code Watermarking

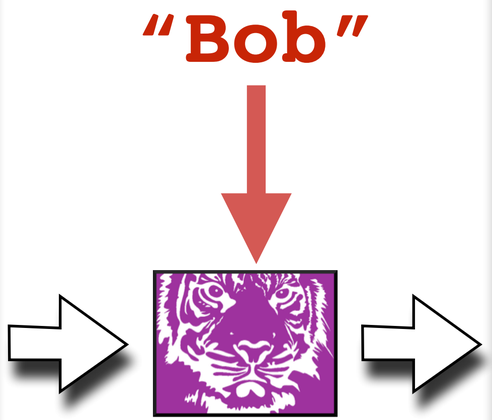
A **code watermarking** tool transforms a program into one that

- **embeds** a unique identifier
- that is difficult to remove

Protects against **availability** violations.

Code Watermarking

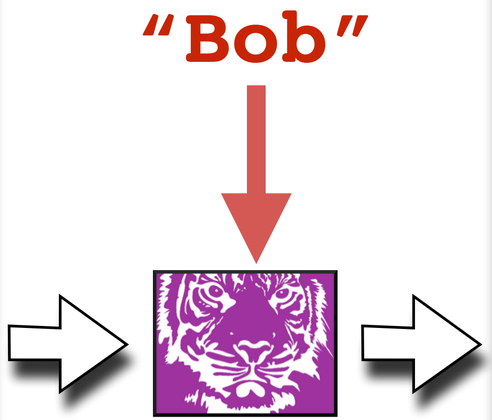
```
int main() {
    …
}
```




```
int main() {




}
```

Code Watermarking

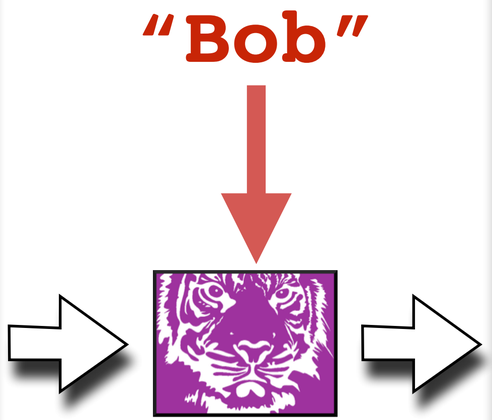
```
int main() {
    …
}
```




```
string wm = "Bob";

int main() {




}
```

Code Watermarking

```
int main() {
    …
}
```




```
string wm = “Bob”;

int main() {
    char wm;
    wm = ‘B’;
    wm = ‘o’;
    wm = ‘b’;
}
```

Code Diversification

A **code diversification** tool transforms a program into multiple programs where

- each program is **unique and**
- has the **same semantics**

Protects against **class-attacks**.

Code Diversification

```
int main() {
    x = x + x;
}
```

```
int main() {
    x = x*2;
}
```

Code Diversification

```
int main() {
    x = x*2;
}
```

```
int main() {
    x = x + x;
}
```

```
int main() {
    x = x << 1;
}
```

Code Diversification

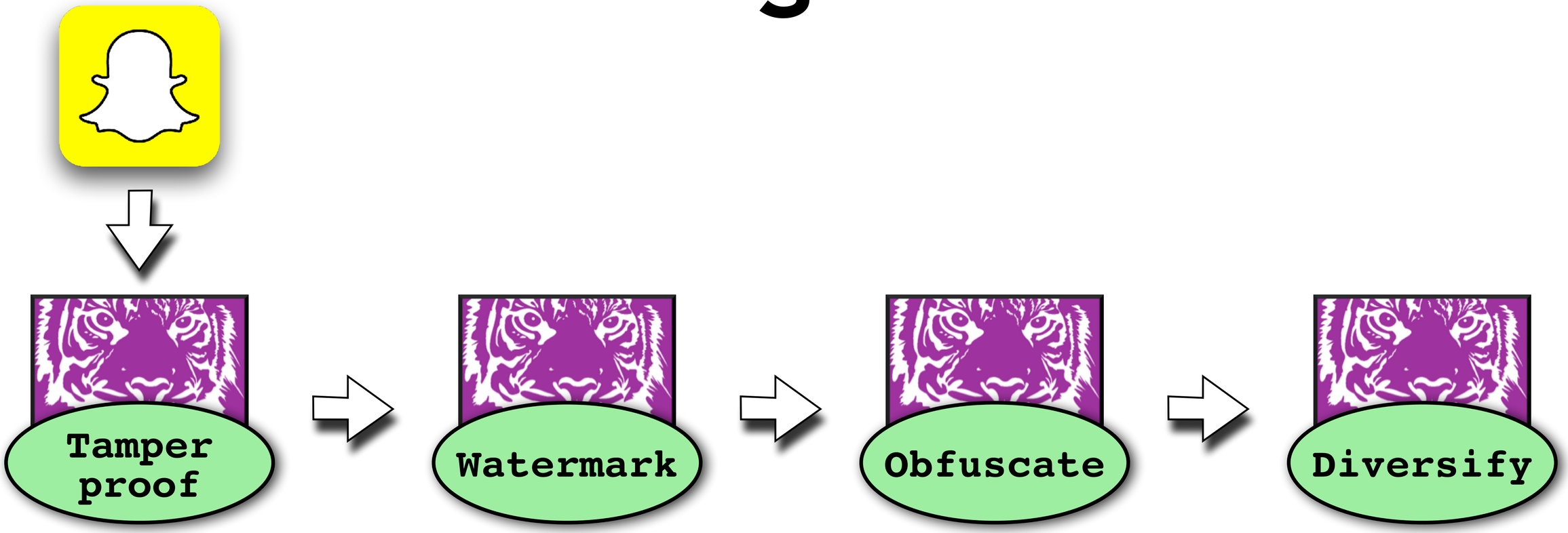
```
int main() {
    x = x*2;
}
```

```
int main() {
    x = x + x;
}
```

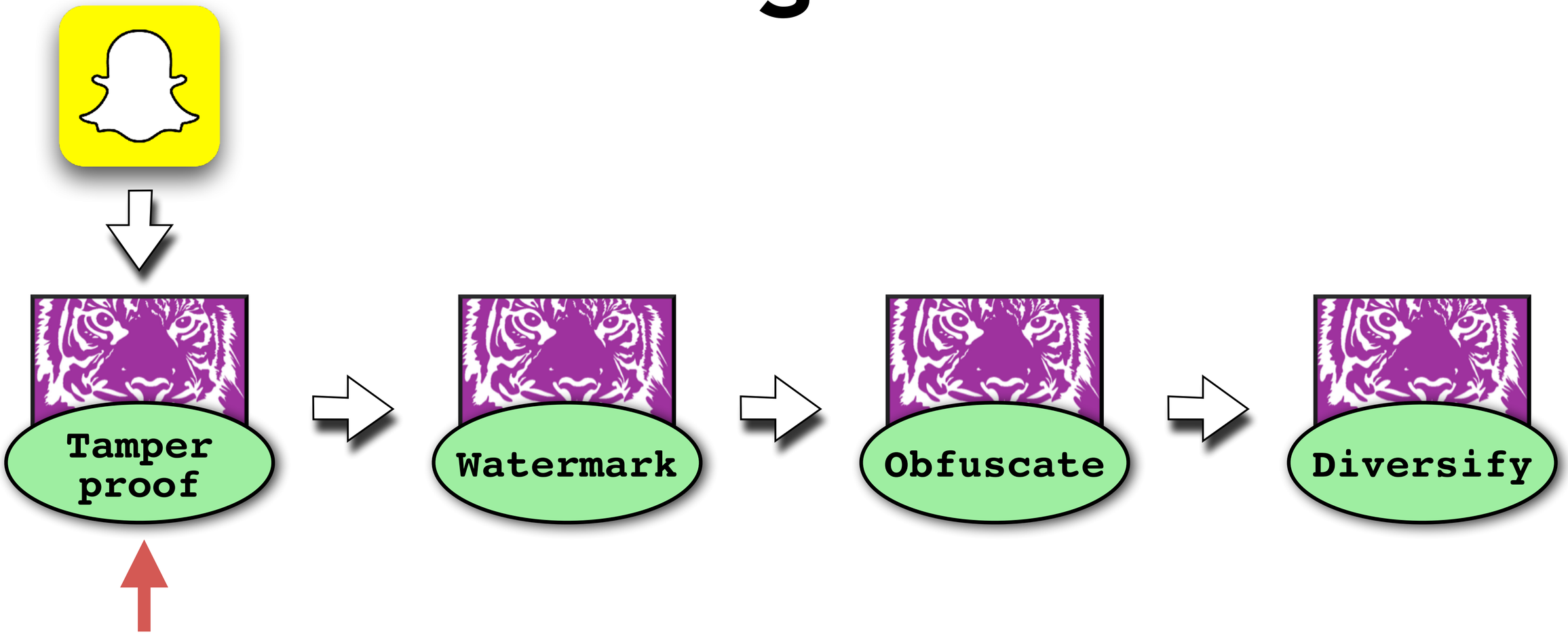
```
int main() {
    x = x << 1;
}
```

```
int main() {
  for(i=0;i++;i<x)
      x++;
}
```

Combining Protections

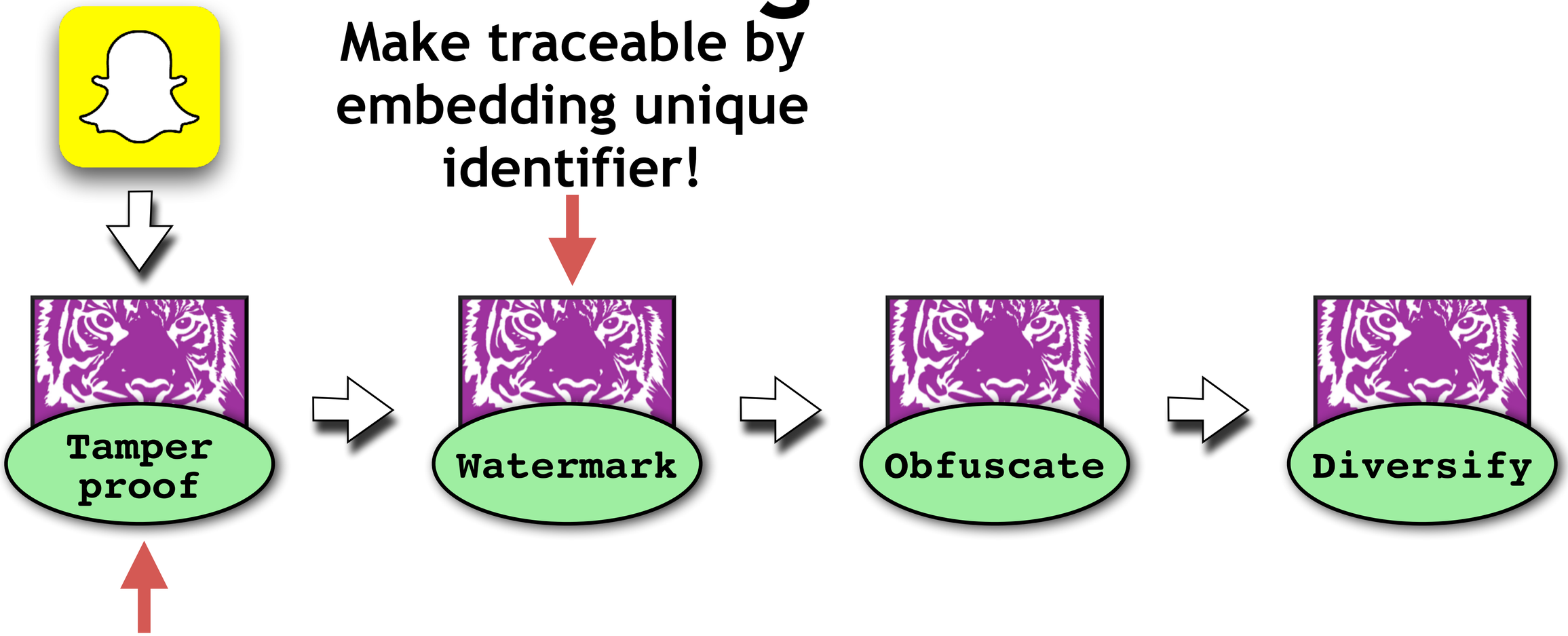


Combining Protections

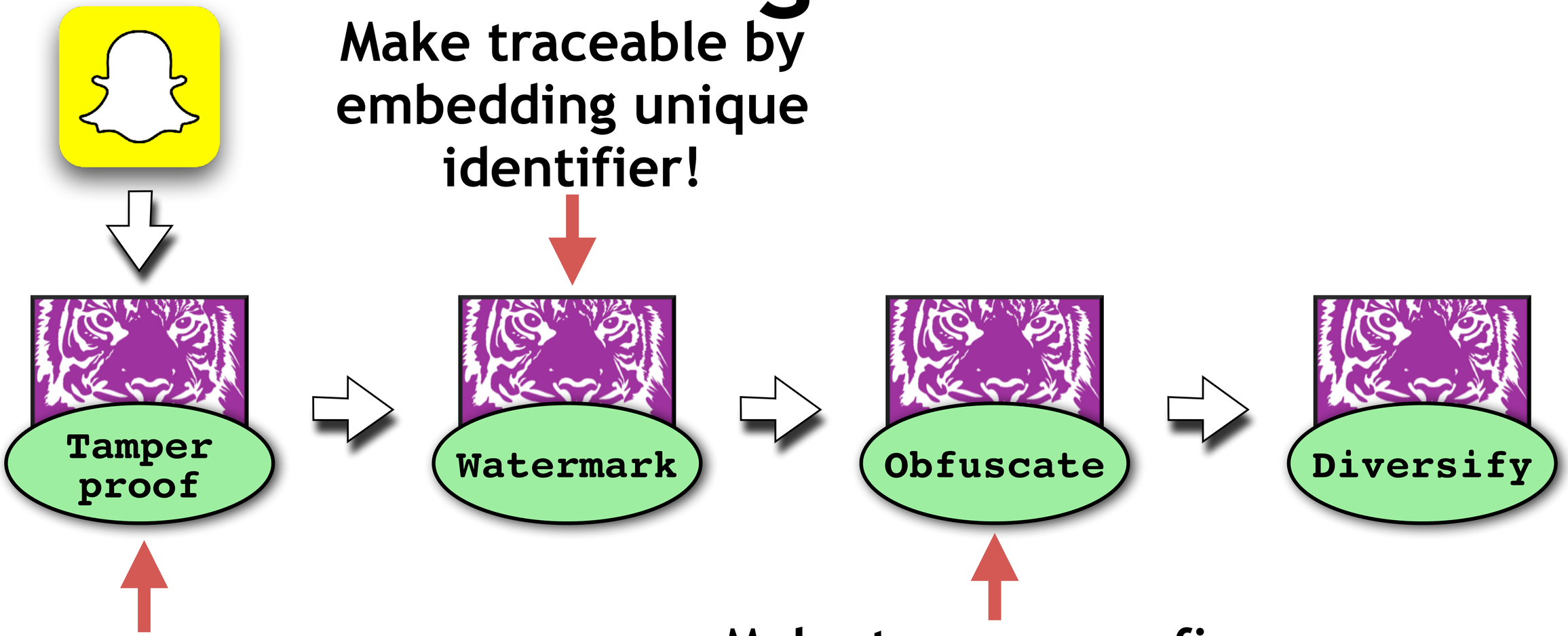


Make hard to modify!

Combining Protections



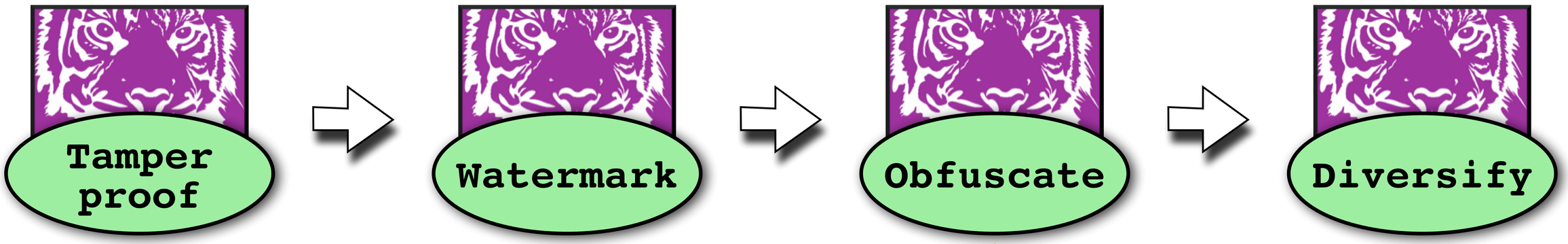
Combining Protections



Combining Protections

Make traceable by embedding unique identifier!

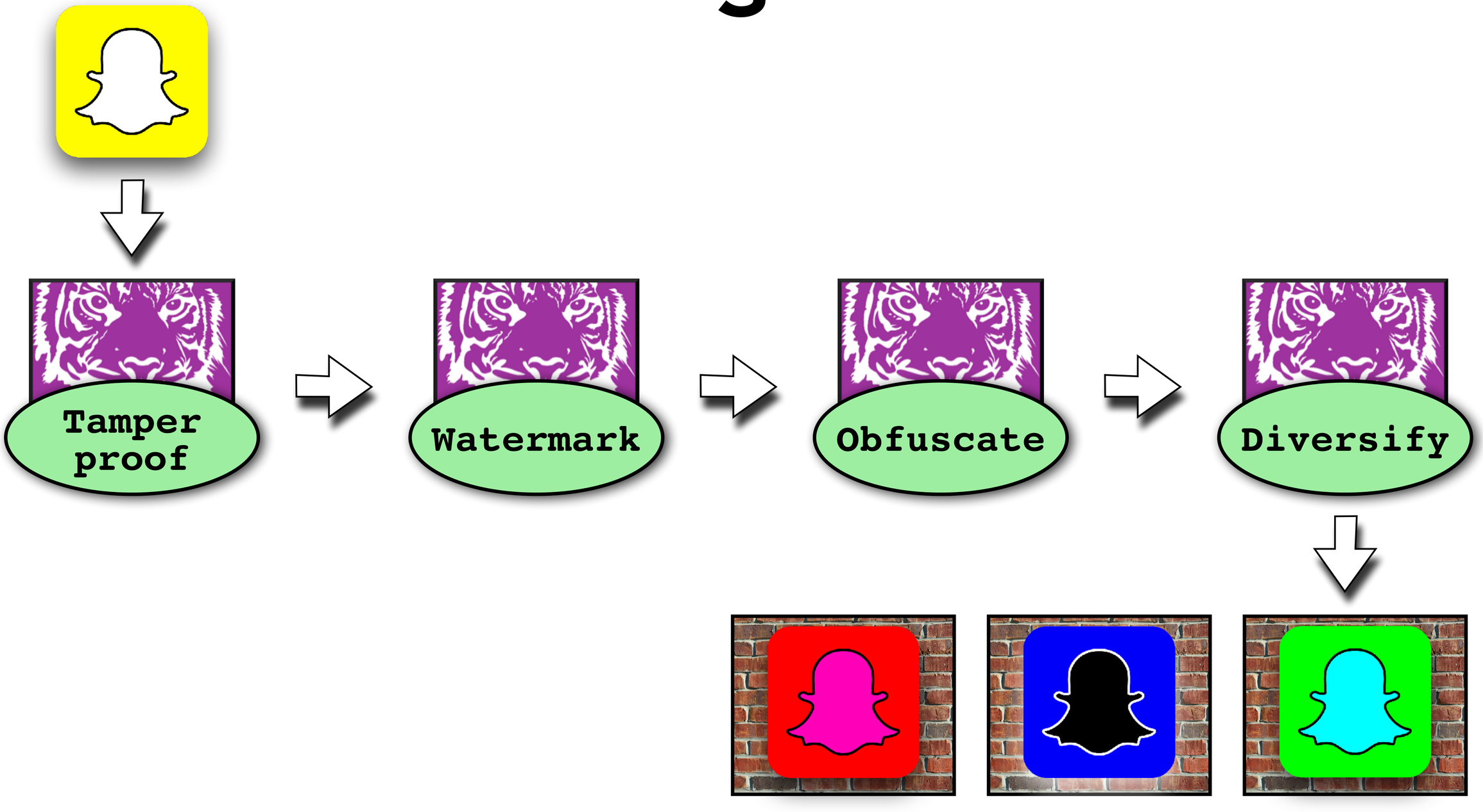
Avoid class attacks by giving every user a unique variant!



Make hard to modify!

Make tamperproofing and watermark hard to find!

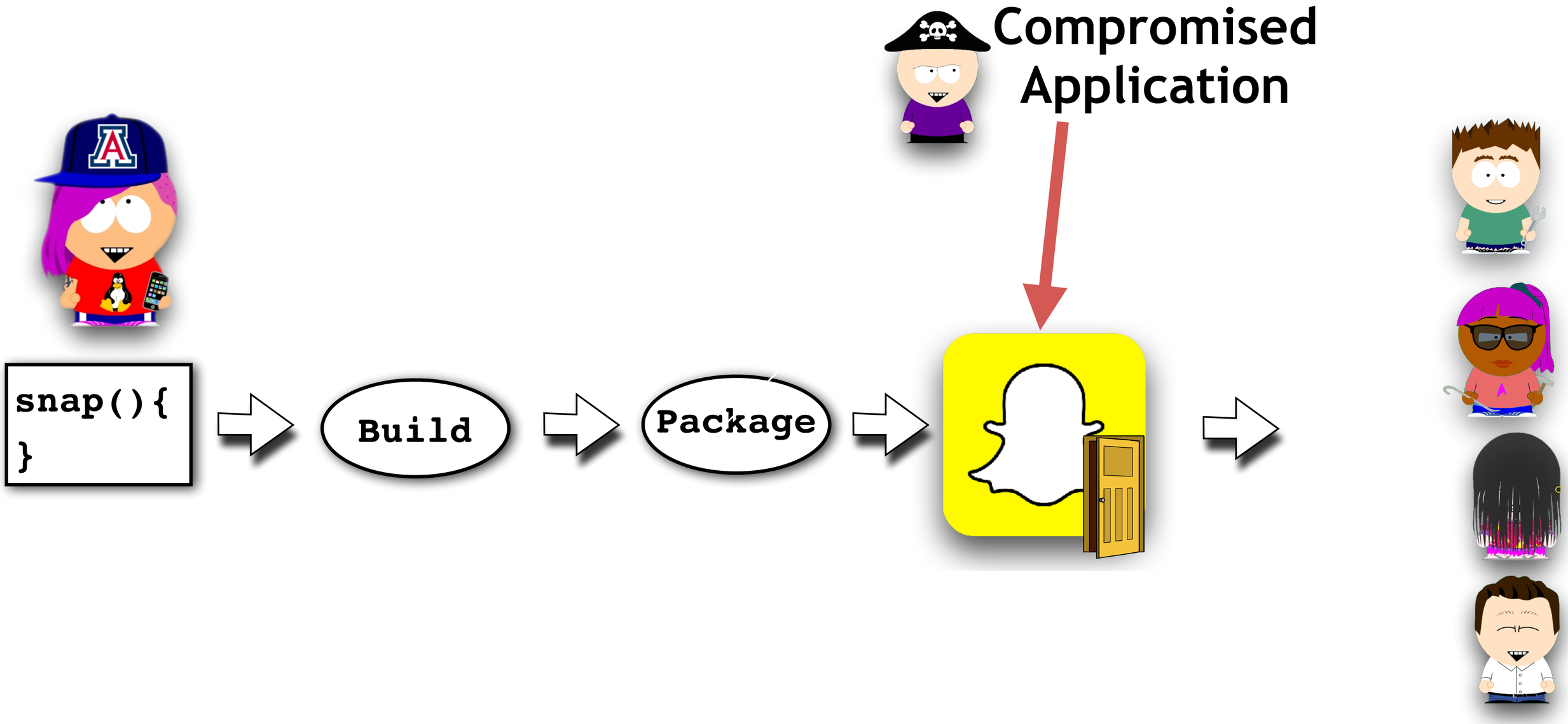
Combining Protections



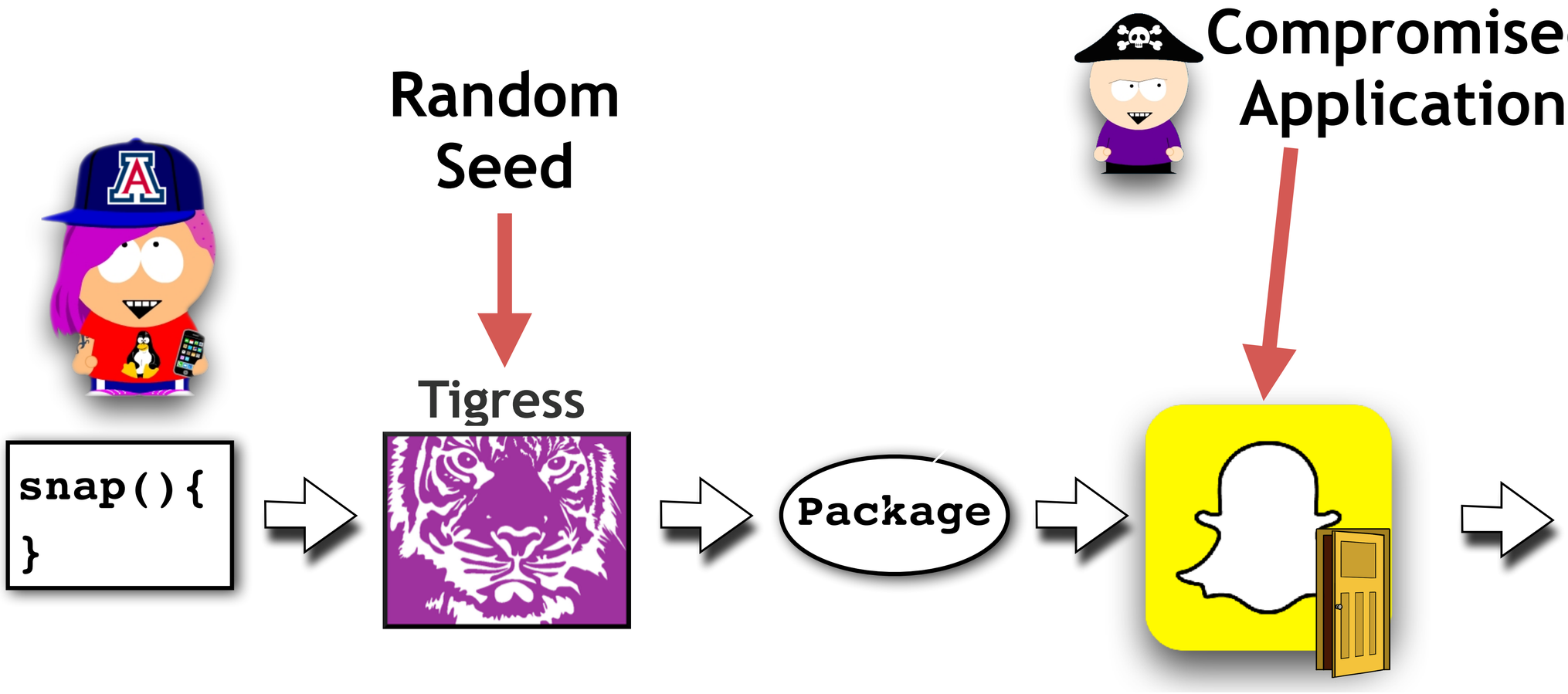
Diversification Algorithms

Diversification Algorithms

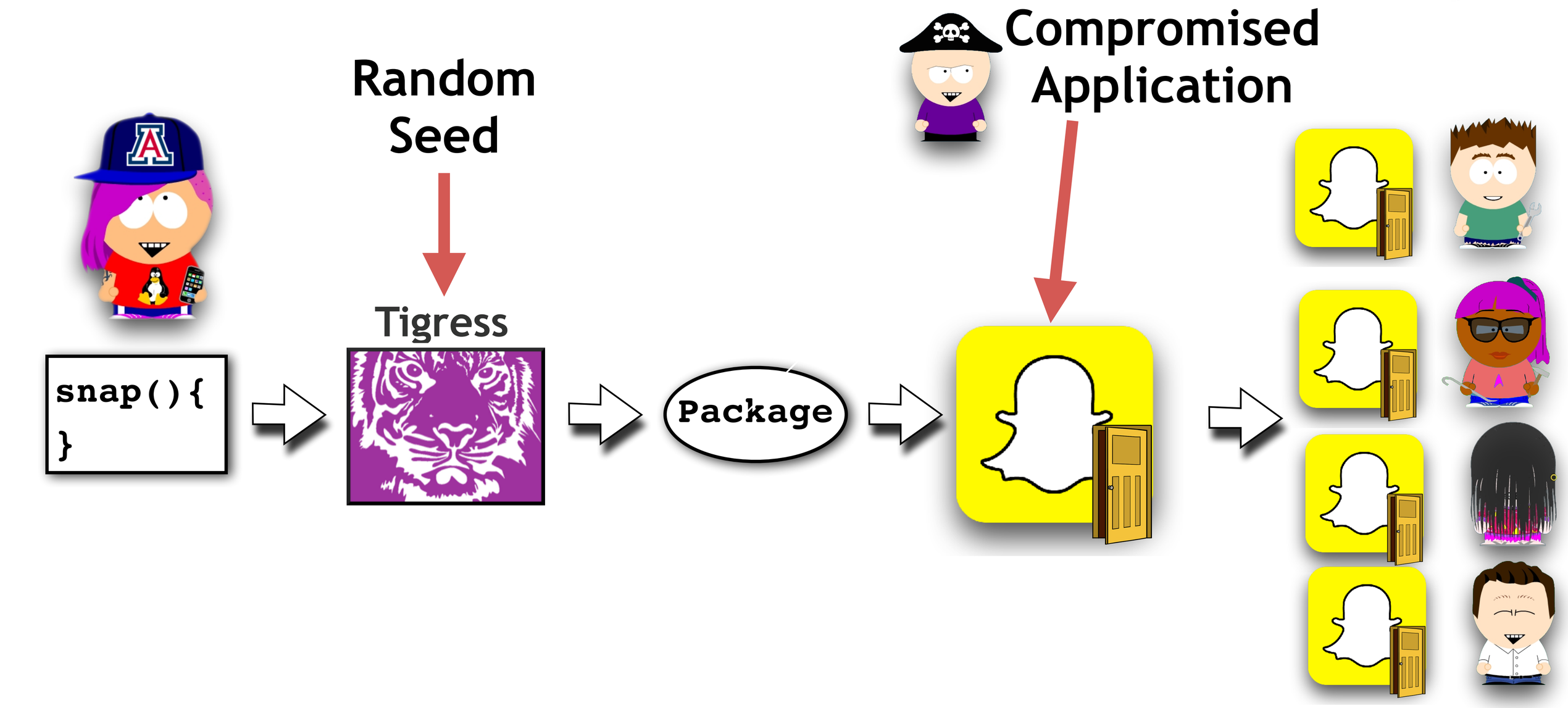
Automated Software Diversity



Automated Software Diversity



Automated Software Diversity



mated Software Diversity

BREAK ONCE
BREAK EVERYWHERE!

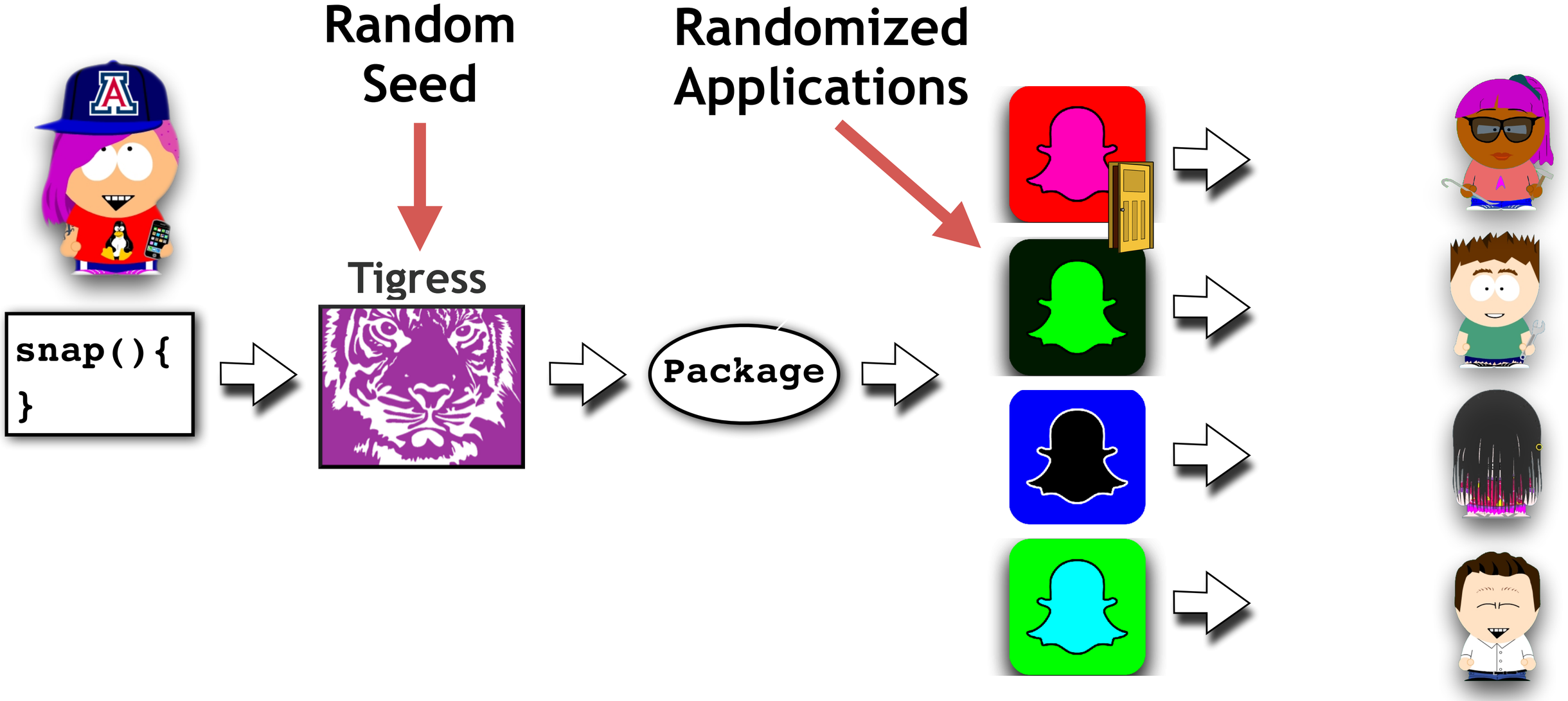
Compromised
Application

Tigress

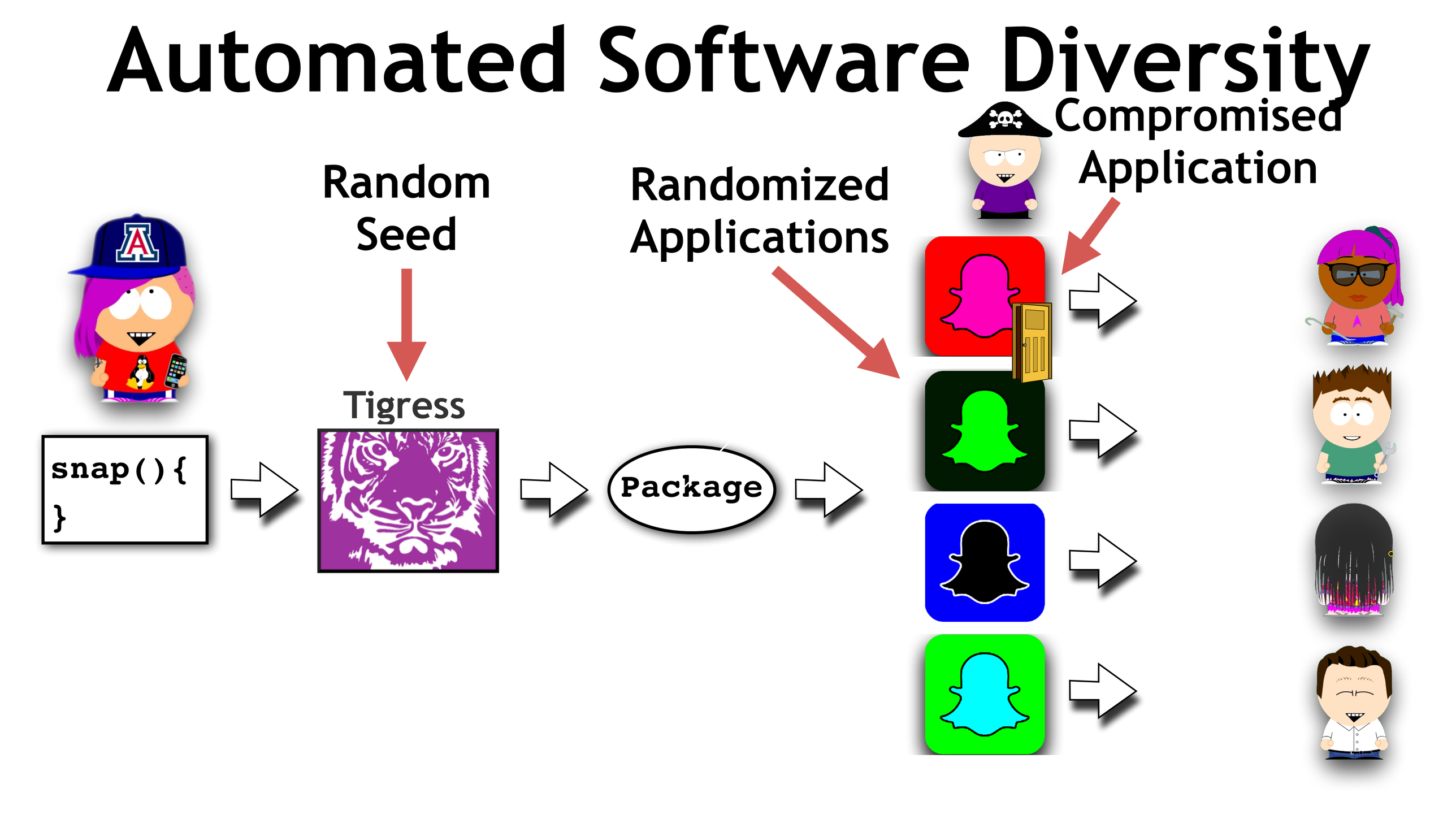
```
snap(){
}
```

Packag

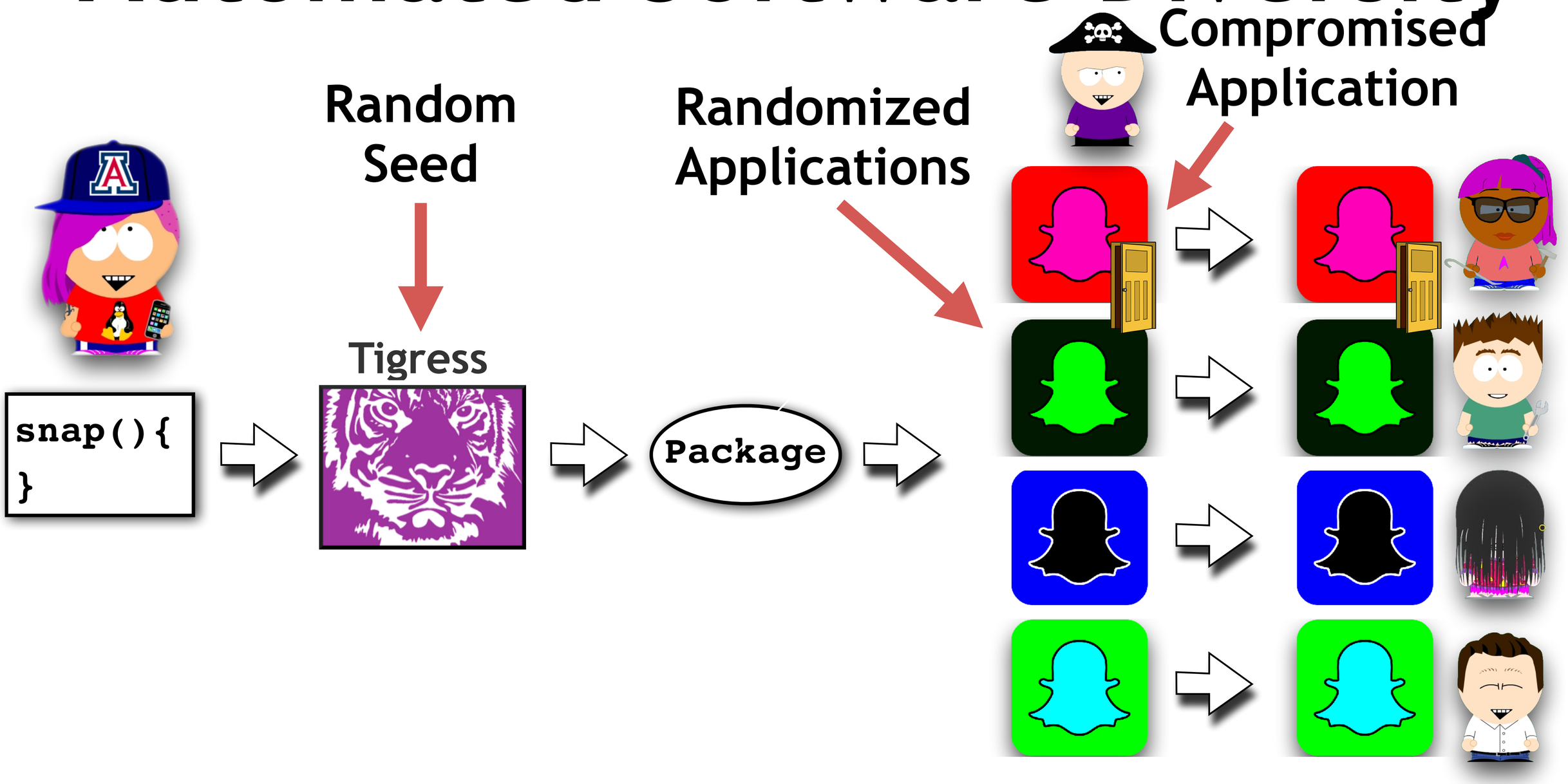
Automated Software Diversity



Automated Software Diversity



Automated Software Diversity



Diversifying Integer Expressions

$$x+y = \begin{cases} x + y \\ \\ \\ \end{cases}$$

Diversifying Integer Expressions

$$x+y = \begin{cases} x + y \\ x - \neg y - 1 \end{cases}$$

Diversifying Integer Expressions

$$x+y = \begin{cases} x + y \\ x-\neg y-1 \\ (x \oplus y)+2\cdot(x \wedge y) \end{cases}$$

Diversifying Integer Expressions

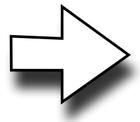
$$x+y = \begin{cases} x + y \\ x-\neg y-1 \\ (x \oplus y)+2\cdot(x \wedge y) \\ (x \vee y)+(x \wedge y) \end{cases}$$

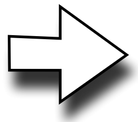
Diversifying Integer Expressions

$$x+y = \begin{cases} x + y \\ x - \neg y - 1 \\ (x \oplus y) + 2 \cdot (x \wedge y) \\ (x \vee y) + (x \wedge y) \\ 2 \cdot (x \vee y) - (x \oplus y) \end{cases}$$

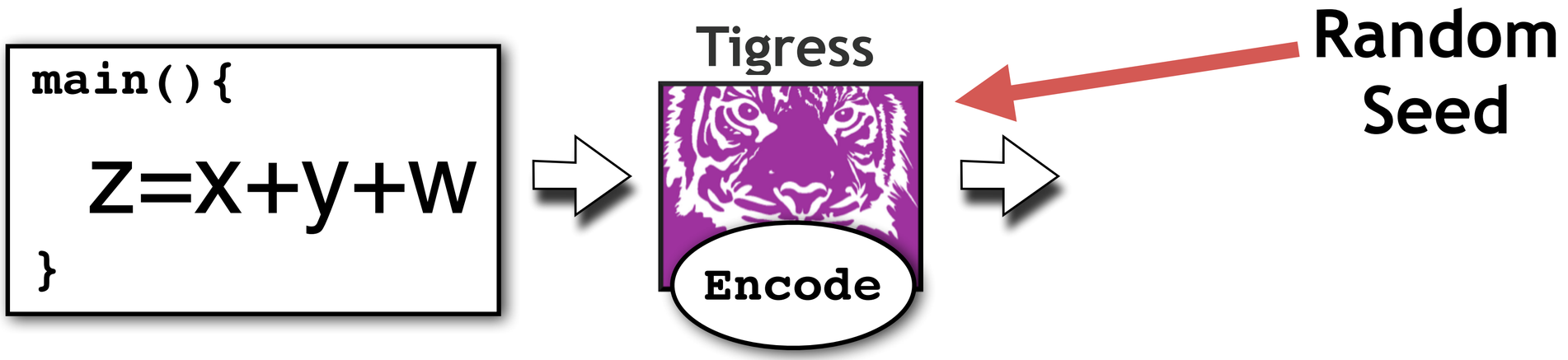
Diversifying Integer Expressions

```
main(){
  z=x+y+w
}
```



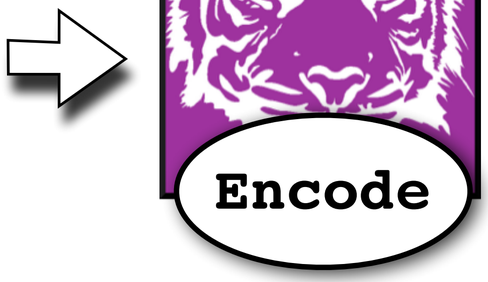


Diversifying Integer Expressions



Diversifying Integer Expressions

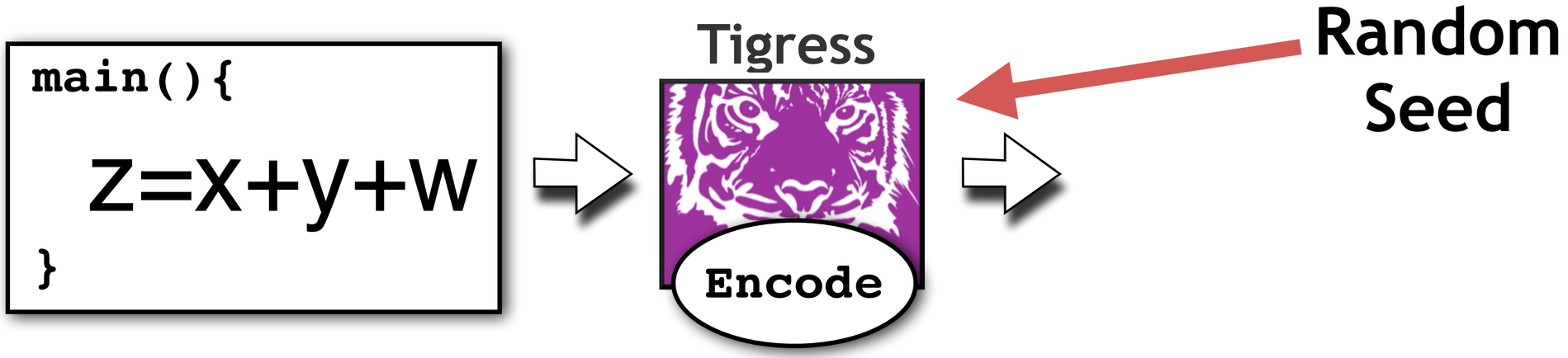
```
main(){
  z=x+y+w
}
```



Random Seed

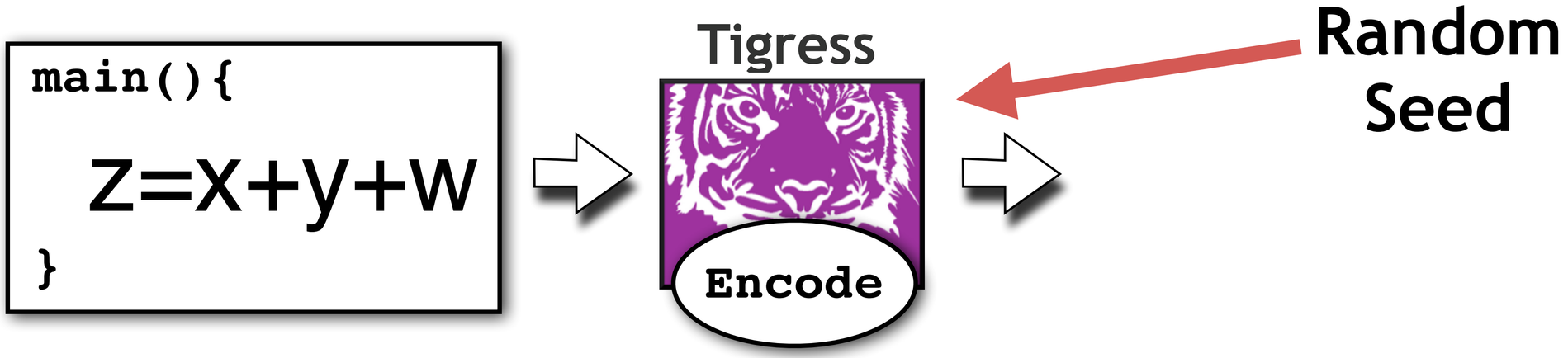
```
main(){
  z = ((x + y) ^ w) +
      (((x + y) & w) + ((x + y) & w)));
}
```

Diversifying Integer Expressions




```
main(){
    main(){
        z = ((x + y) | w) + ((x + y) & w));
    }
```

Diversifying Integer Expressions

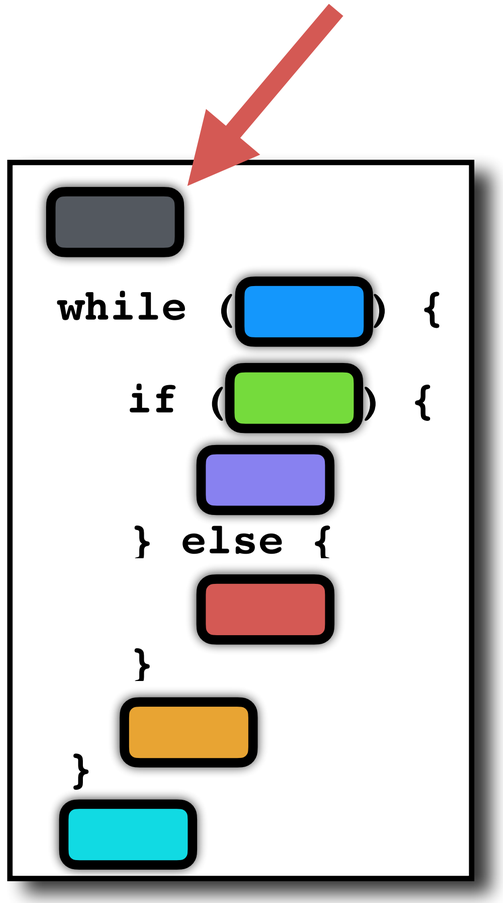



```
main(){
  main(){
    main(){
      z = (((x ^ y) + ((x & y) << 1)) | w) +
          (((x ^ y) + ((x & y) << 1)) & w);
    }
  }
}
```

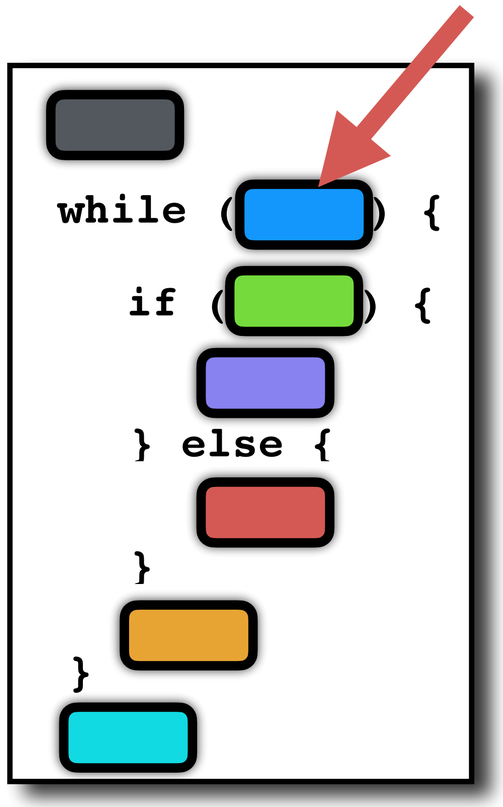
Diversifying Integer Expressions

```
main(){
    z = (((((x ^ ~ y) +
          ((x | y) + (x | y))) +   1) ^ ~ w) +
          (((((x ^ ~ y) +
         ((x | y) + (x | y))) + 1) | w) +
          ((((x ^ ~ y) + ((x | y) + (x | y)))
          + 1) | w))) + 1);
}
```

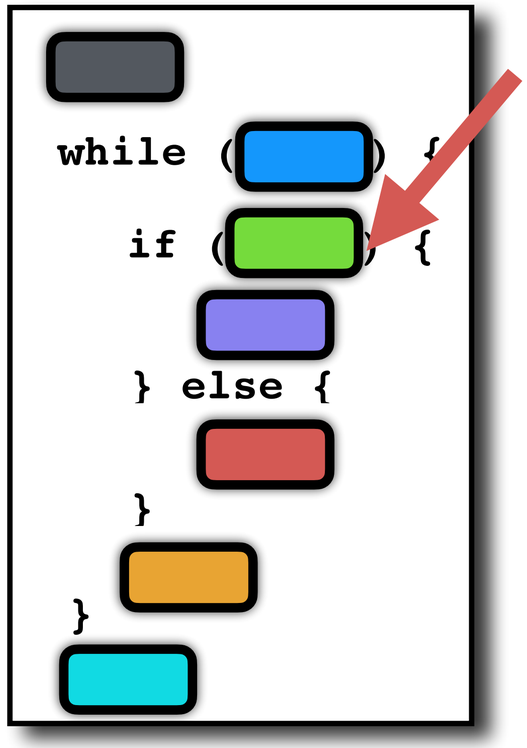
Diversifying Control Flow



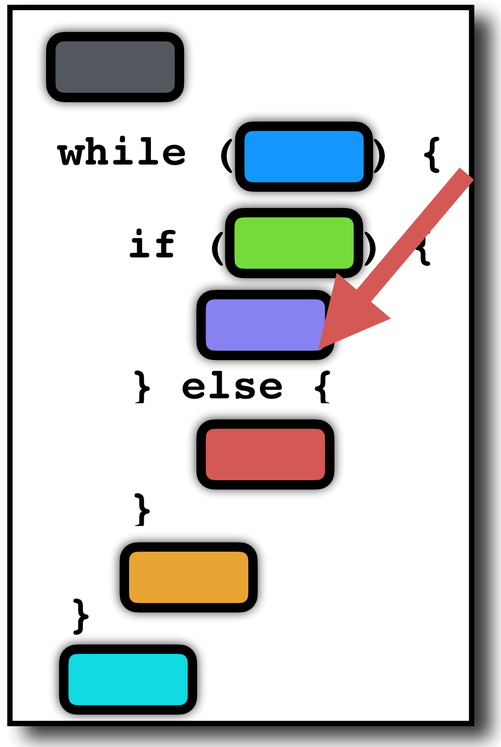
Diversifying Control Flow



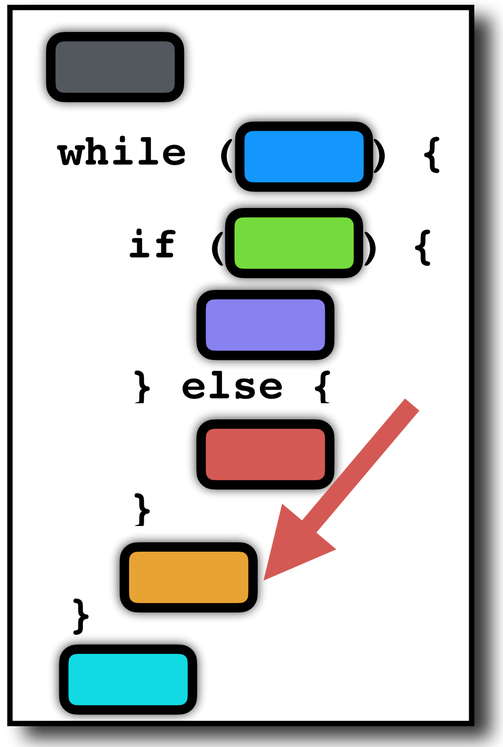
Diversifying Control Flow



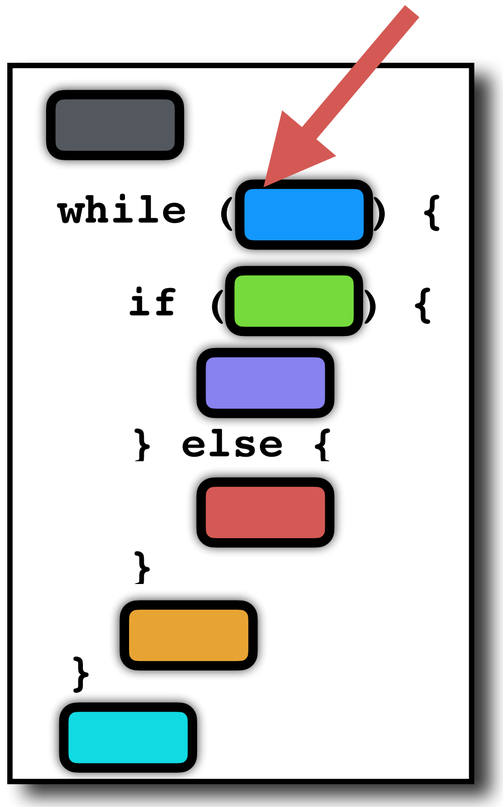
Diversifying Control Flow



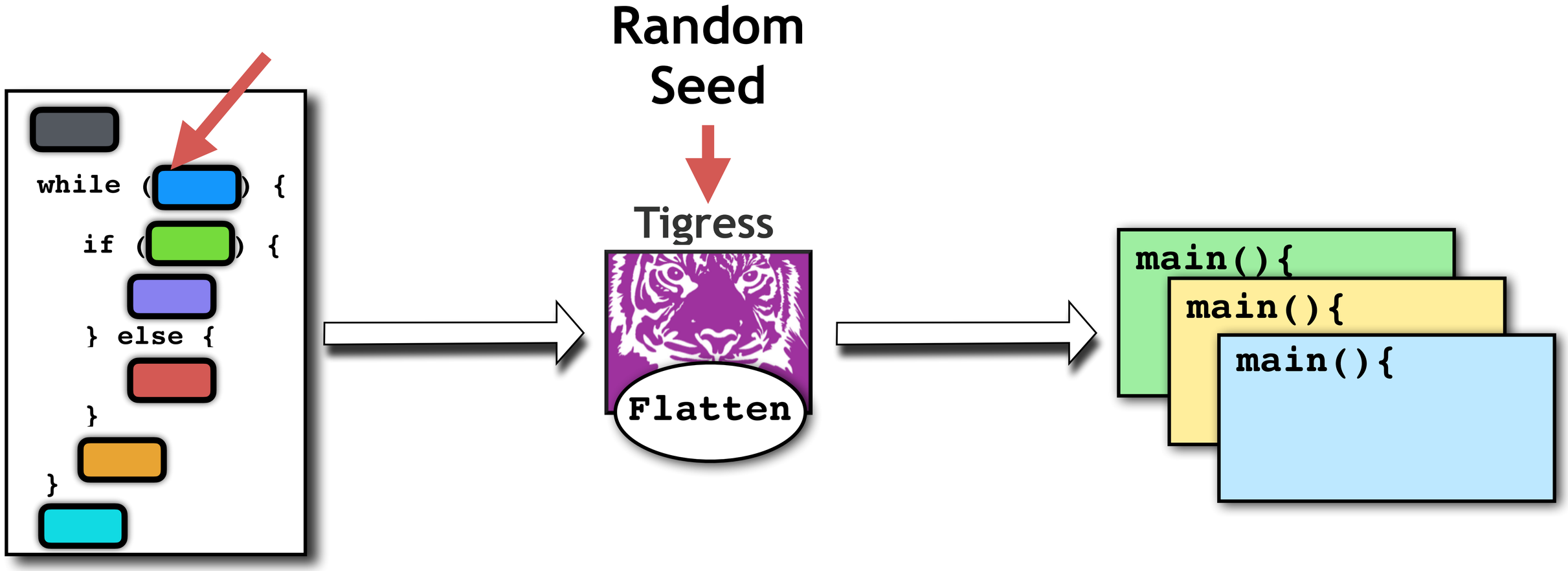
Diversifying Control Flow

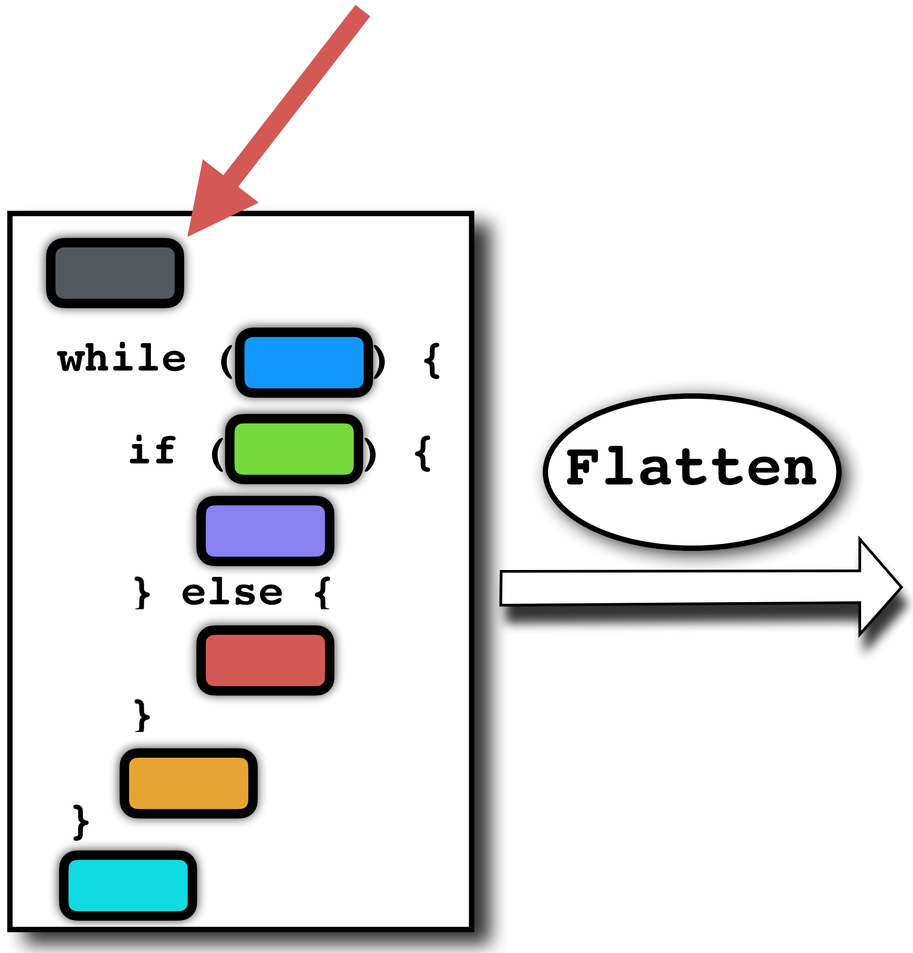


Diversifying Control Flow



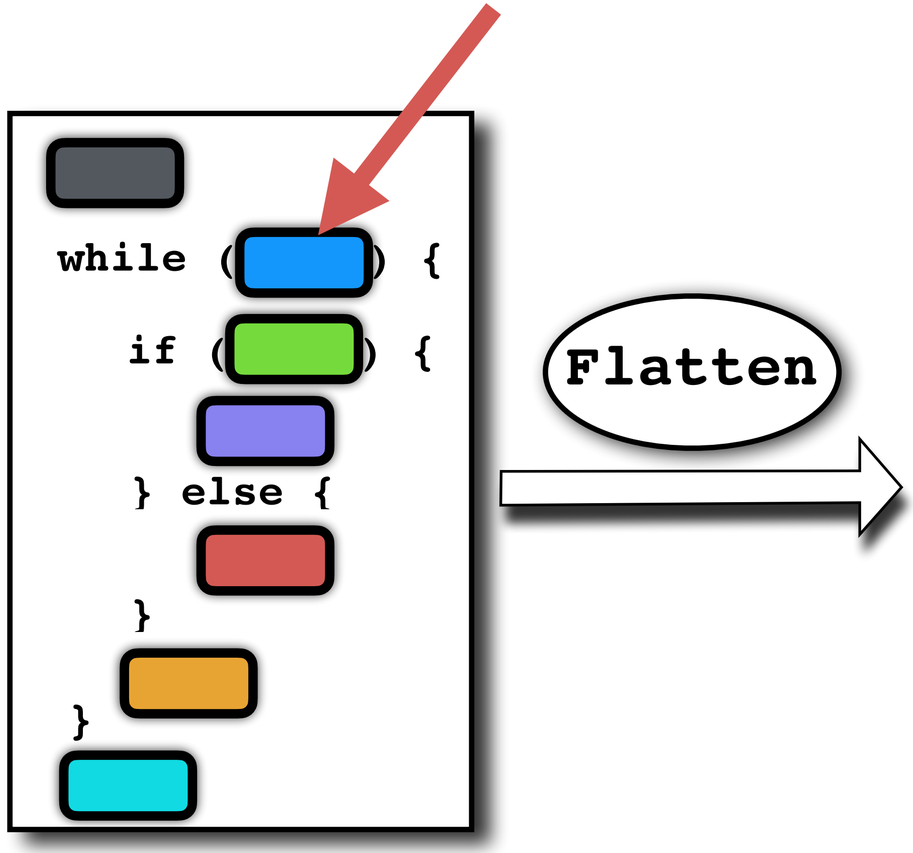
Diversifying Control Flow





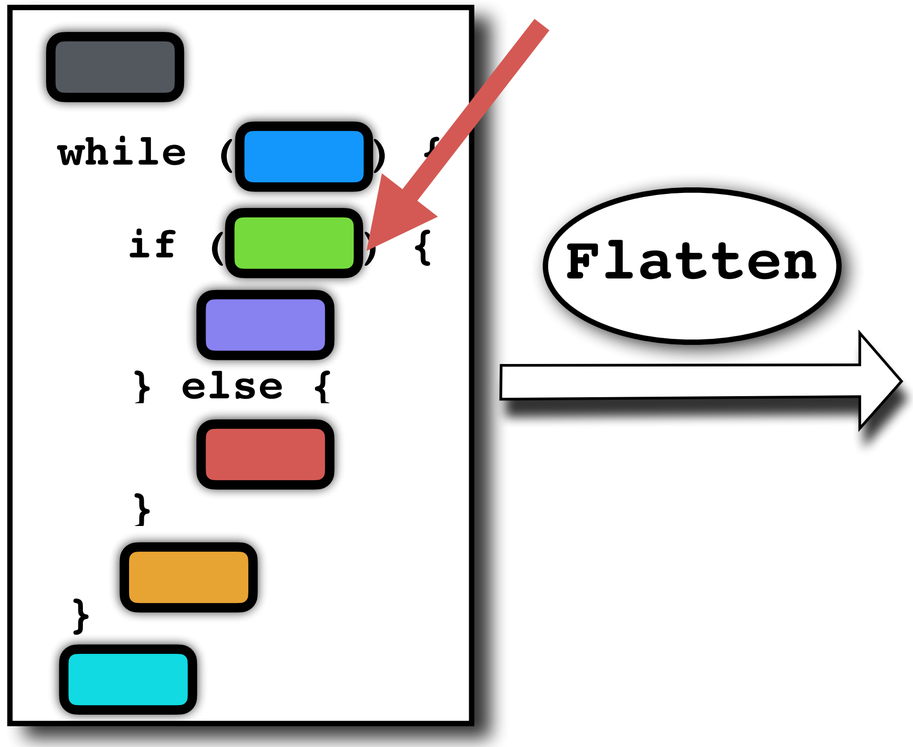
Flatten

```
int next=0;
for(;;)
    switch(next) {
    case 0: ; next=1; break;
    case 1: if ( ) next=2;
            else next=6; break;
    case 2: if ( ) next=3;
            else next=4; break;
    case 3: ; next=5; break;
    case 4: ; next=5; break;
    case 5: ; next=1; break;
    case 6: ; return L;
}
```



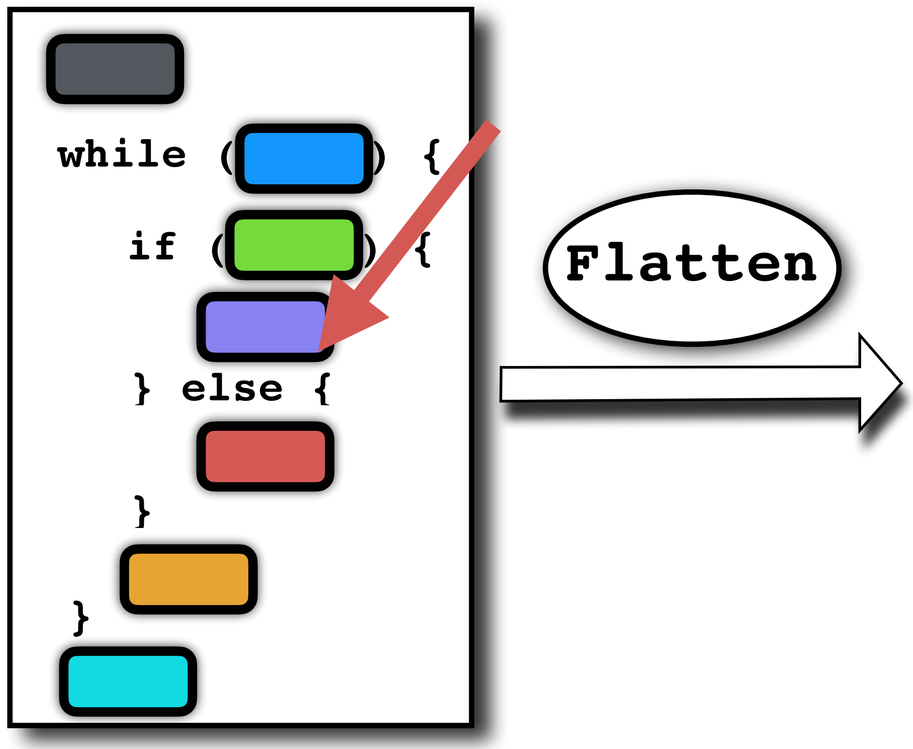
Flatten

```
int next=0;
for(;;)
    switch(next) {
    case 0: ; next=1; break;
    case 1: if ( ) next=2;
                else next=6; break;
    case 2: if ( ) next=3;
                else next=4; break;
    case 3: ; next=5; break;
    case 4: ; next=5; break;
    case 5: ; next=1; break;
    case 6: ; return L;
}
```



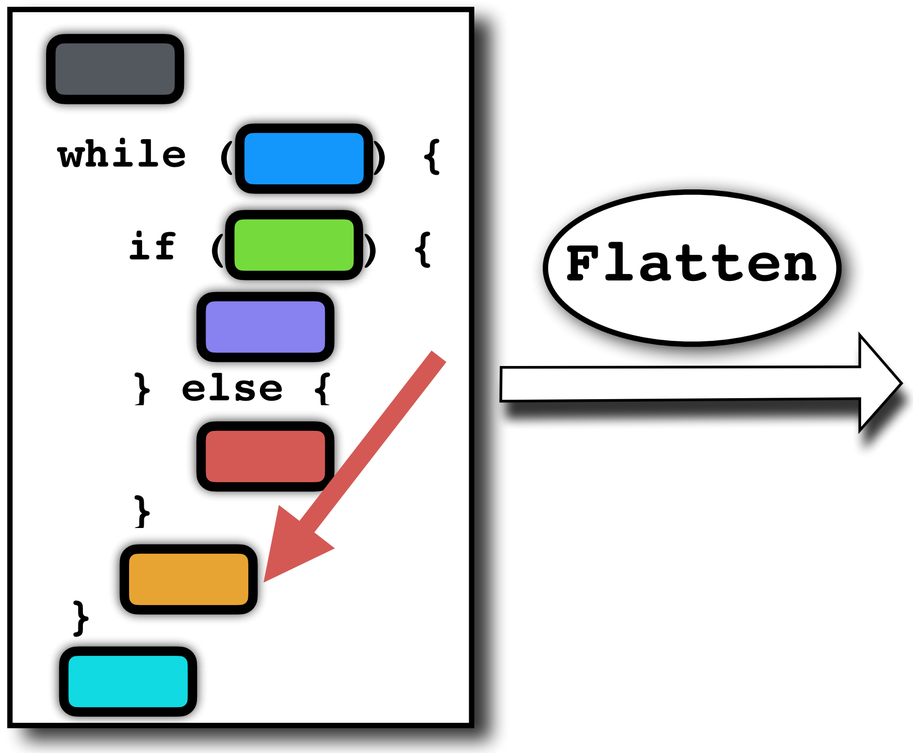
Flatten

```
int next=0;
for(;;)
    switch(next) {
    case 0:  ; next=1; break;
    case 1:  if (  ) next=2;
                 else next=6; break;
    case 2:  if (  ) next=3;
                 else next=4; break;
    case 3:  ; next=5; break;
    case 4:  ; next=5; break;
    case 5:  ; next=1; break;
    case 6:  ; return L;
}
```



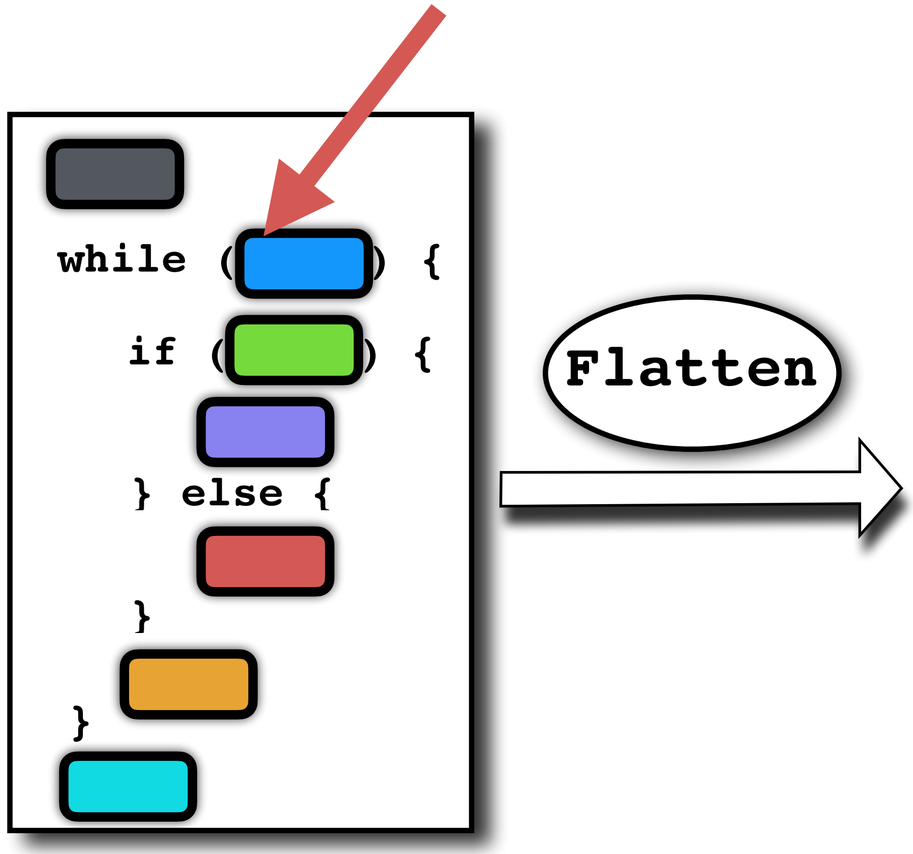
Flatten

```
int next=0;
for(;;)
    switch(next) {
    case 0: ; next=1; break;
    case 1: if ( ) next=2;
                else next=6; break;
    case 2: if ( ) next=3;
                else next=4; break;
    case 3: ; next=5; break;
    case 4: ; next=5; break;
    case 5: ; next=1; break;
    case 6: ; return L;
}
```



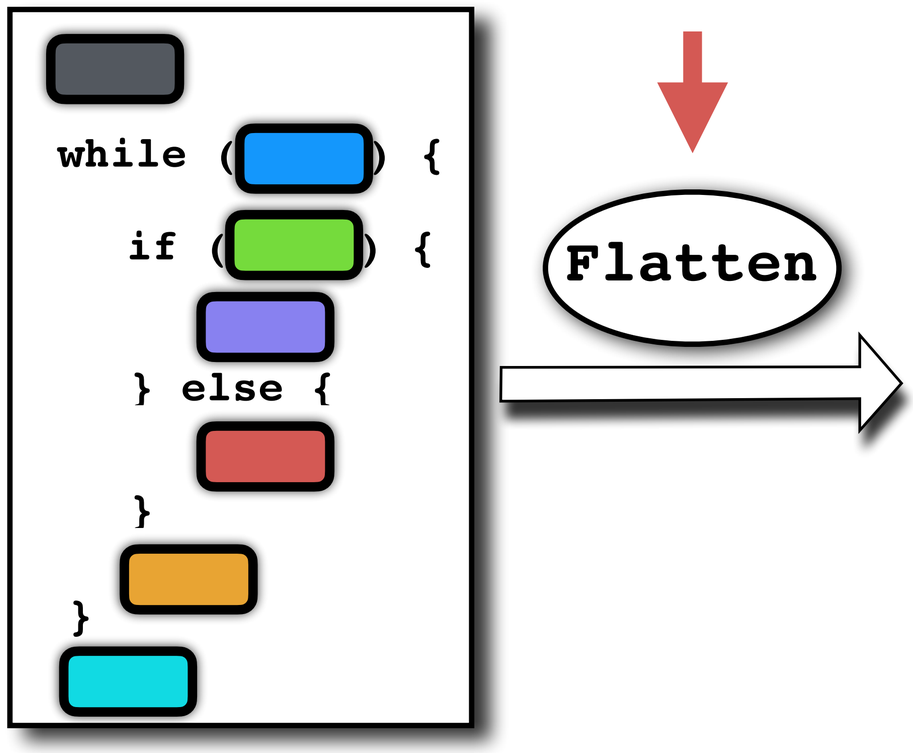
Flatten

```
int next=0;
for(;;)
    switch(next) {
    case 0:  ; next=1; break;
    case 1:  if ( ) next=2;
                 else next=6; break;
    case 2:  if ( ) next=3;
                 else next=4; break;
    case 3:  ; next=5; break;
    case 4:  ; next=5; break;
    case 5:  ; next=1; break;
    case 6:  ; return L;
}
```



Flatten

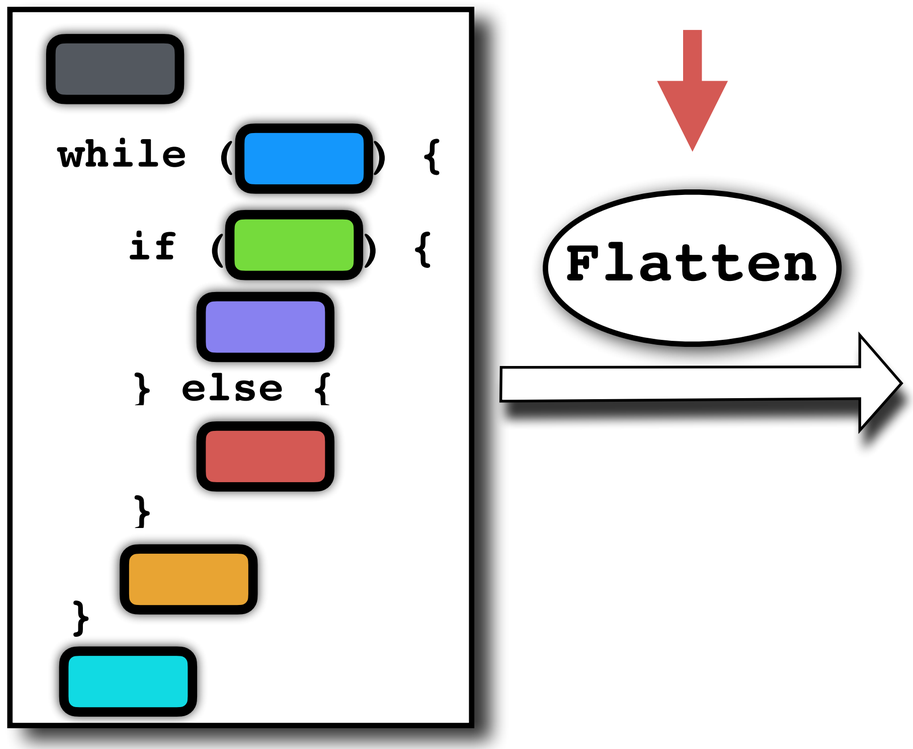
```
int next=0;
for(;;)
    switch(next) {
    case 0:  ; next=1; break;
    case 1:  if ( ) next=2;
             else next=6; break;
    case 2:  if ( ) next=3;
             else next=4; break;
    case 3:  ; next=5; break;
    case 4:  ; next=5; break;
    case 5:  ; next=1; break;
    case 6:  ; return L;
}
```



Seed=99

Flatten

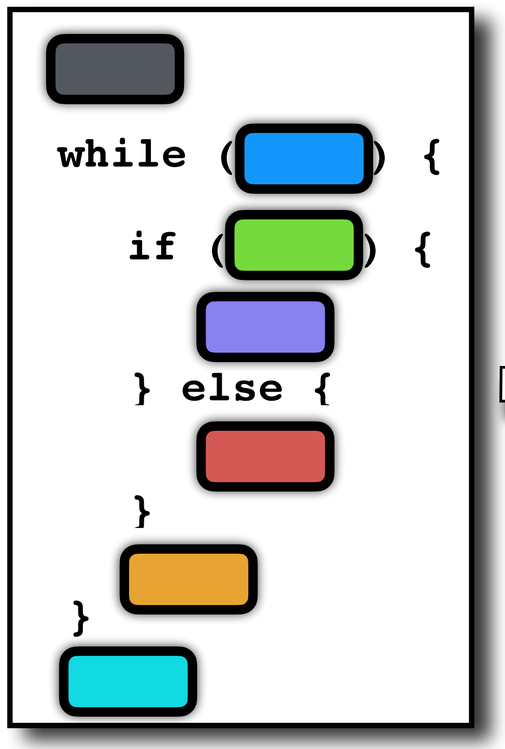
```
int next=0;
for(;;)
    switch(next) {
    case 0: ; next=1; break;
    case 1: if ( ) next=2;
                else next=6; break;
    case 2: if ( ) next=3;
                else next=4; break;
    case 3: ; next=5; break;
    case 4: ; next=5; break;
    case 5: ; next=1; break;
    case 6: ; return L;
}
```



Seed=13

Flatten

```
int next=6;
for(;;)
    switch(next) {
    case 0: ; return L;
    case 1: ; next=5; break;
    case 2: if ( ) next=1;
                else next=3; break;
    case 3: ; next=5; break;
    case 4: if ( ) next=2;
                else next=0; break;
    case 5: ; next=4; break;
    case 6: ; next=4; break;
}
```



Seed=42

Flatten

```
int next=7;
for(;;)
    switch(next) {
    case 2:  ; next=6; break;
    case 1:  ; return L;
    case 5:  if ( ) next=3;
                 else next=1; break;
    case 4:  ; next=6; break;
    case 7:  ; next=5; break;
    case 3:  if ( ) next=2;
                 else next=4; break;
    case 6:  ; next=5; break;
}
```

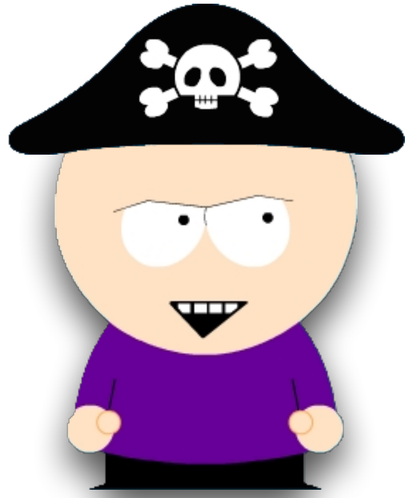
Typography

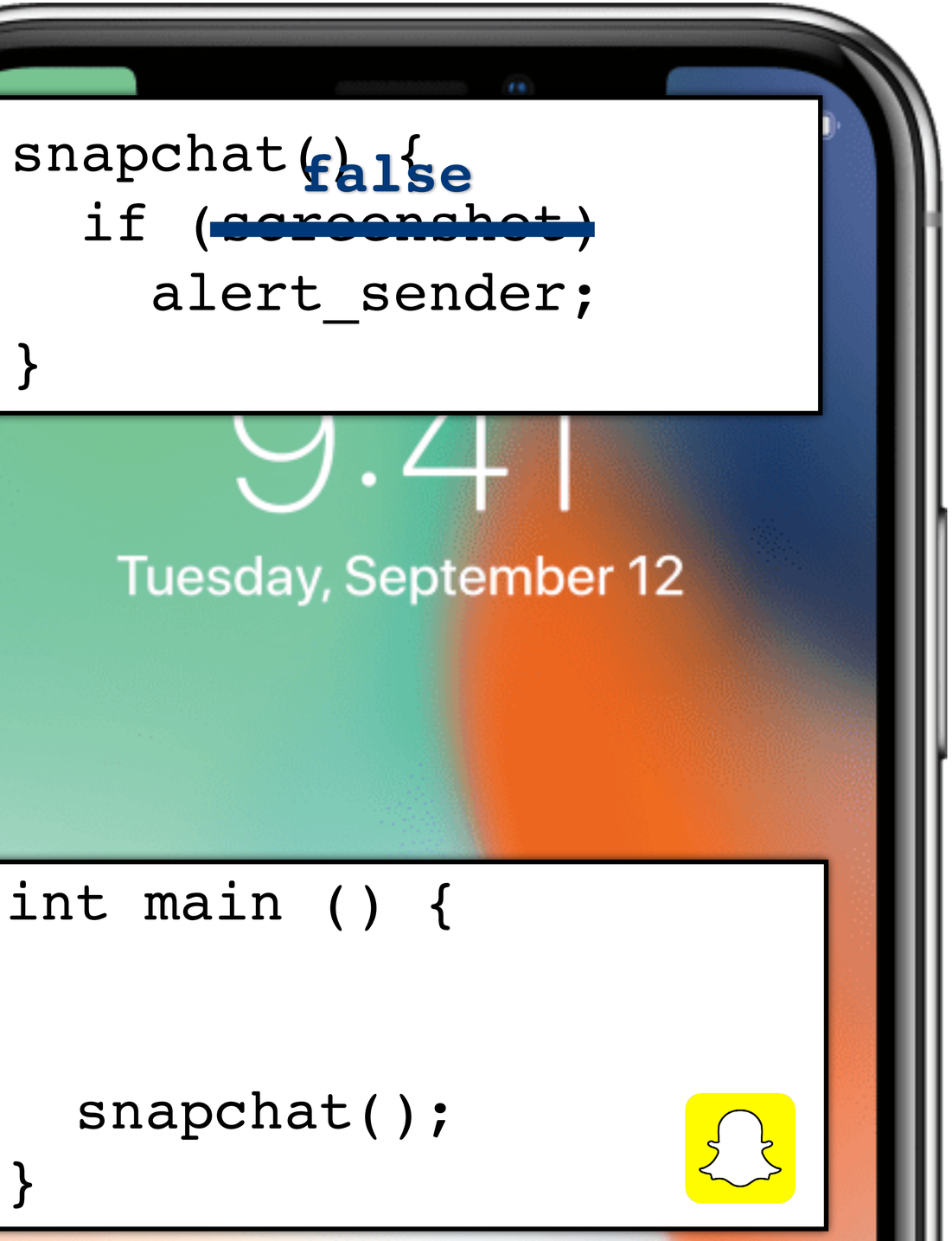
Algorithms

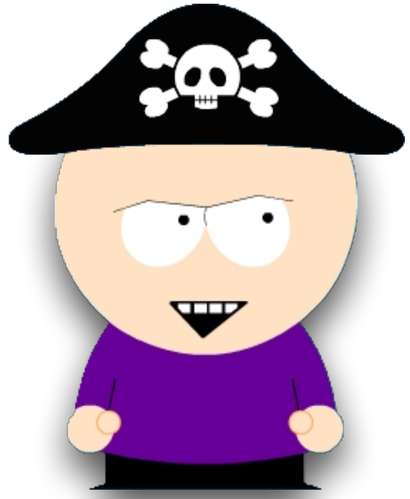
Tamperproofing Algorithms


snapchat() {
if (screenshot)
alert_sender;
}
9:41
Tuesday, September 12
int main () {
snapchat();
}

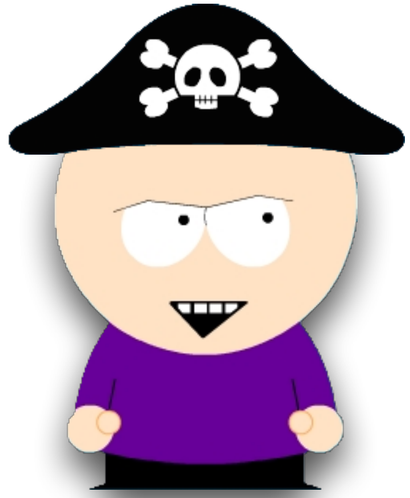
false




snapchat() {
false
if (screenshot)
alert_sender;
}
9:41
Tuesday, September 12
int main () {
snapchat();
}



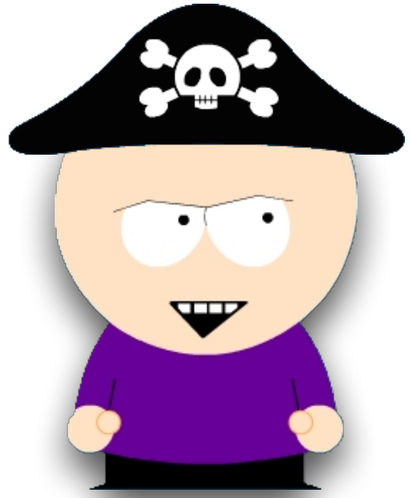

snapchat() {
if (screenshot)
alert_sender;
}
9:41
Tuesday, September 12
int main () {
if (Detect Tampering)
snapchat();
}



```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

```
respond_to_hack() {



}
```

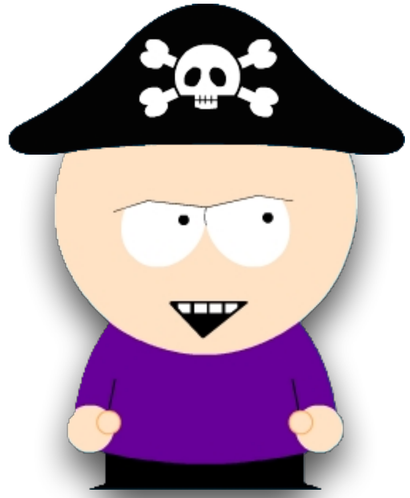
```
int main () {
  if (Detect Tampering)
    respond_to_hack();
  snapchat();
}
```



```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

```
respond_to_hack() {


}
```

```
int main () {
  if (Detect Tampering)
    respond_to_hack();
  snapchat();
}
```

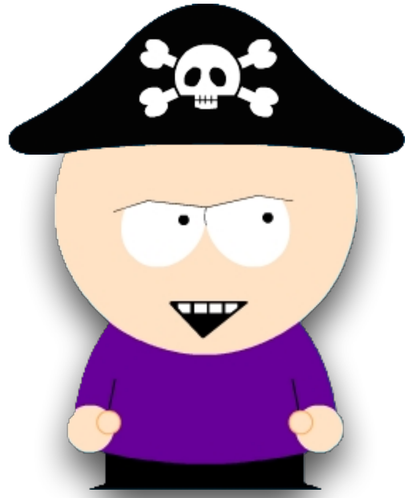


Bob hacked me!!!

```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

```
respond_to_hack() {
   ◆ crash the program
   ◆ refuse to run
   ◆ run slower
   ◆ make wrong results
}
```

```
int main () {
  if (Detect Tampering)
      respond_to_hack();
  snapchat();
}
```

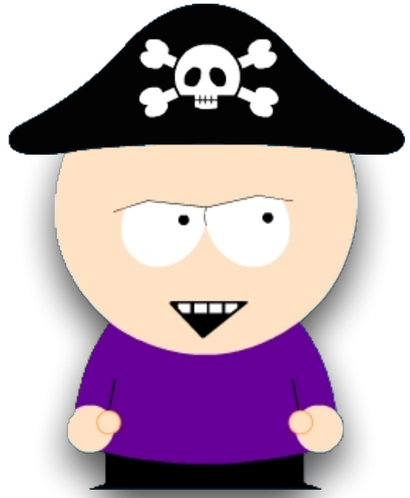


Bob hacked me!!!

```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

```
respond_to_hack() {
  ◆ crash the program
  ◆ refuse to run
  ◆ run slower
  ◆ make wrong results
}
```

```
int main () {
  if (hash() != 0x4C49F346)
    respond_to_hack();
  snapchat();
}
```



Bob hacked me!!!

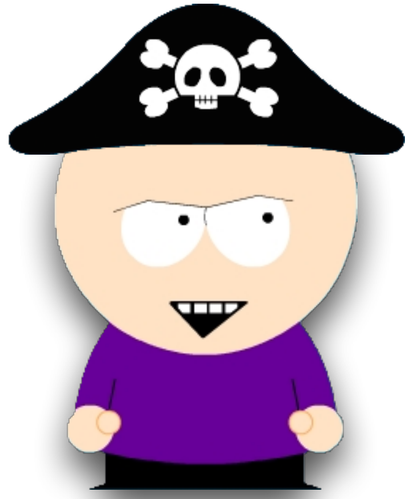
```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

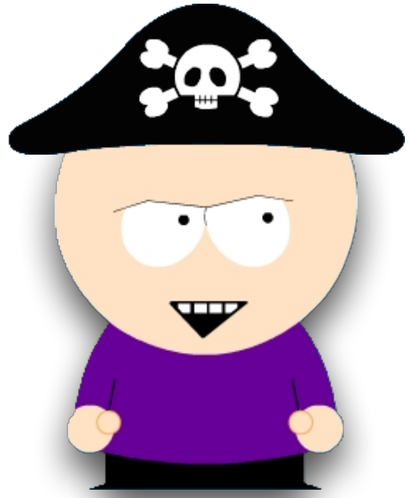
```
respond_to_hack() {
  ◆ crash the program
  ◆ refuse to run
  ◆ run slower
  ◆ make wrong results
}
```

```
int main () {
  if (hash() != 0x4C49F346)
    respond_to_hack();
  snapchat();
}
```

false

false





```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

false

```
respond_to_hack() {
  ◆ crash the program
  ◆ refuse to run
  ◆ run slower
  ◆ make wrong results
}
```

```
int main() {
  if (hash() != 0x1C497346)
    respond_to_hack();
  snapchat();
}
```

false

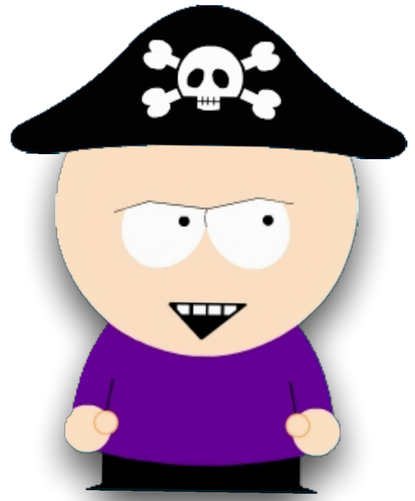
```
snapchat() {
  if (screenshot)
    alert_sender;
}
```

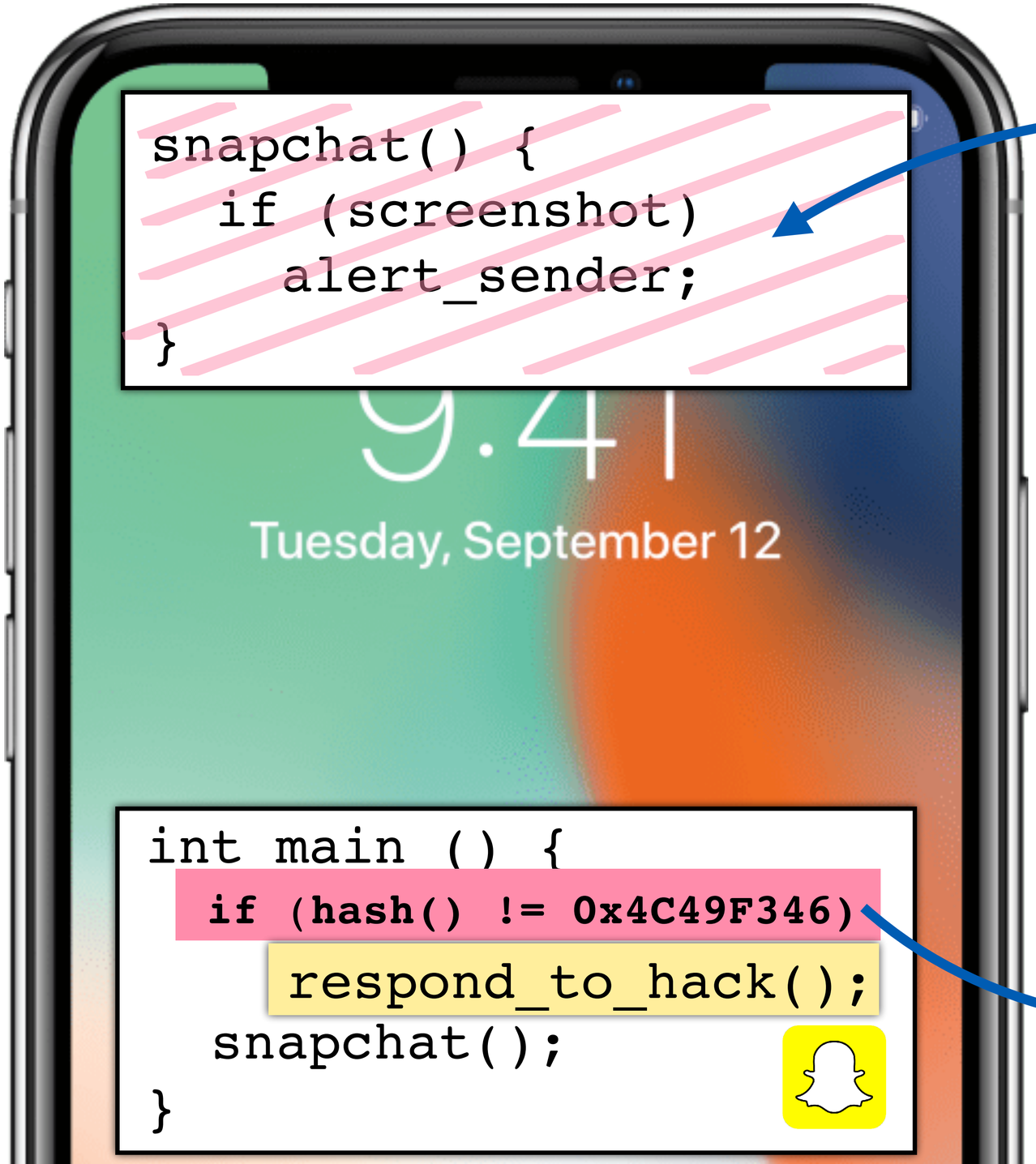
false

```
respond_to_hack() {
  ◆ crash the program
  ◆ refuse to run
  ◆ run slower
  ◆ make wrong results
}
```

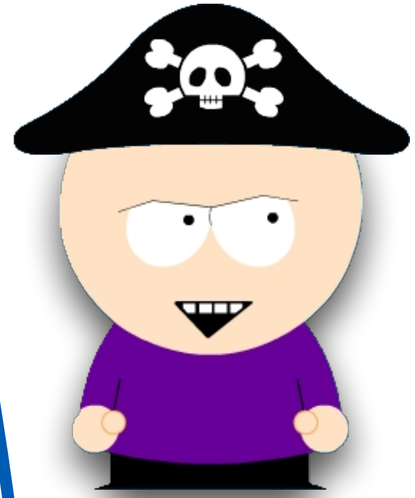
```
int main() {
  if (hash() != 0x1C497346)
    respond_to_hack();
  snapchat();
}
```

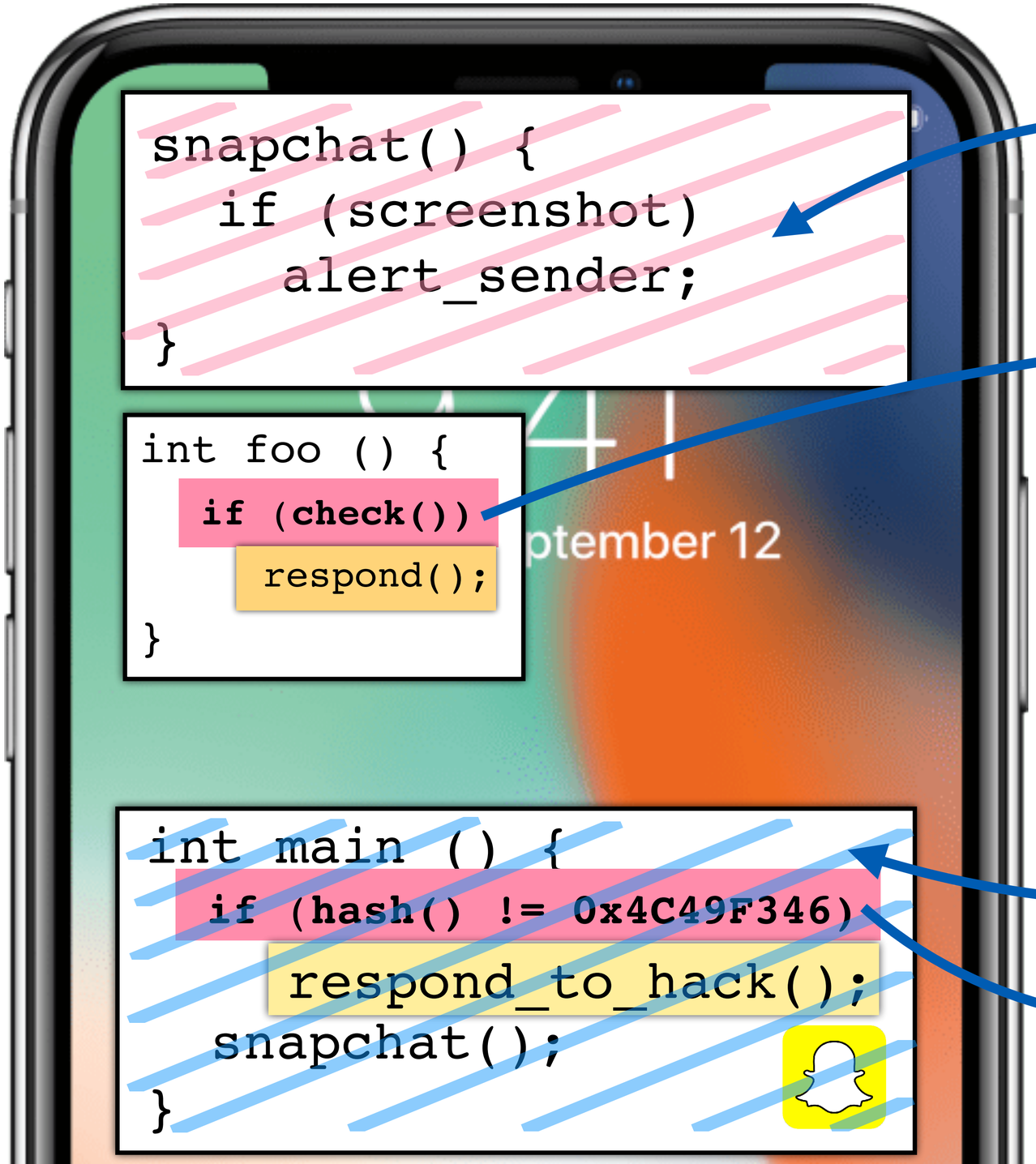
false




snapchat() {
if (screenshot)
alert_sender;
}
9:41
Tuesday, September 12
int main () {
if (hash() != 0x4C49F346)
respond_to_hack();
snapchat();
}

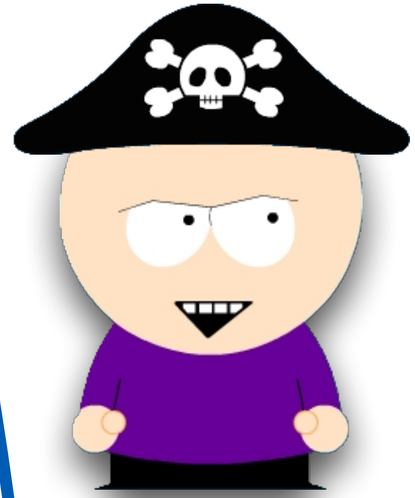
Check!

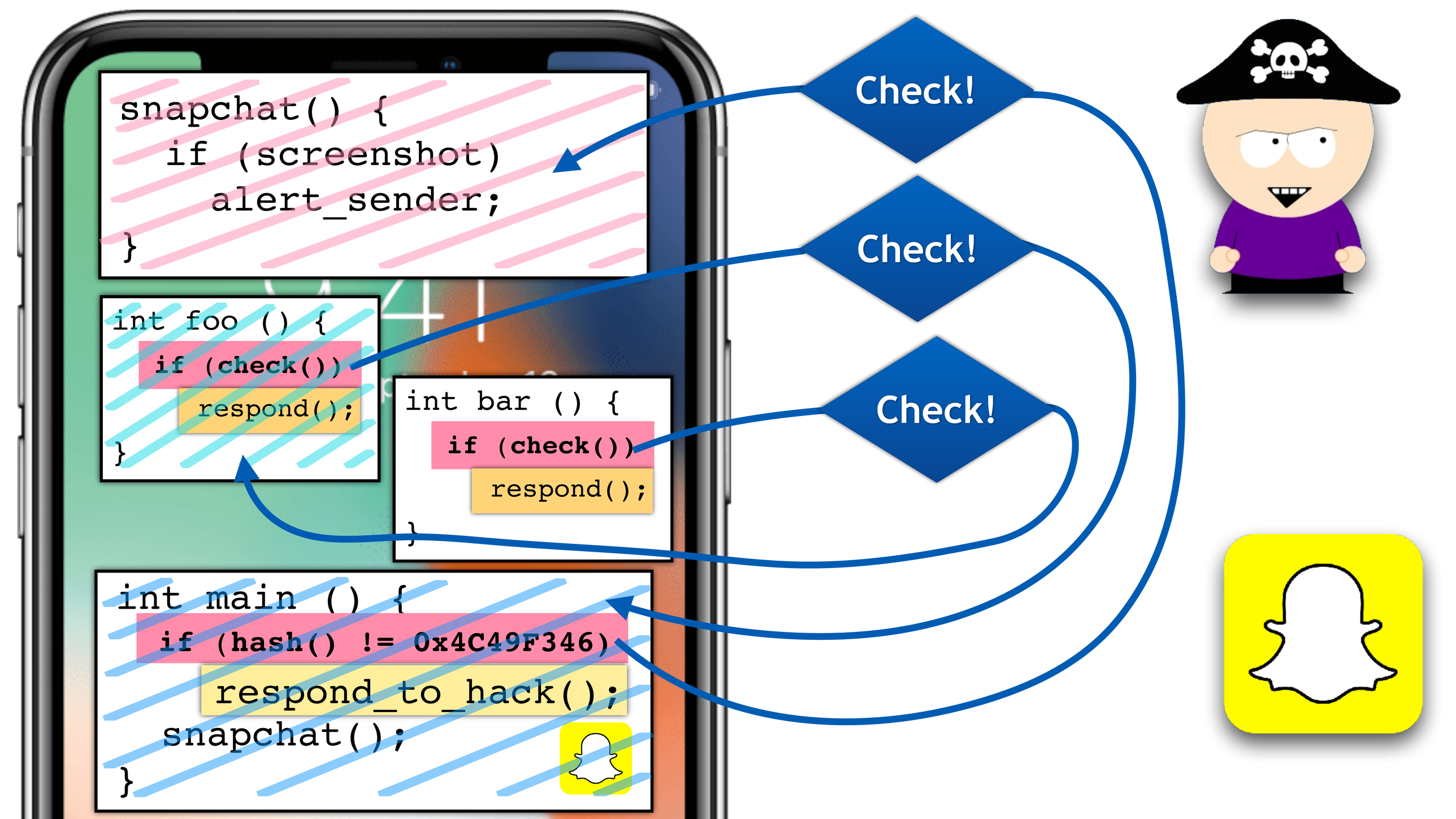



snapchat() {
if (screenshot)
alert_sender;
}
int foo () {
if (check())
respond();
}
int main () {
if (hash() != 0x4C49F346)
respond_to_hack();
snapchat();
}

Check!

Check!




snapchat() {
if (screenshot)
alert_sender;
}
int foo () {
if (check())
respond();
}
int bar () {
if (check())
respond();
}
int main () {
if (hash() != 0x4C49F346)
respond_to_hack();
snapchat();
}
Check!
Check!
Check!

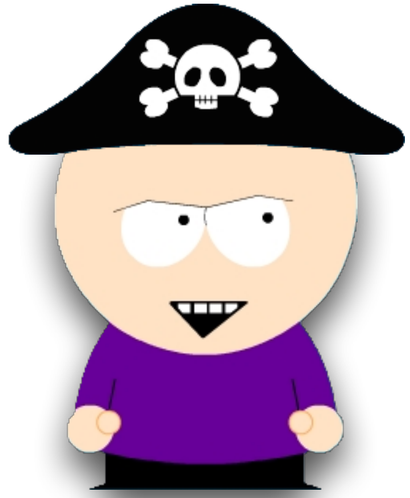
```
int foo () {
  if (check())
    respond();
}
```

```
int foo () {
  if (check())
    respond();
}
```

```
int bar () {
  if (check())
    REPAIR();
}
```

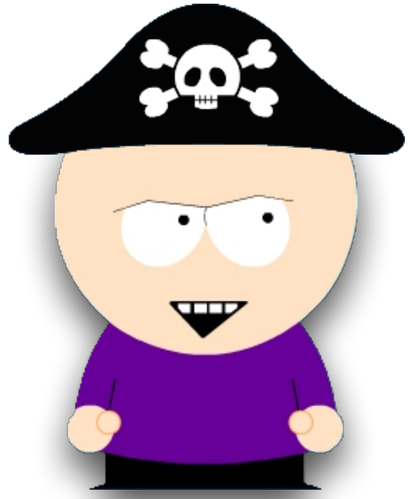
Repair

```
int main () {
  if (hash() != 0x4C49F346)
    respond_to_hack();
  snapchat();
}
```




9:41
int foo () {
if (check())
respond();
}
int bar () {
if (check())
REPAIR();
}
int main () {
if (hash() != 0x4C49F346)
respond_to_hack();
snapchat();
}

Repair



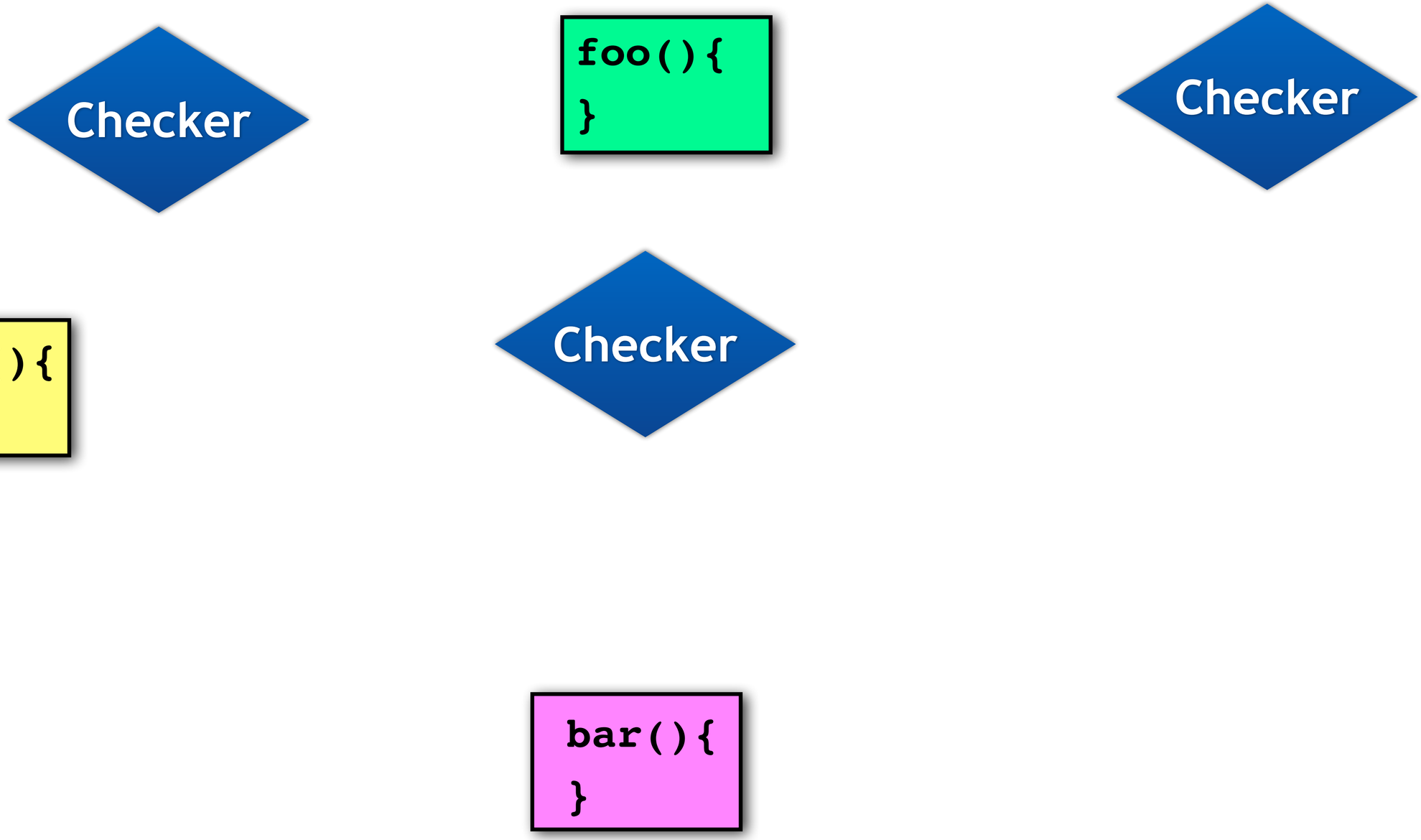
Check/Repair Network

```
foo(){
}
```

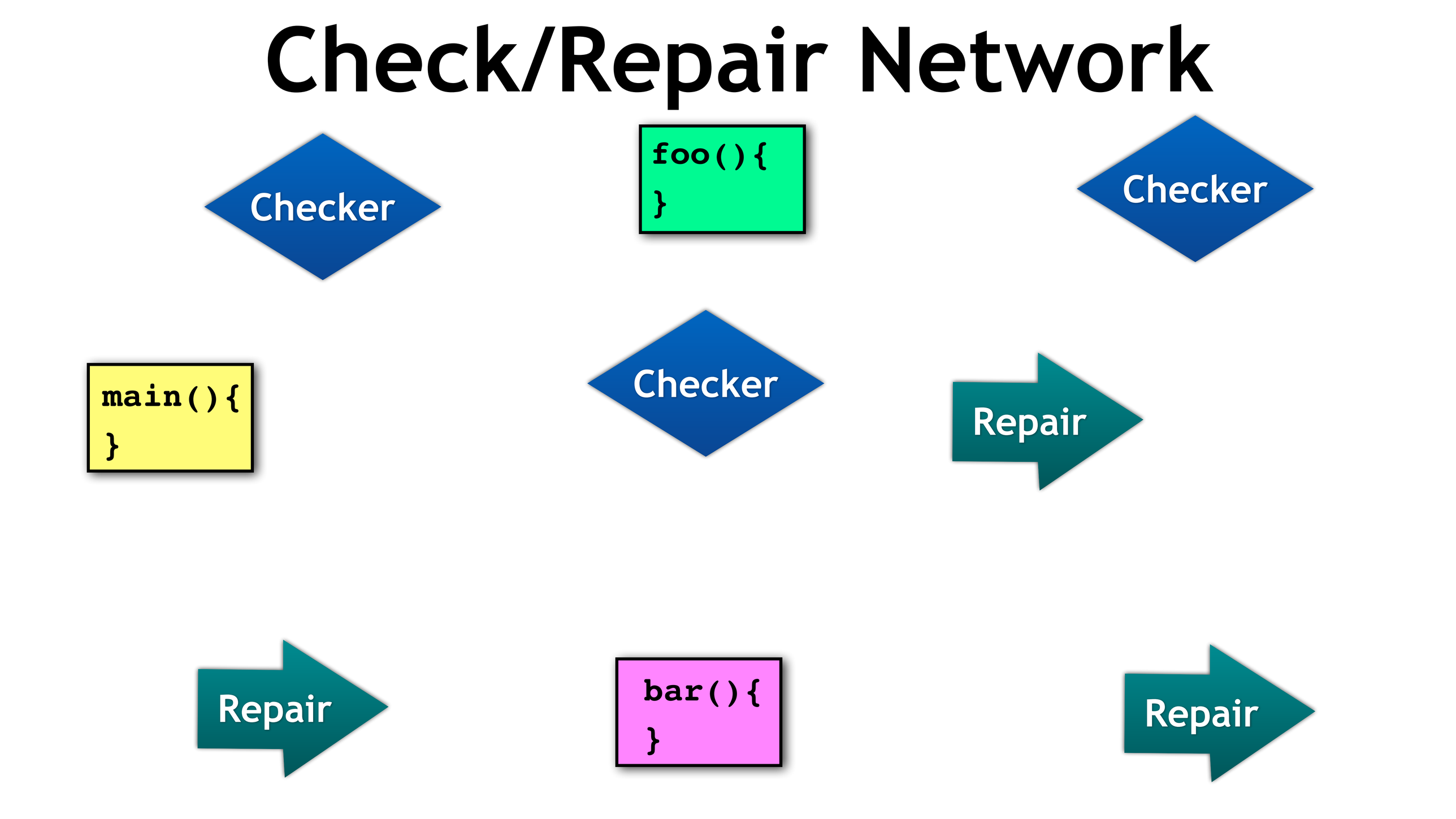
```
main(){
}
```

```
bar(){
}
```

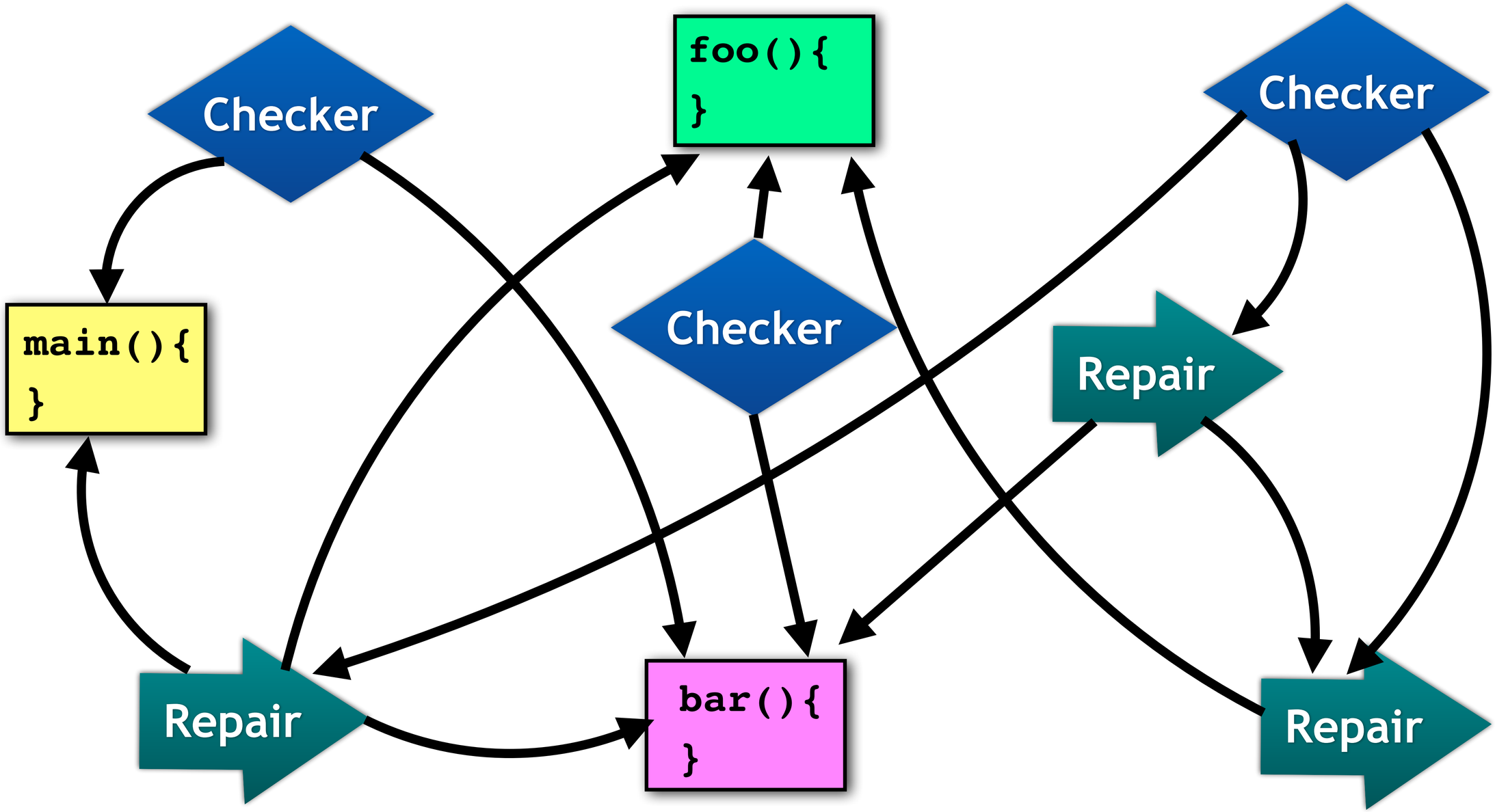
Check/Repair Network



Check/Repair Network



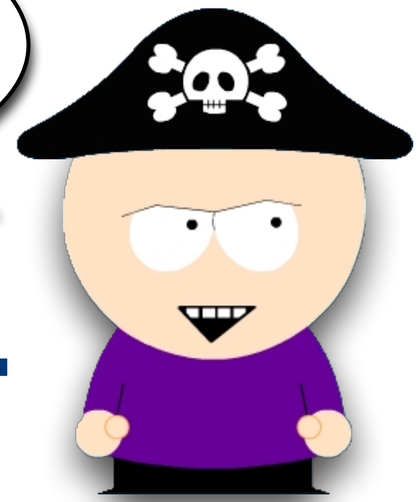
Check/Repair Network



Obfuscation Algorithms: Virtualisation

Obfuscation Algorithms: Virtualization

Where to edit???

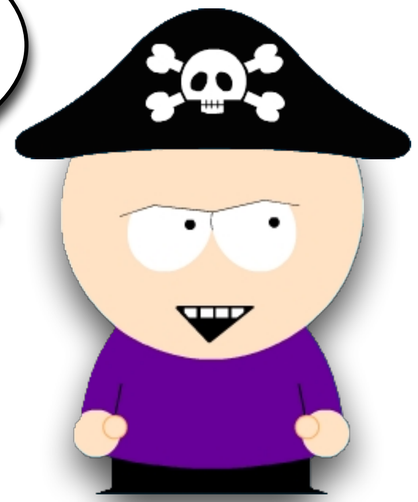


`false`


snapchat() {
false
if (screenshot)
alert_sender;
}
9:41
Tuesday, September 12
int main () {
snapchat();
}

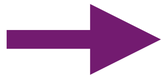

Where to edit???

Where to edit???



Haha!
Analyze this!

```
P(){
   …
}
```



Tigress

```
P(){
   …
}
```

Tigress

Opcode	Mnemonic	Semantics
0	add	push(pop()+pop())
1	store L	Mem[L]=pop()
2	breq L	if pop()=pop() goto L

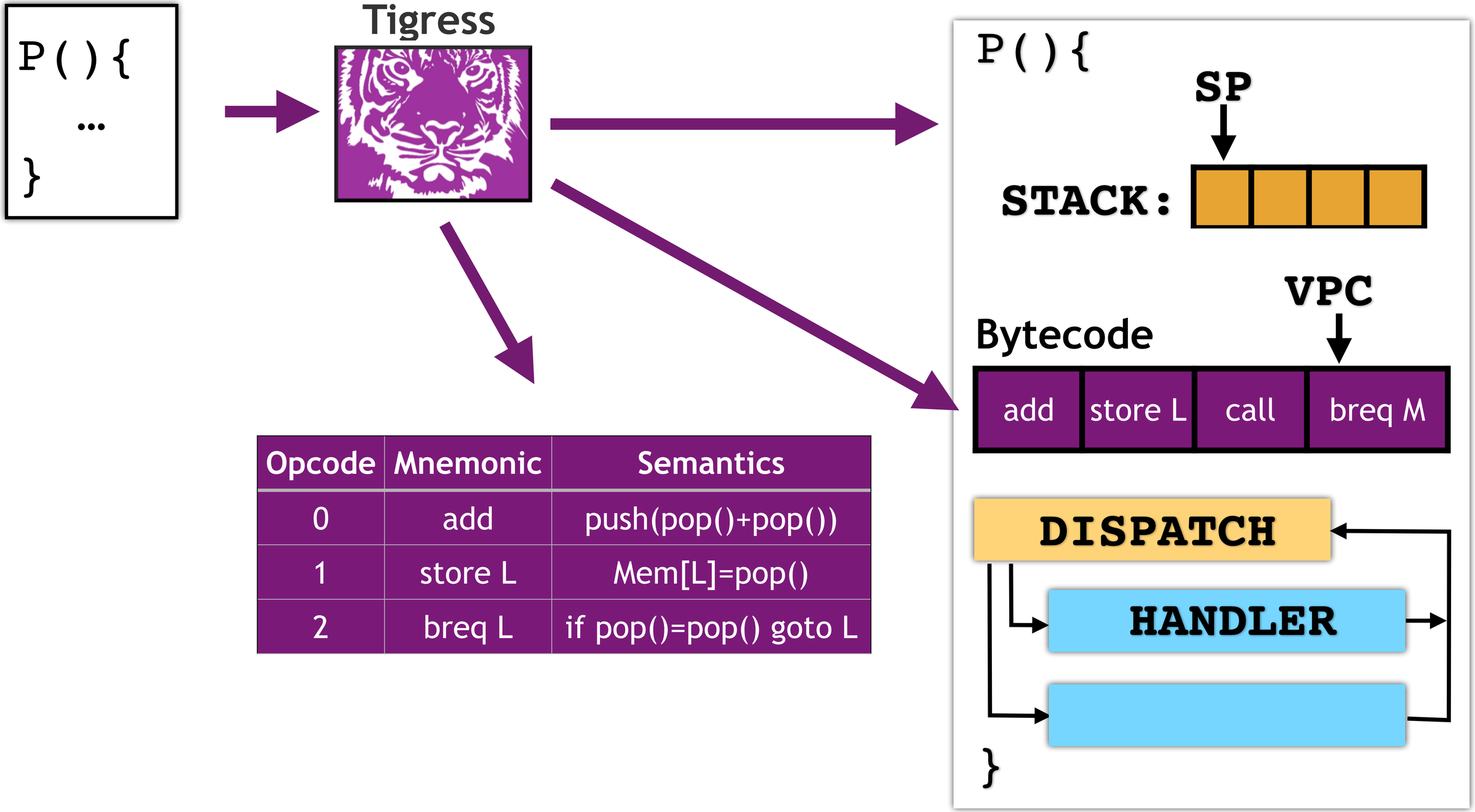
```
P(){
   …
}
```

Tigress

Bytecode

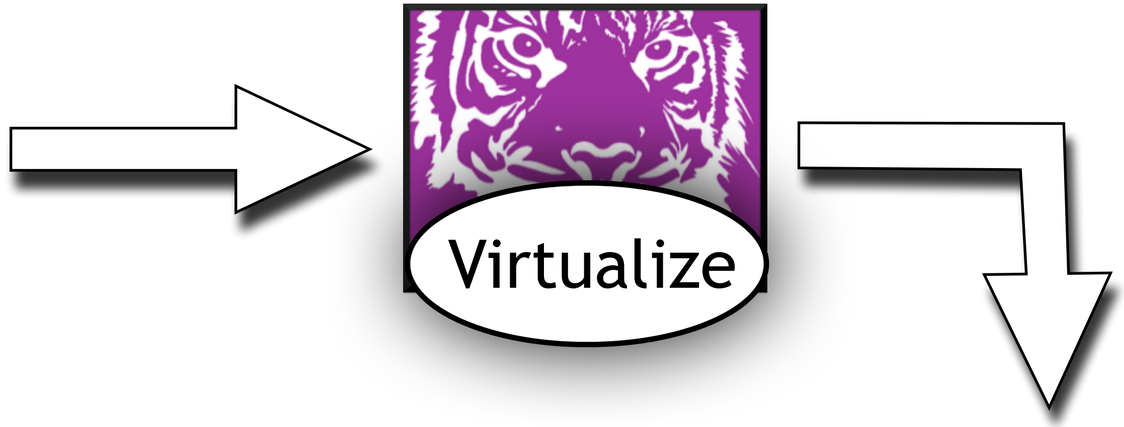
add	store L	call	breq M

Opcode	Mnemonic	Semantics
0	add	push(pop()+pop())
1	store L	Mem[L]=pop()
2	breq L	if pop()=pop() goto L



Opcode	Mnemonic	Semantics
0	add	push(pop()+pop())
1	store L	Mem[L]=pop()
2	breq L	if pop()=pop() goto L

```
int main(){
 int x;
 x++;
}
```



```
int main(){
 int x;
 x++;
}
```


```
enum ops {Locals = 116, Plus  = 135, Load   = 60, Goto = 231,
          Const  = 3,   Store = 122, Return = 72};
unsigned char bytecode[41]  = {
 Locals,24,0,0,0,Const,1,0,0,0,Locals,24,0,0,0,Load,Plus,Store,Locals,
 28,0,0,0,Const,0,0,0,0,Store,Goto,4,0,0,0,Locals,28,0,0,0,Load,Return};
int main() {
  while (1) {
    switch (*pc) {
    case Const: pc++; (sp+1)->_int = *((int *)pc); sp++; pc+=4; break;
    case Load: pc++; sp->_int = *((int *)sp->_vs); break;
    case Goto: pc++; pc+= *((int *)pc); break;
    case Plus: pc++; (sp+-1)->_int=sp->_int+(sp+-1)->_int; sp--; break;
    case Return: pc++; return (sp->_int); break;
    case Store: pc++; *((int *)(sp+-1)->_vs)=sp->_int; sp+=-2; break;
    case Locals: pc++; (sp+1)->_vs=(void *)(vars+*((int *)pc));
                 sp++; pc+=4; break;
}}
```

```
int main(){
 int x;
 x++;
}
```


```
enum ops {Locals = 116, Plus  = 135, Load   = 60, Goto = 231,
          Const  = 3,   Store = 122, Return = 72};
unsigned char bytecode[41]  = {
 Locals,24,0,0,0,Const,1,0,0,0,Locals,24,0,0,0,Load,Plus,Store,Locals,
 28,0,0,0,Const,0,0,0,0,Store,Goto,4,0,0,0,Locals,28,0,0,0,Load,Return};
int main() {
  while (1) {
    switch (*pc) {
    case Const: pc++; (sp+1)->_int = *((int *)pc); sp++; pc+=4; break;
    case Load: pc++; sp->_int = *((int *)sp->_vs); break;
    case Goto: pc++; pc+= *((int *)pc); break;
    case Plus: pc++; (sp+-1)->_int=sp->_int+(sp+-1)->_int; sp--; break;
    case Return: pc++; return (sp->_int); break;
    case Store: pc++; *((int *)(sp+-1)->_vs)=sp->_int; sp+=-2; break;
    case Locals: pc++; (sp+1)->_vs=(void *)(vars+*((int *)pc));
                 sp++; pc+=4; break;
}}
```

```
int main(){
 int x;
 x++;
}
```


```
enum ops {Locals = 116, Plus  = 135, Load   = 60, Goto = 231,
          Const  = 3,   Store = 122, Return = 72};
unsigned char bytecode[41]  = {
 Locals,24,0,0,0,Const,1,0,0,0,Locals,24,0,0,0,Load,Plus,Store,Locals,
 28,0,0,0,Const,0,0,0,0,Store,Goto,4,0,0,0,Locals,28,0,0,0,Load,Return};
int main() {
  while (1) {
    switch (*pc) {
    case Const: pc++; (sp+1)->_int = *((int *)pc); sp++; pc+=4; break;
    case Load: pc++; sp->_int = *((int *)sp->_vs); break;
    case Goto: pc++; pc+= *((int *)pc); break;
    case Plus: pc++; (sp+-1)->_int=sp->_int+(sp+-1)->_int; sp--; break;
    case Return: pc++; return (sp->_int); break;
    case Store: pc++; *((int *)(sp+-1)->_vs)=sp->_int; sp+=-2; break;
    case Locals: pc++; (sp+1)->_vs=(void *)(vars+*((int *)pc));
                 sp++; pc+=4; break;
}}
```

```
int main(){
 int x;
 x++;
}
```


```
enum ops {Locals = 116, Plus  = 135, Load   = 60, Goto = 231,
          Const  = 3,   Store = 122, Return = 72};
unsigned char bytecode[41]  = {
 Locals,24,0,0,0,Const,1,0,0,0,Locals,24,0,0,0,Load,Plus,Store,Locals,
 28,0,0,0,Const,0,0,0,0,Store,Goto,4,0,0,0,Locals,28,0,0,0,Load,Return};
int main() {
  while (1) {
    switch (*pc) {
    case Const: pc++; (sp+1)->_int = *((int *)pc); sp++; pc+=4; break;
    case Load: pc++; sp->_int = *((int *)sp->_vs); break;
    case Goto: pc++; pc+= *((int *)pc); break;
    case Plus: pc++; (sp+-1)->_int=sp->_int+(sp+-1)->_int; sp--; break;
    case Return: pc++; return (sp->_int); break;
    case Store: pc++; *((int *)(sp+-1)->_vs)=sp->_int; sp+=-2; break;
    case Locals: pc++; (sp+1)->_vs=(void *)(vars+*((int *)pc));
                 sp++; pc+=4; break;
}}
```

```
int main(){
 int x;
 x++;
}
```


```
enum ops {Locals = 116, Plus  = 135, Load   = 60, Goto = 231,
          Const  = 3,   Store = 122, Return = 72};
unsigned char bytecode[41]  = {
 Locals,24,0,0,0,Const,1,0,0,0,Locals,24,0,0,0,Load,Plus,Store,Locals,
 28,0,0,0,Const,0,0,0,0,Store,Goto,4,0,0,0,Locals,28,0,0,0,Load,Return};
int main() {
  while (1) {
    switch (*pc) {
    case Const: pc++; (sp+1)->_int = *((int *)pc); sp++; pc+=4; break;
    case Load: pc++; sp->_int = *((int *)sp->_vs); break;
    case Goto: pc++; pc+= *((int *)pc); break;
    case Plus: pc++; (sp+-1)->_int=sp->_int+(sp+-1)->_int; sp--; break;
    case Return: pc++; return (sp->_int); break;
    case Store: pc++; *((int *)(sp+-1)->_vs)=sp->_int; sp+=-2; break;
    case Locals: pc++; (sp+1)->_vs=(void *)(vars+*((int *)pc));
                 sp++; pc+=4; break;
}}
```

```
main() {
    int x;
    x = 20 + 22;
}
```

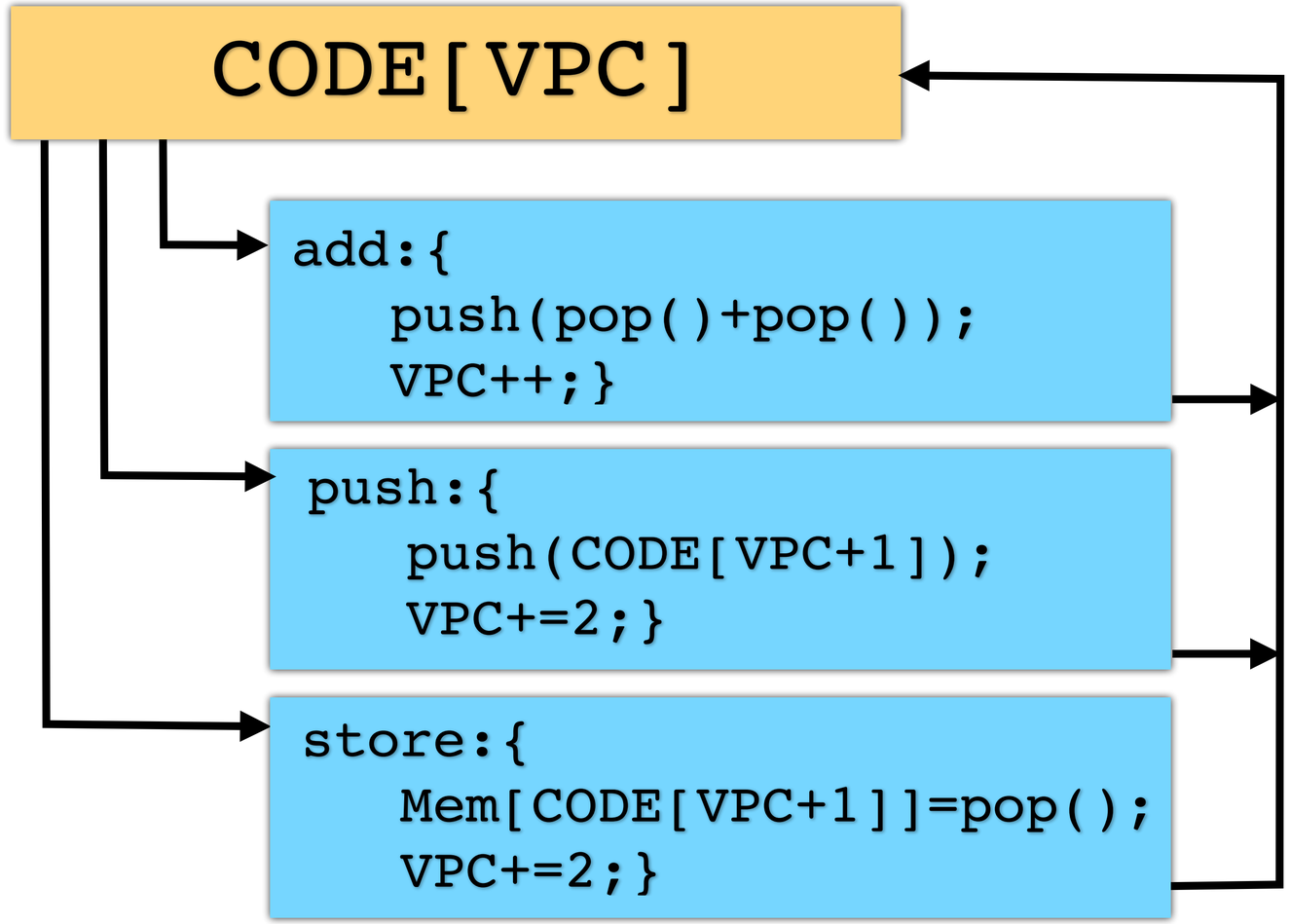
```
main() {
    int x;
    x = 20 + 22;
}
```

push 20	push 22	add	store &x

```
main() {
    int x;
    x = 20 + 22;
}
```

x: 99

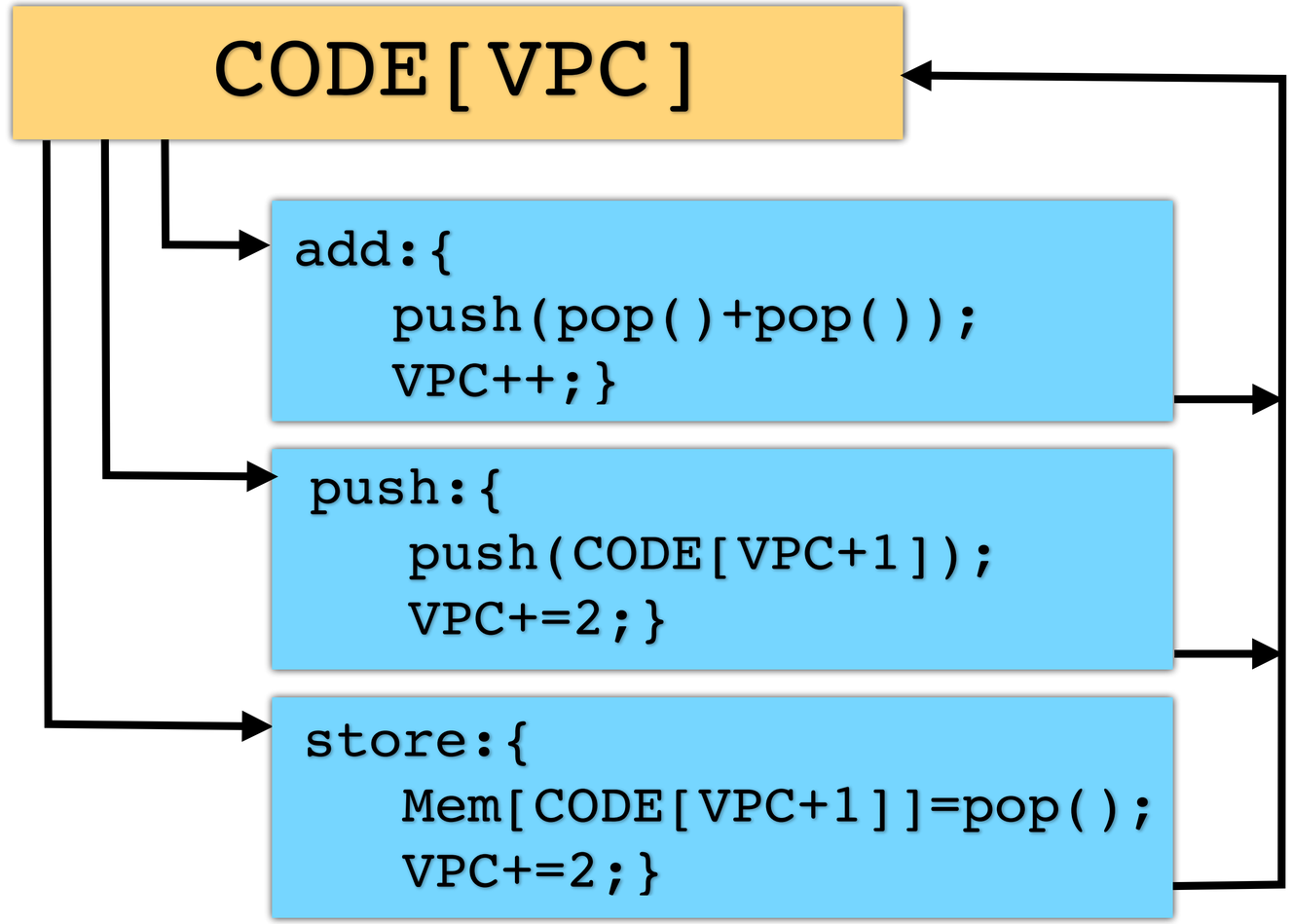
push 20	push 22	add	store &x




```
main() {
    int x;
    x = 20 + 22;
}
```

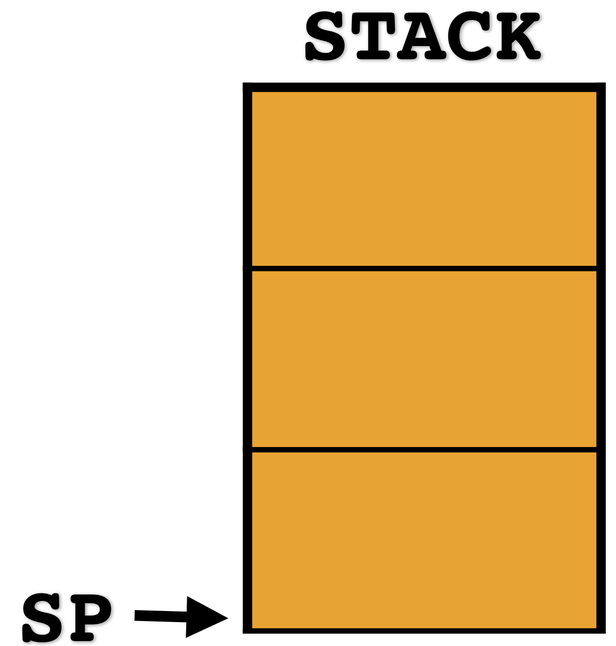
x: 99

push 20	push 22	add	store &x

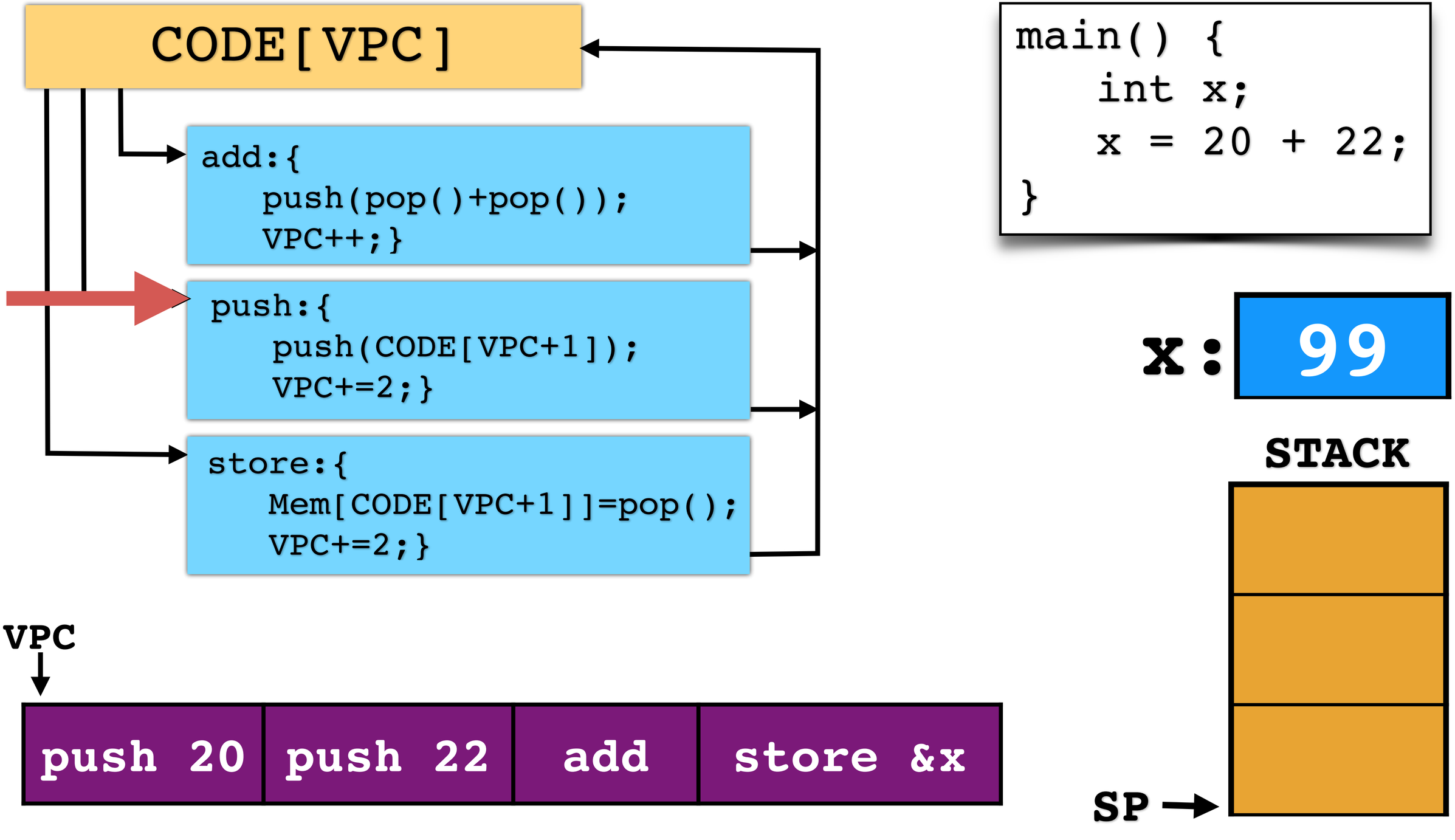


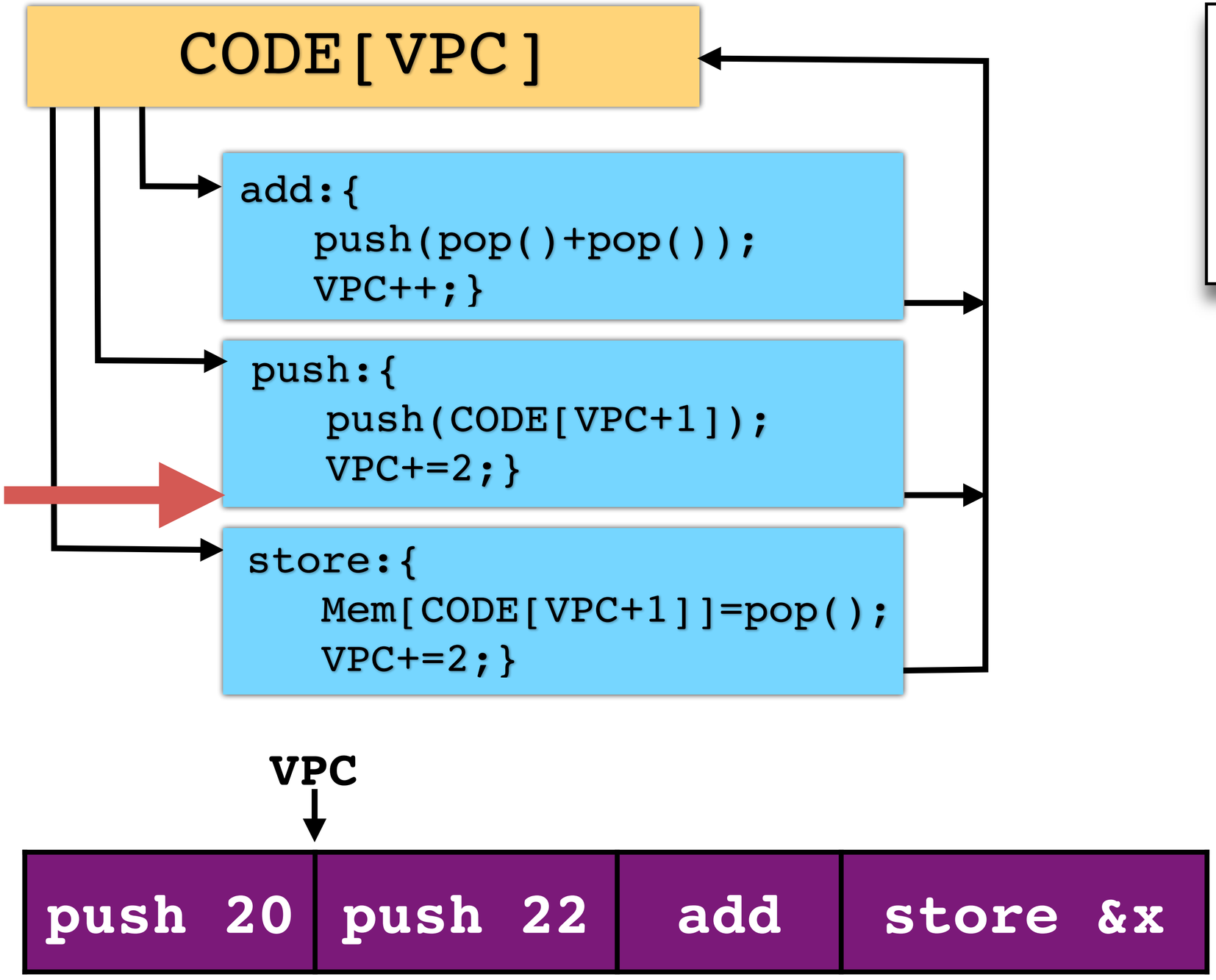

```
main() {
    int x;
    x = 20 + 22;
}
```

x: 99



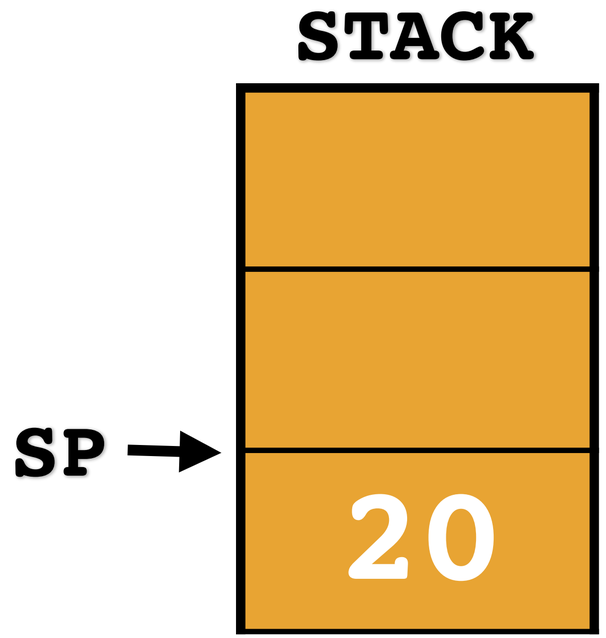
push 20	push 22	add	store &x

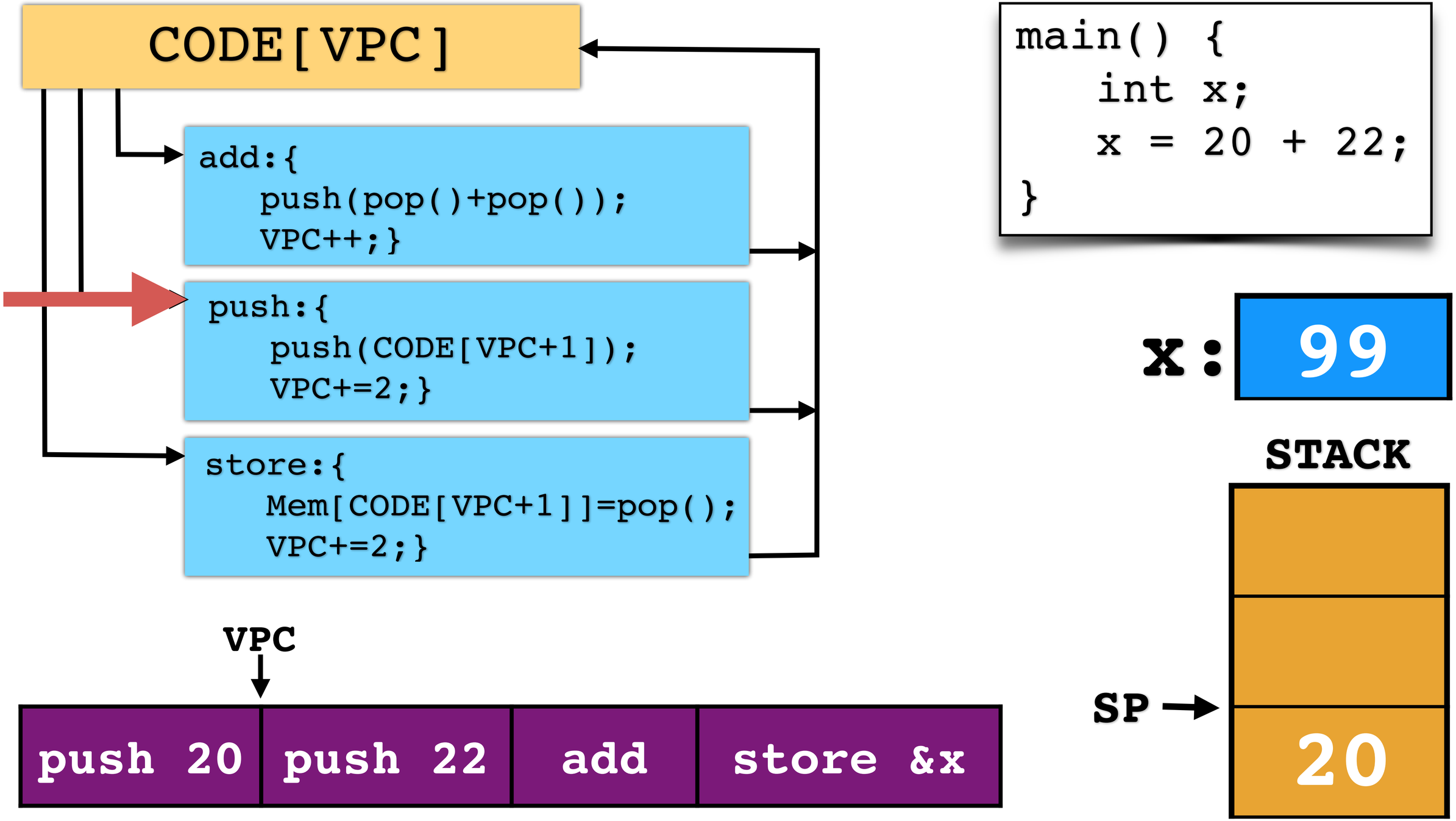

CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x: 99
STACK
VPC
push 20
push 22
add
store &x
SP

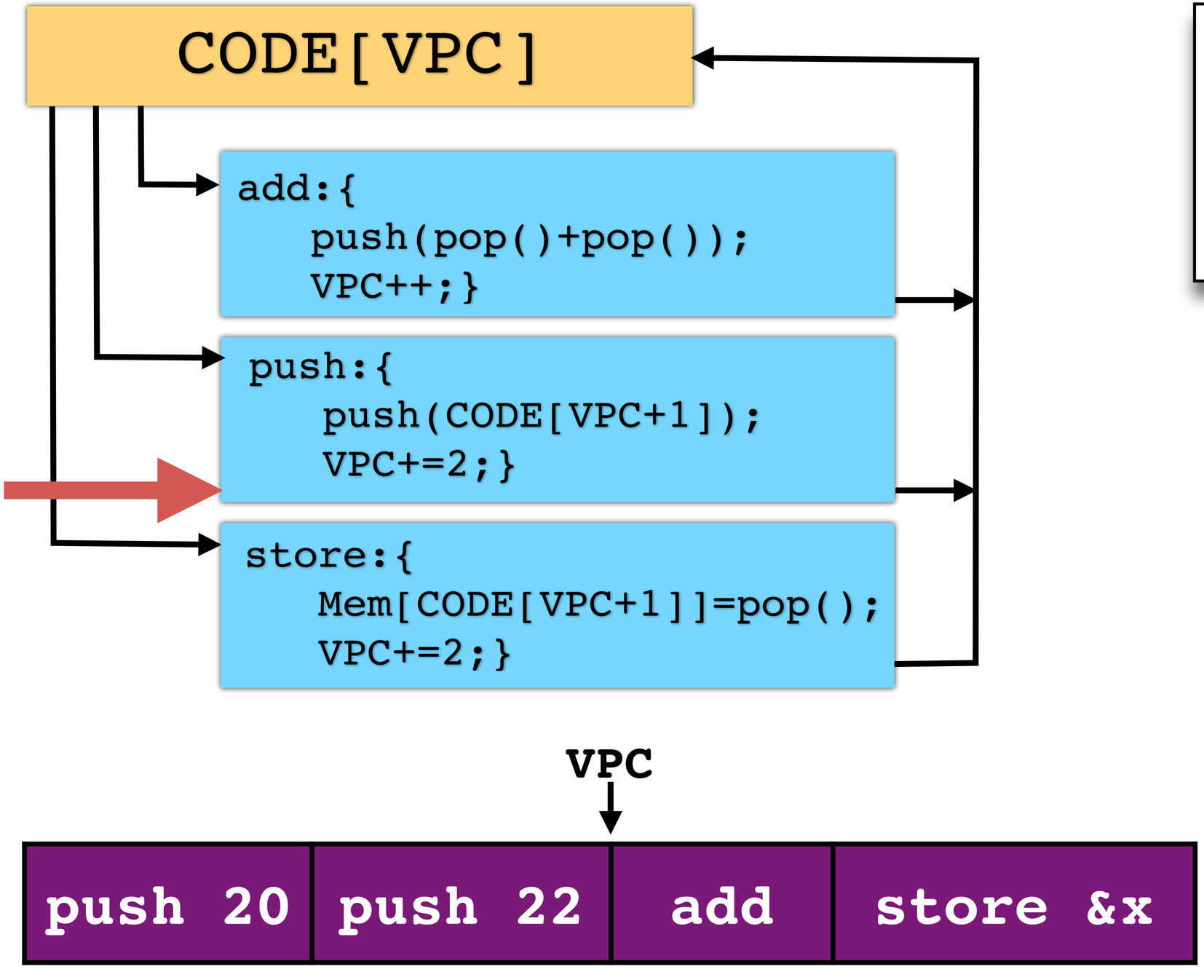



```
main() {
    int x;
    x = 20 + 22;
}
```

x: 99

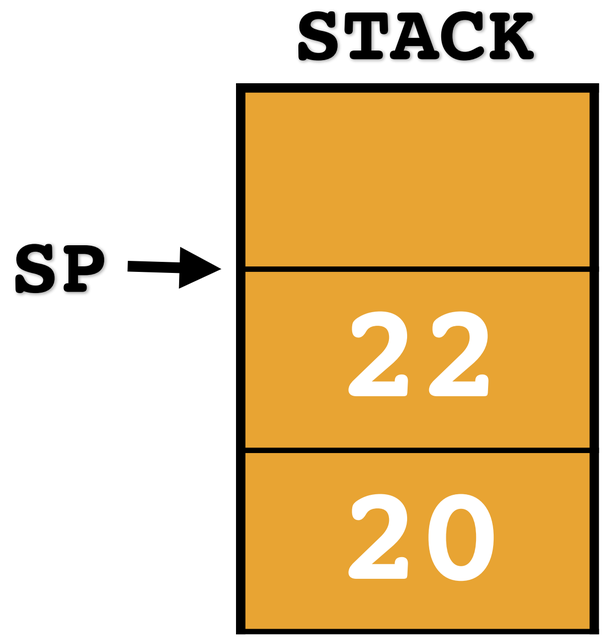


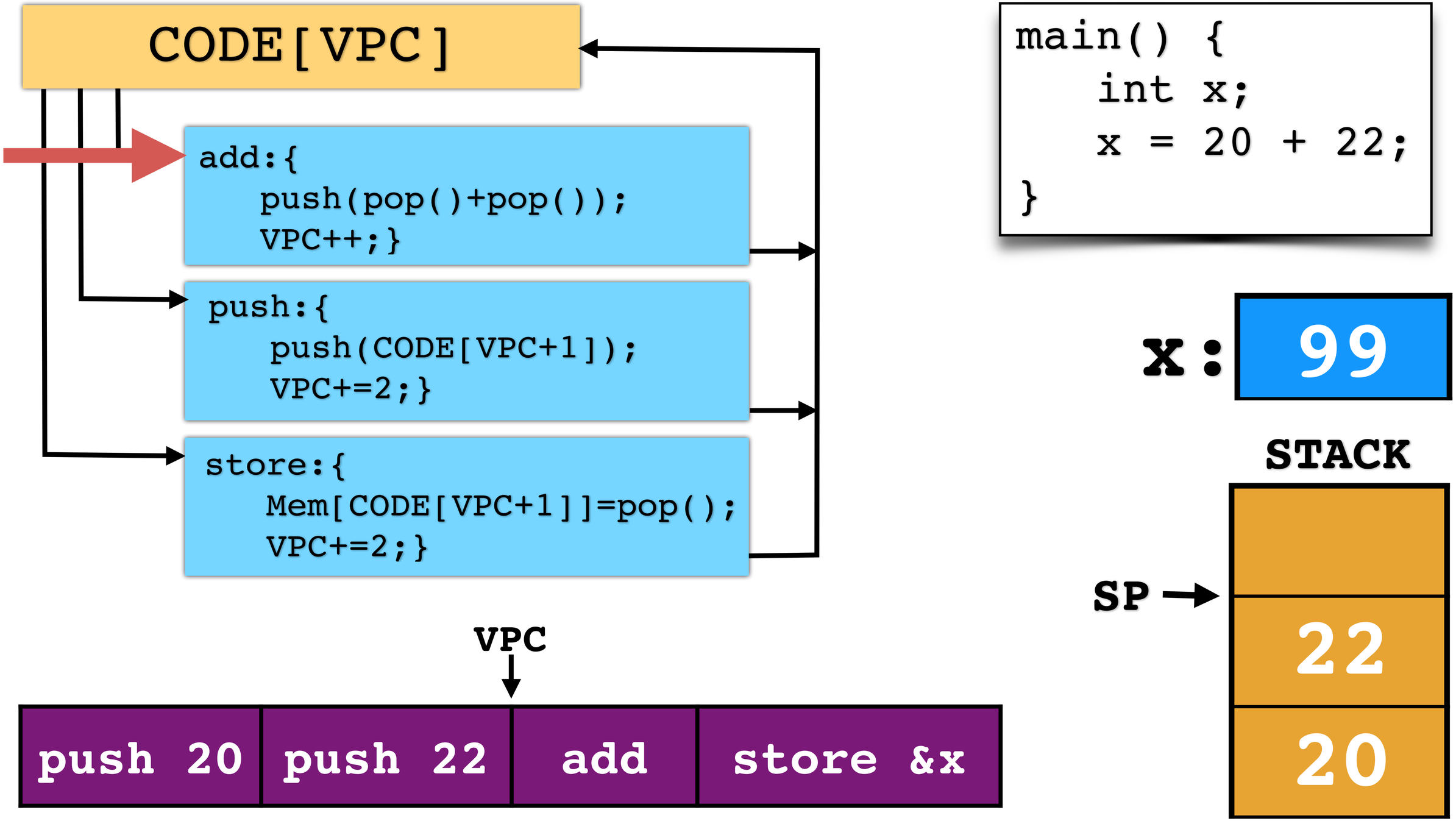

CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x:
99
STACK
SP
20
VPC
push 20
push 22
add
store &x

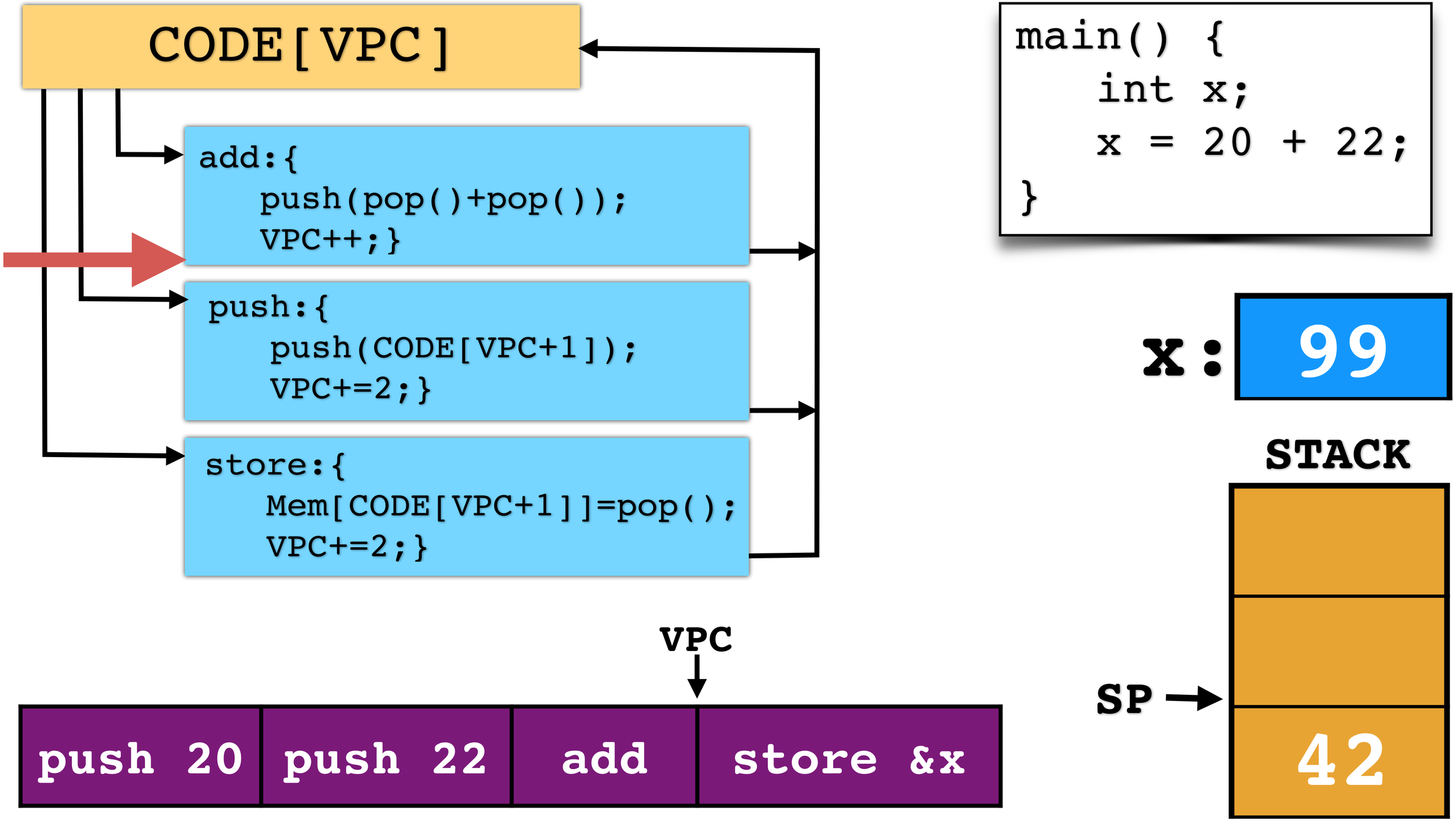


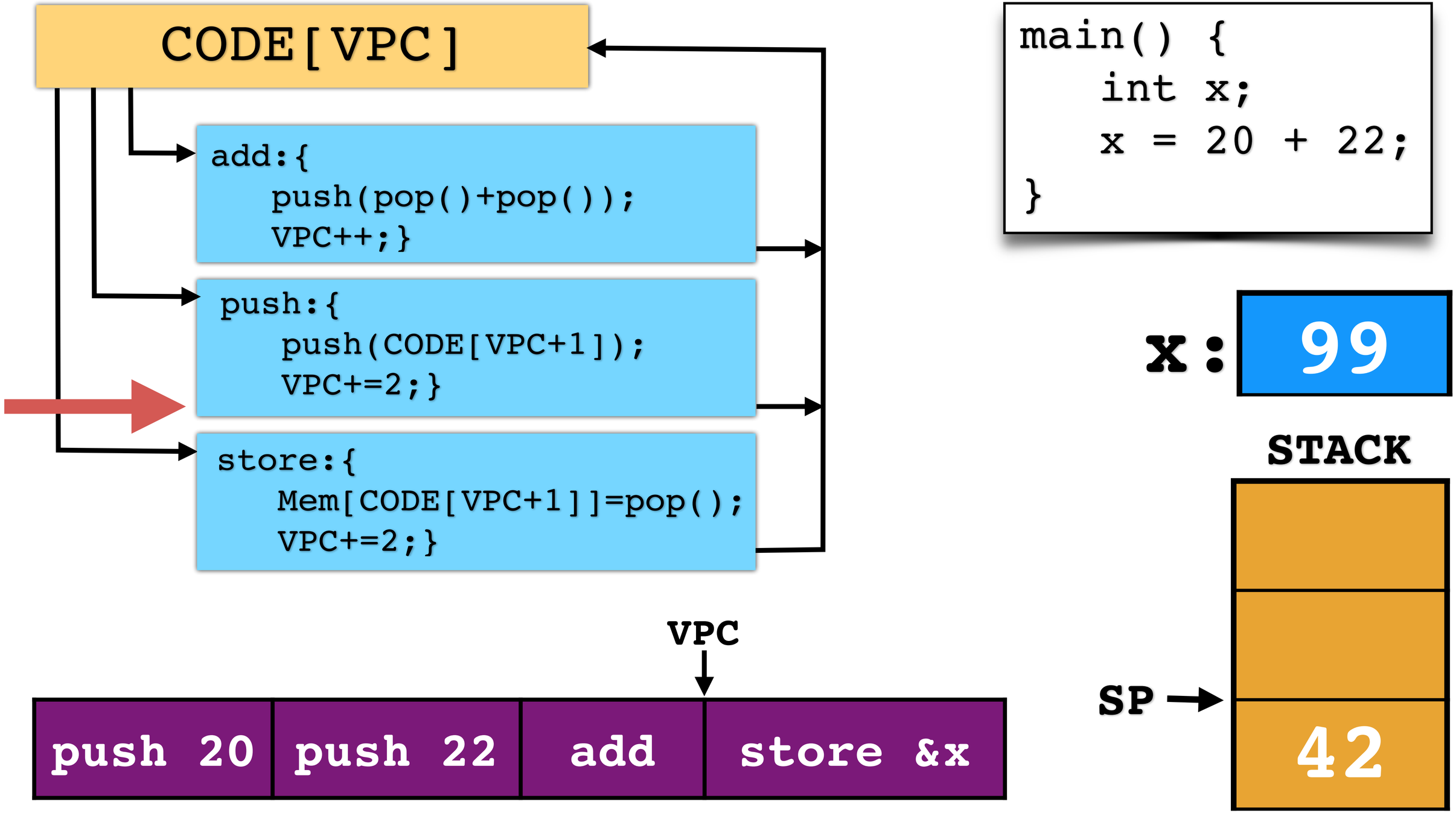

```
main() {
    int x;
    x = 20 + 22;
}
```

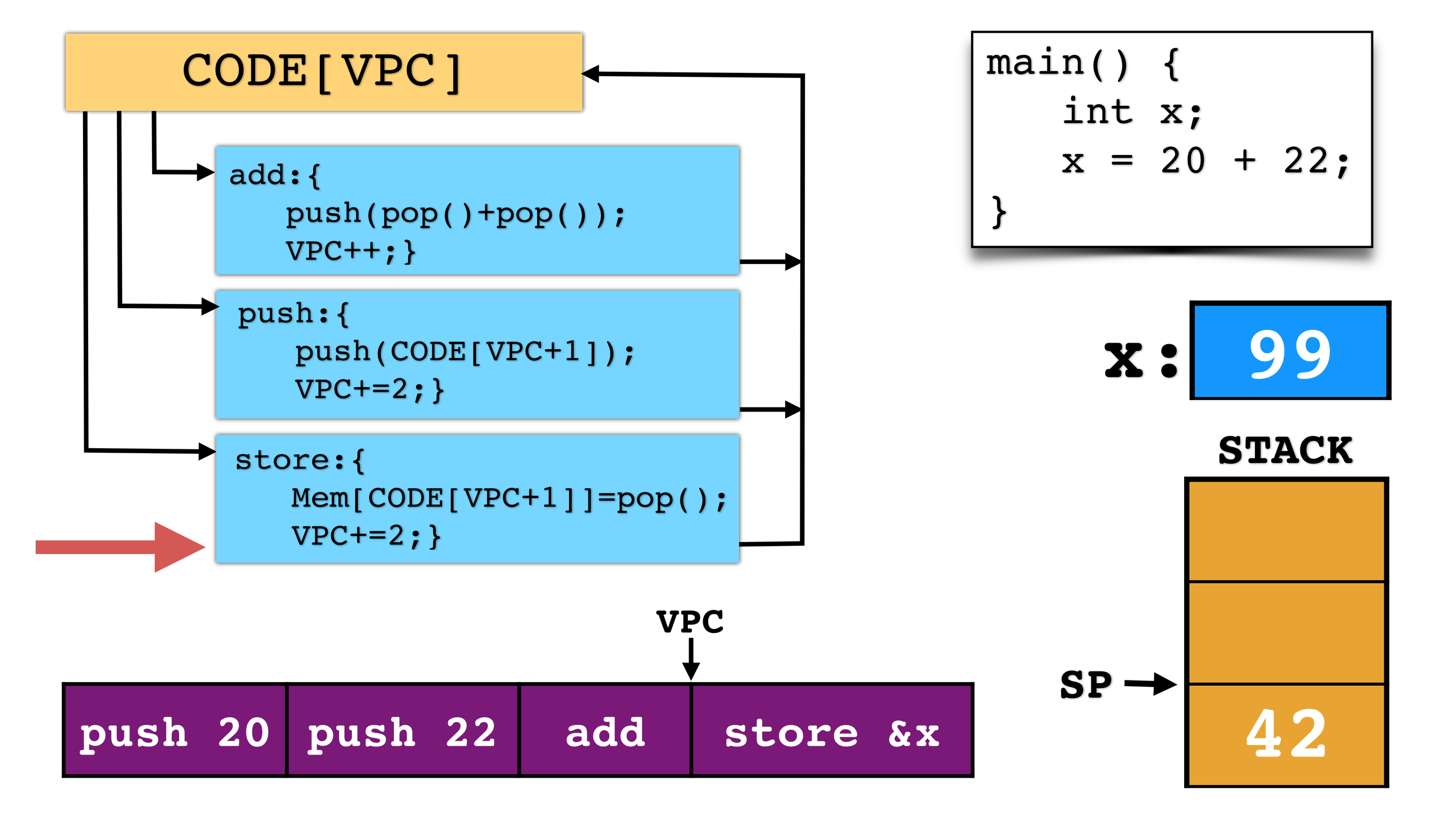
x: 99

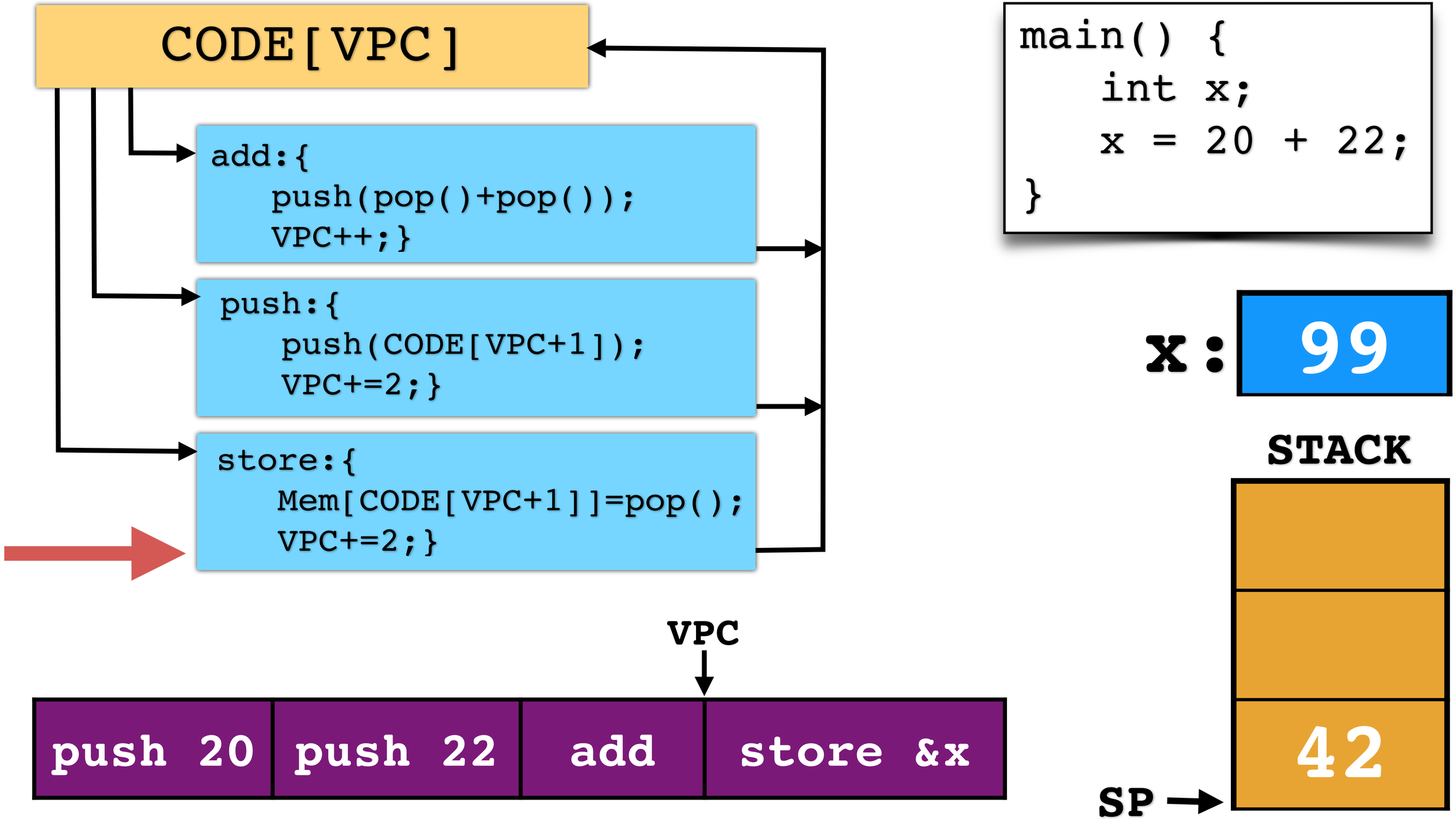


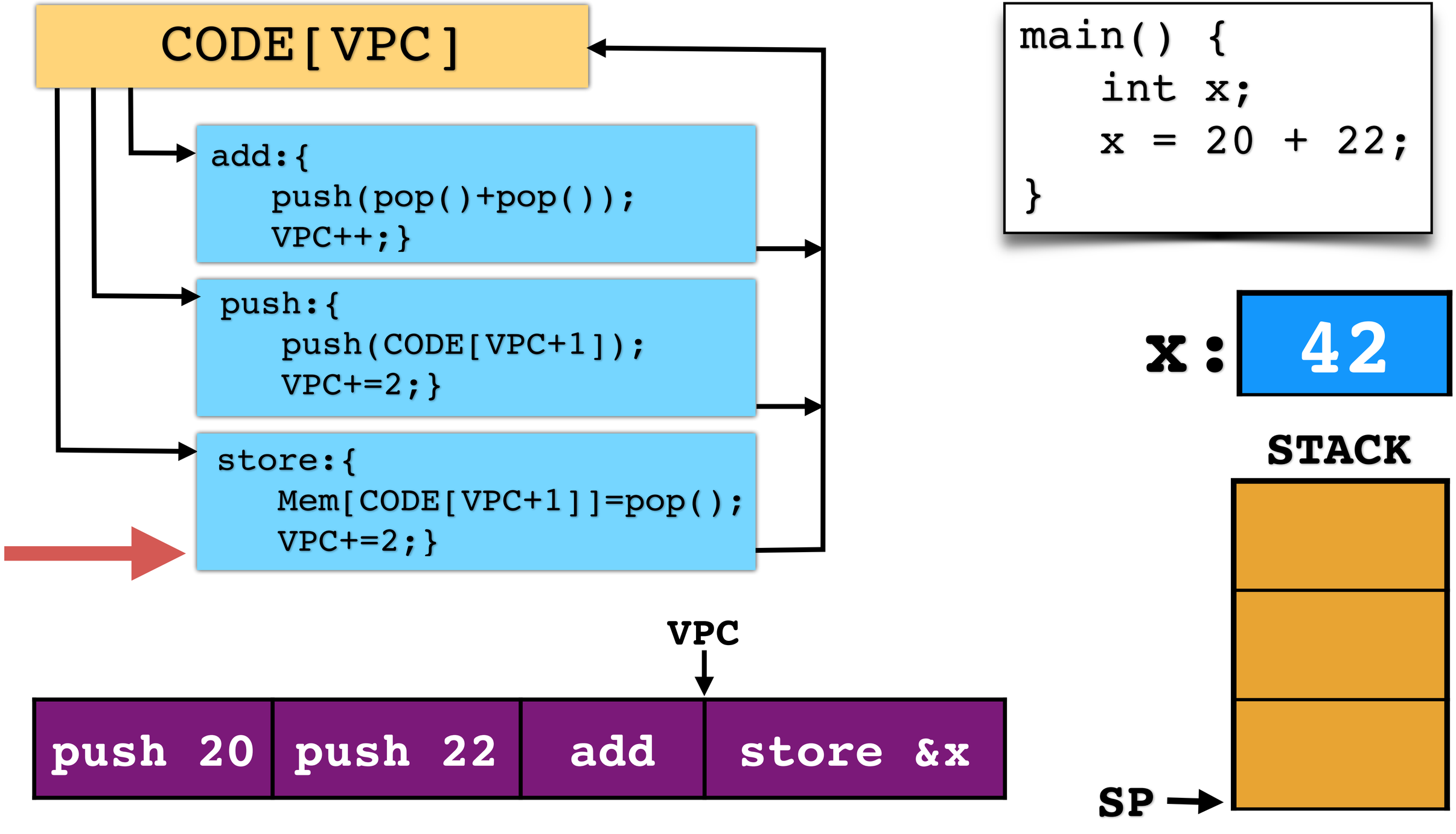

CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x:
99
STACK
SP
22
20
VPC
push 20
push 22
add
store &x

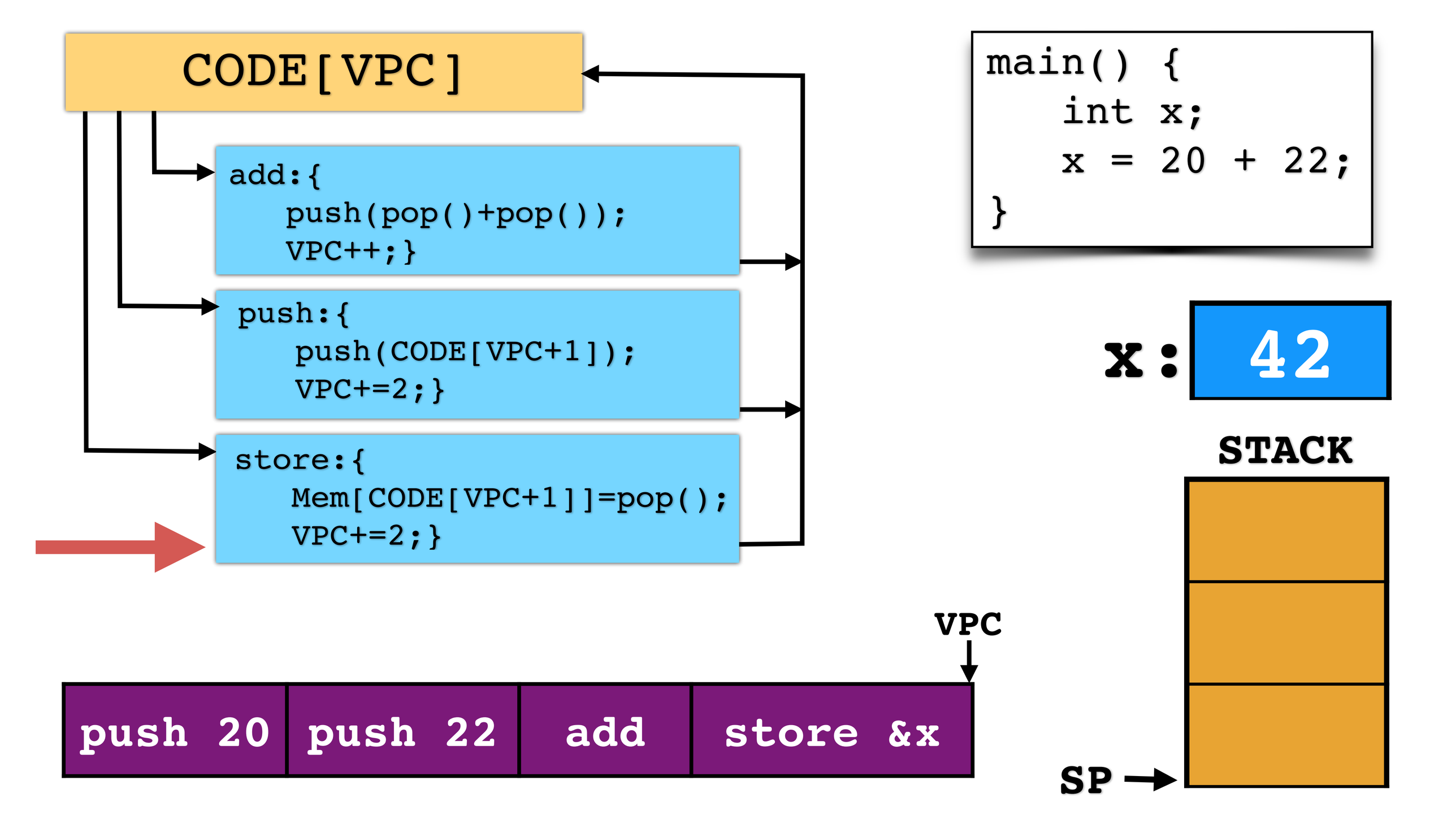

CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x:
99
STACK
SP
42
VPC
push 20
push 22
add
store &x


CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x:
99
STACK
SP
42
VPC
push 20
push 22
add
store &x


CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x:
99
STACK
SP
42
VPC
push 20
push 22
add
store &x

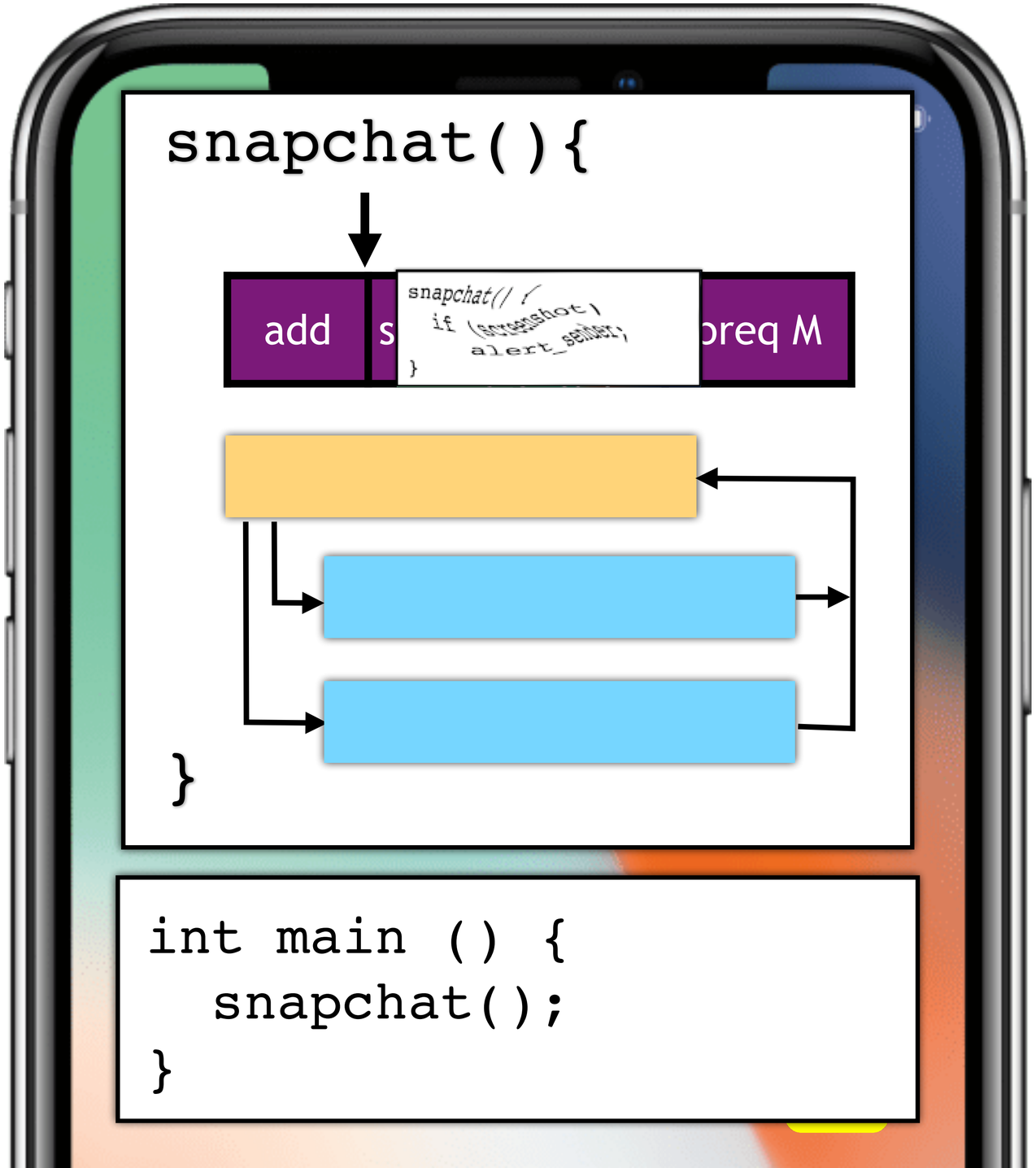

CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x:
99
STACK
42
SP
VPC
push 20
push 22
add
store &x


CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x: 42
STACK
VPC
push 20
push 22
add
store &x
SP

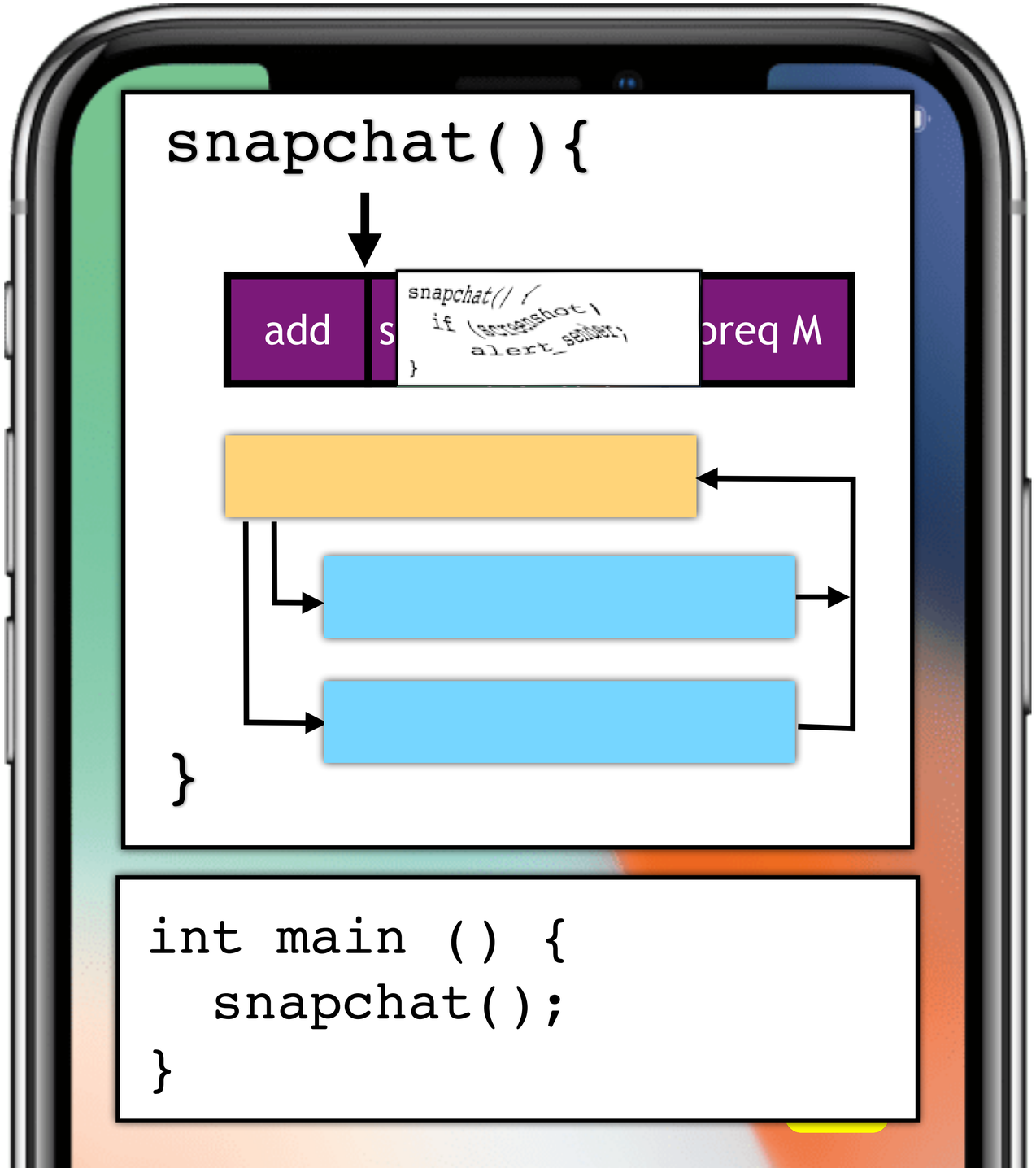

CODE[VPC]
add:{
push(pop()+pop());
VPC++;}
push:{
push(CODE[VPC+1]);
VPC+=2;}
store:{
Mem[CODE[VPC+1]]=pop();
VPC+=2;}
main() {
int x;
x = 20 + 22;
}
x: 42
STACK
VPC
push 20
push 22
add
store &x
SP

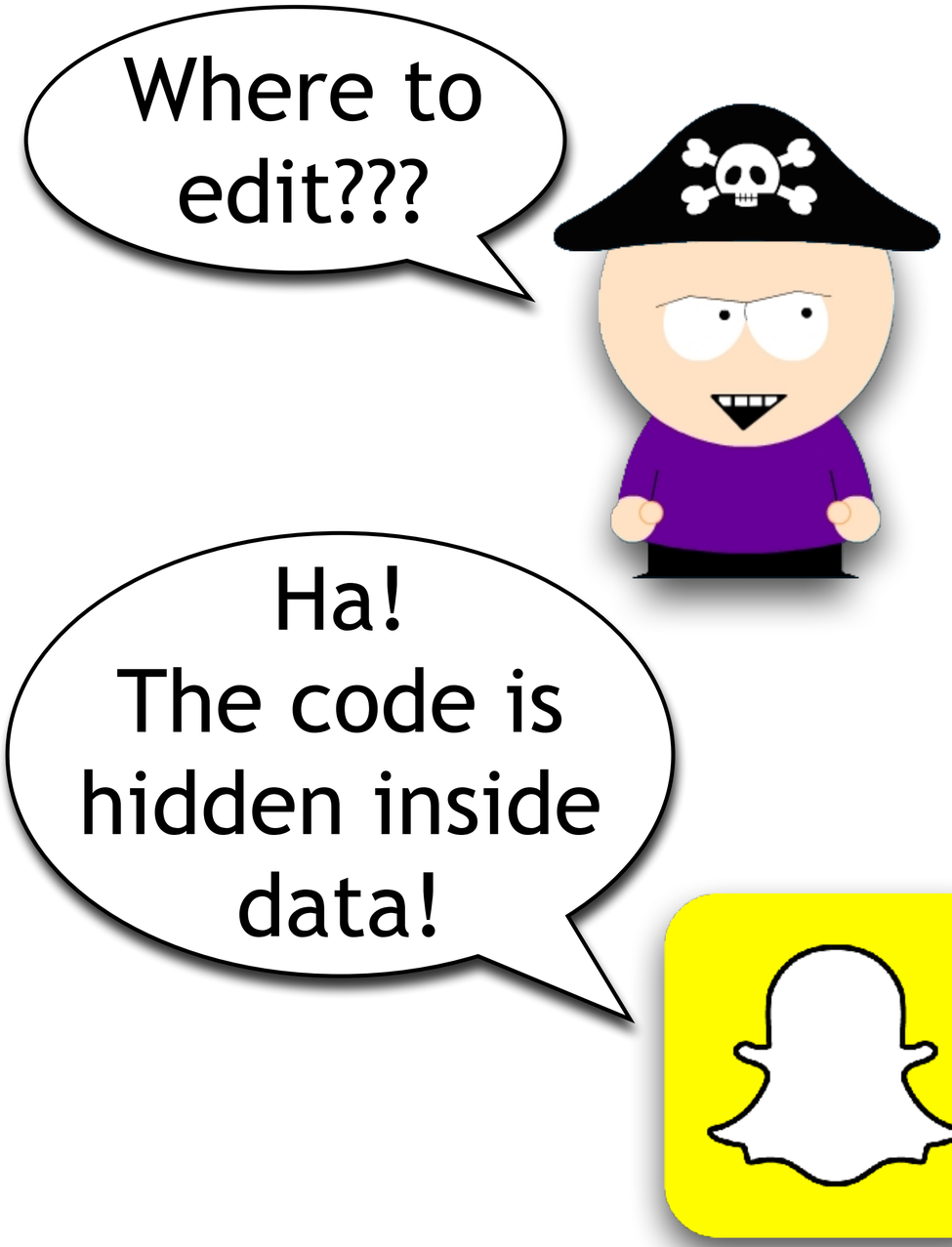

snapchat() {
if (screenshot)
alert_sender;
}
9:41
Tuesday, September 12
int main () {
snapchat();
}


Where to edit???


snapchat(){
add
s
snapchat(){
if (screenshot)
alert_sender;
}
breq M
}
int main () {
snapchat();
}


Where to edit???

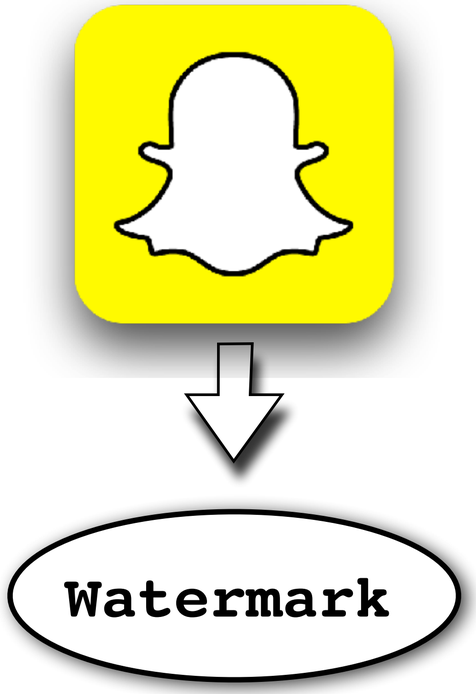

snapchat(){
add
}
int main () {
snapchat();
}


Where to edit???
Ha! The code is hidden inside data!

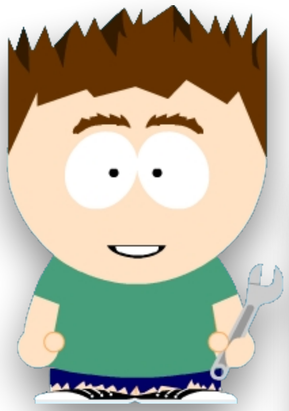
Watermarking

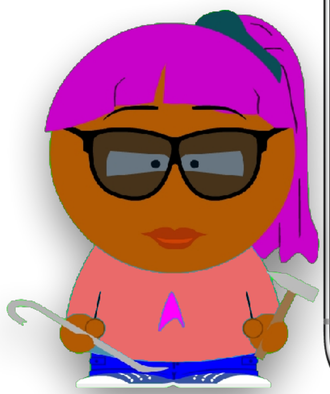
Algorithms

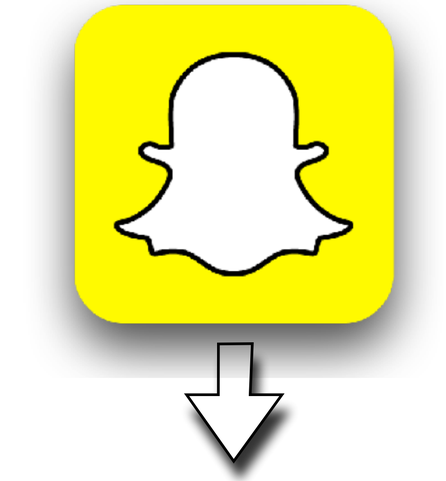
Watermarking Algorithms



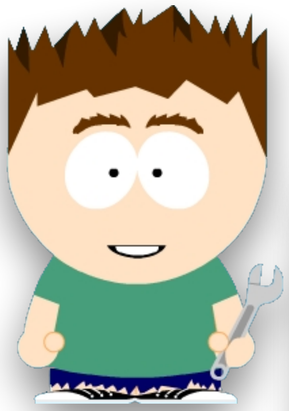
Watermark

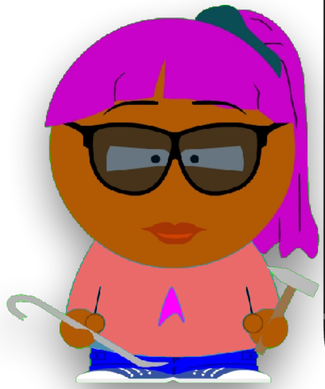


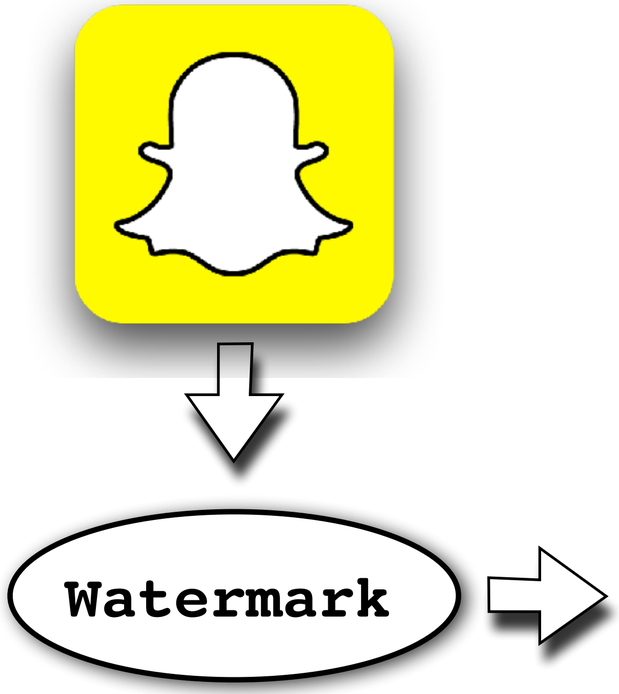




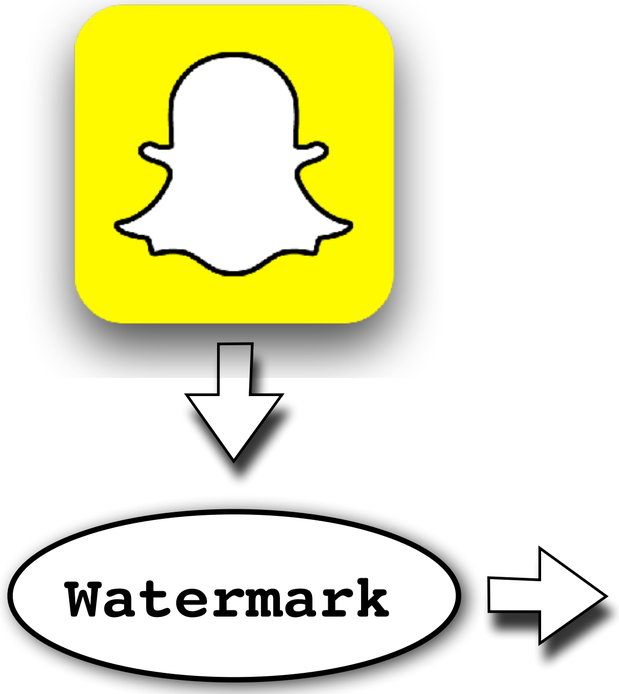
Watermark

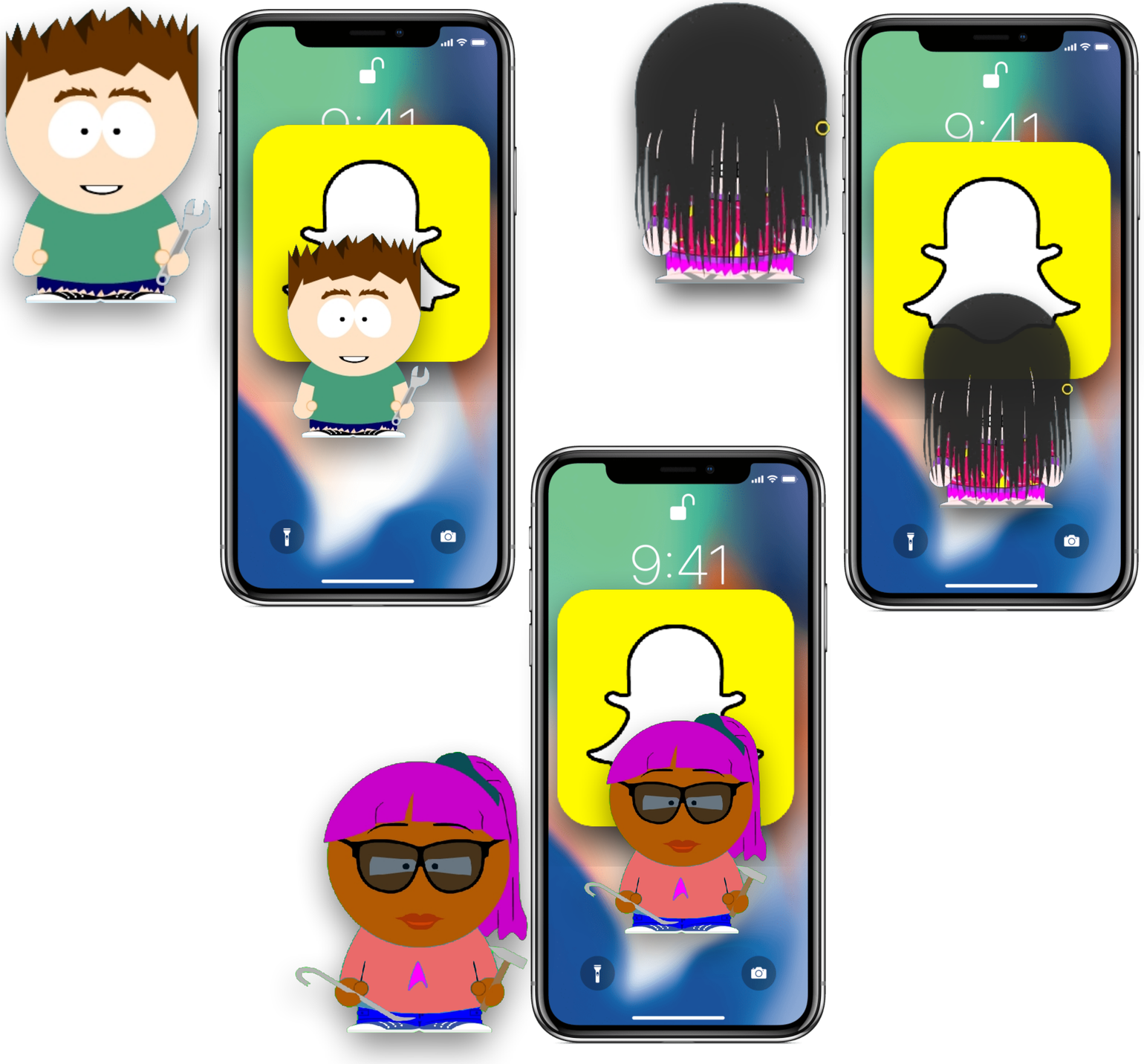


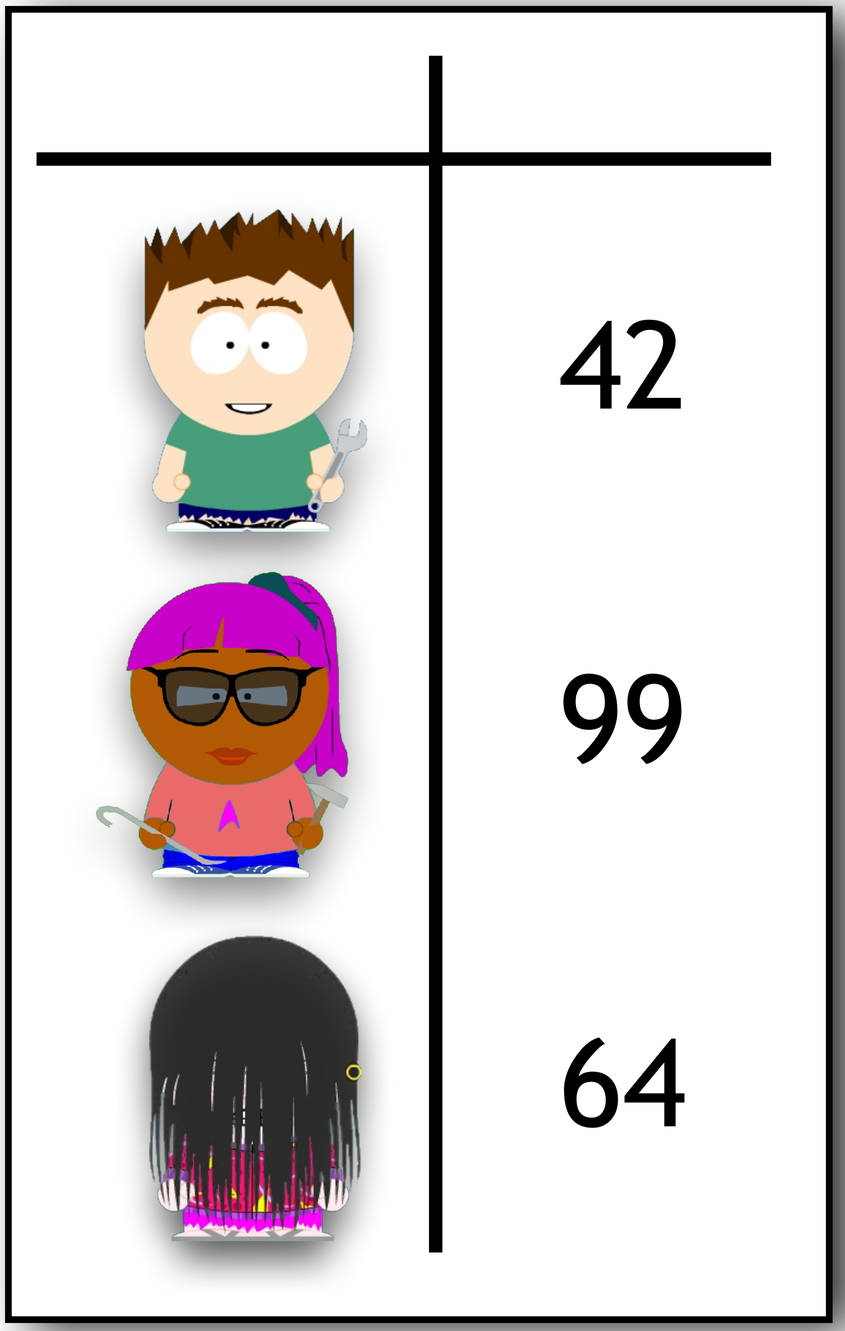


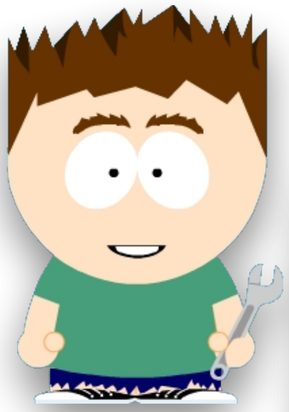

Watermark


9:41
Tuesday, September 12
9:41
9:41


Watermark

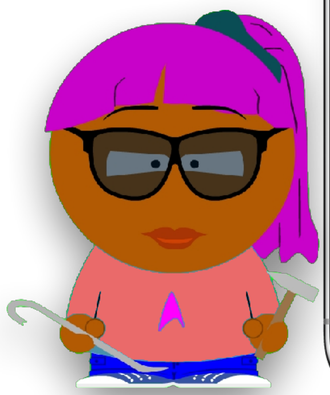



42
99
64



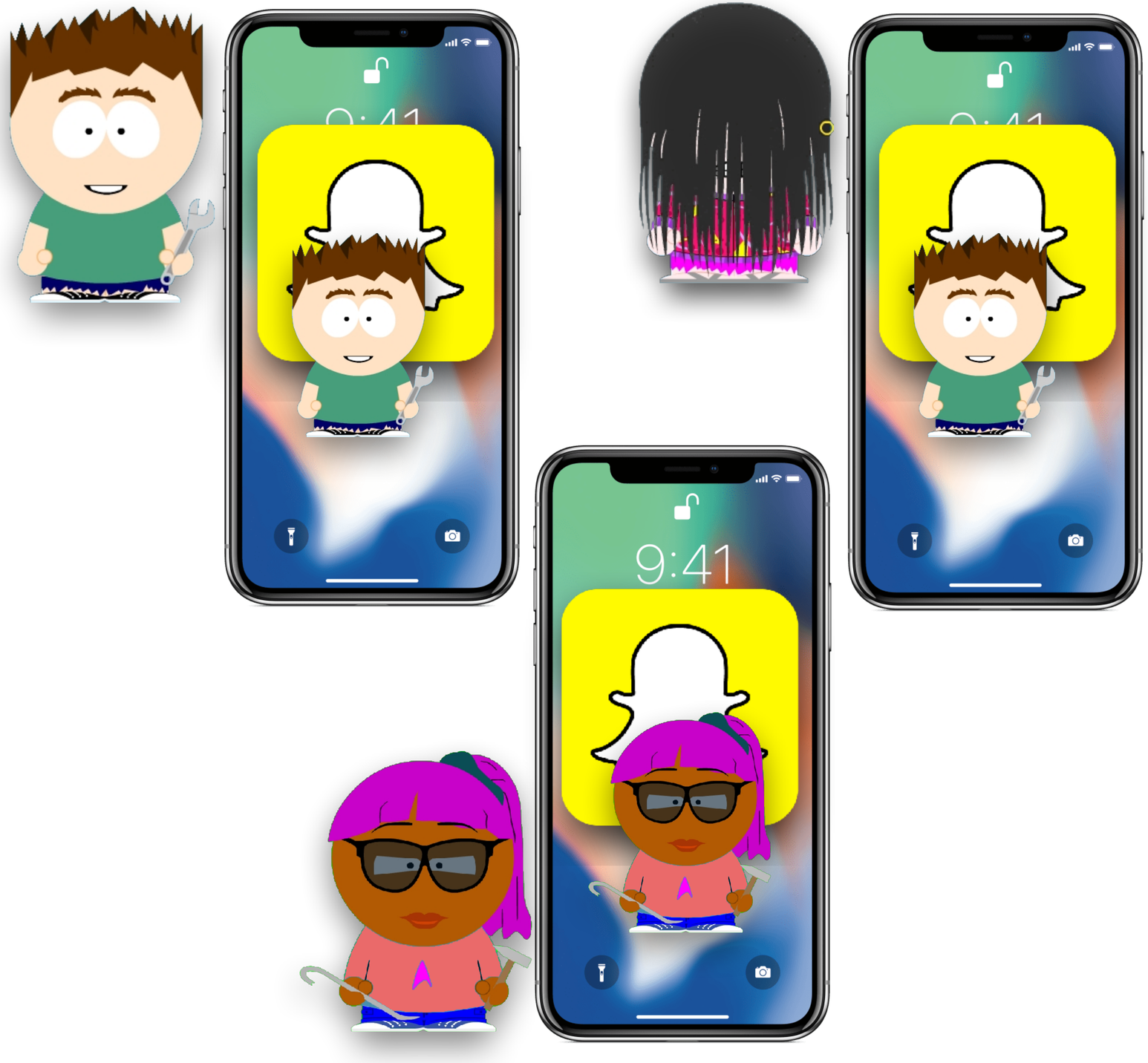

9:41


9:41

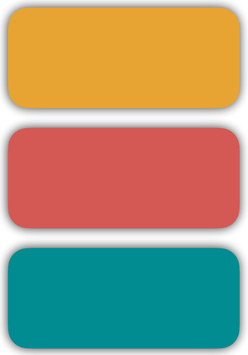



9:41

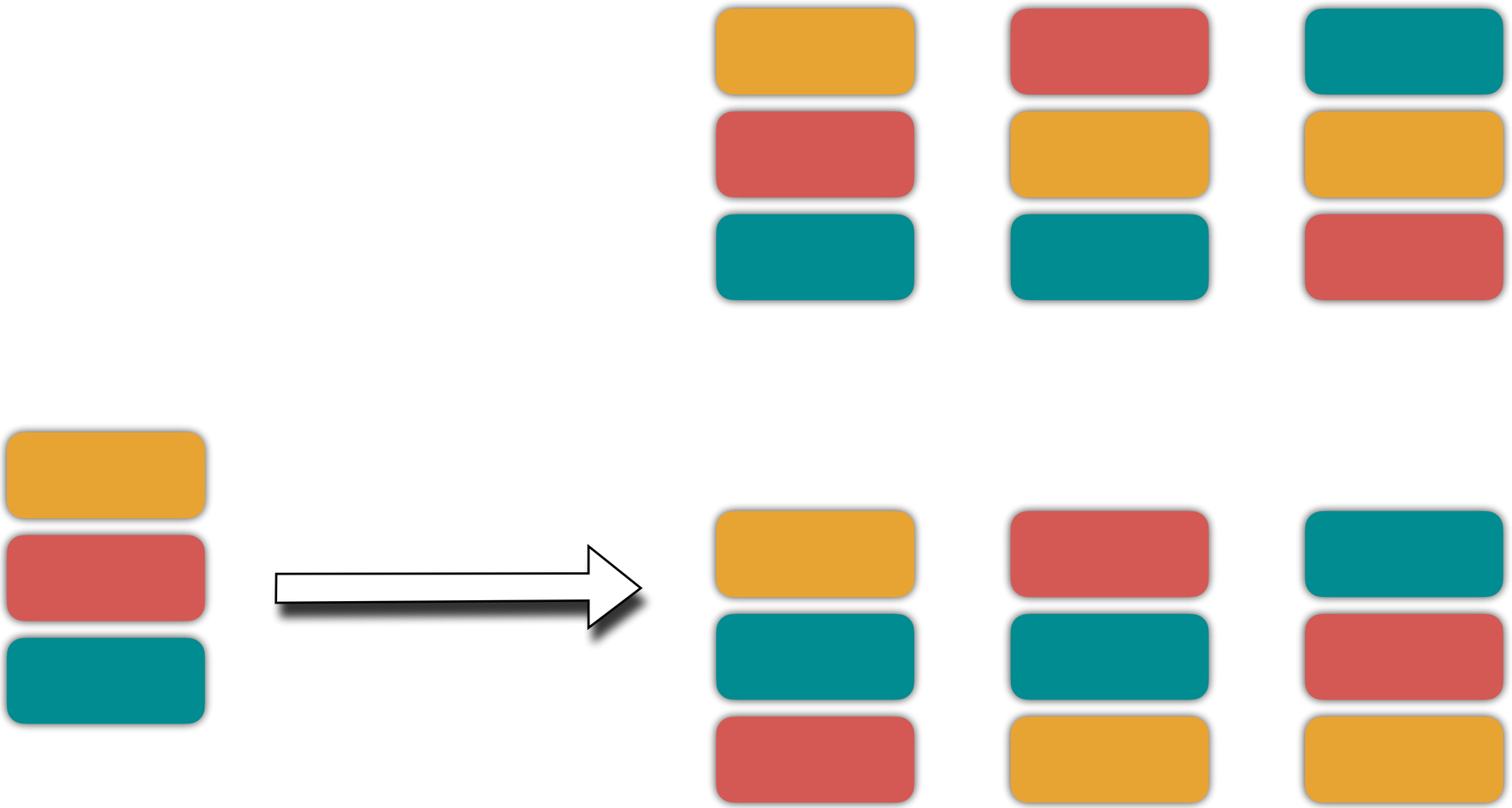

9:41
9:41
Tuesday, September 12
9:41


9:41
9:41

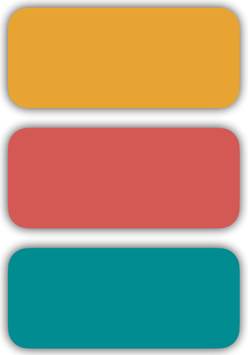
Permutation Watermarks



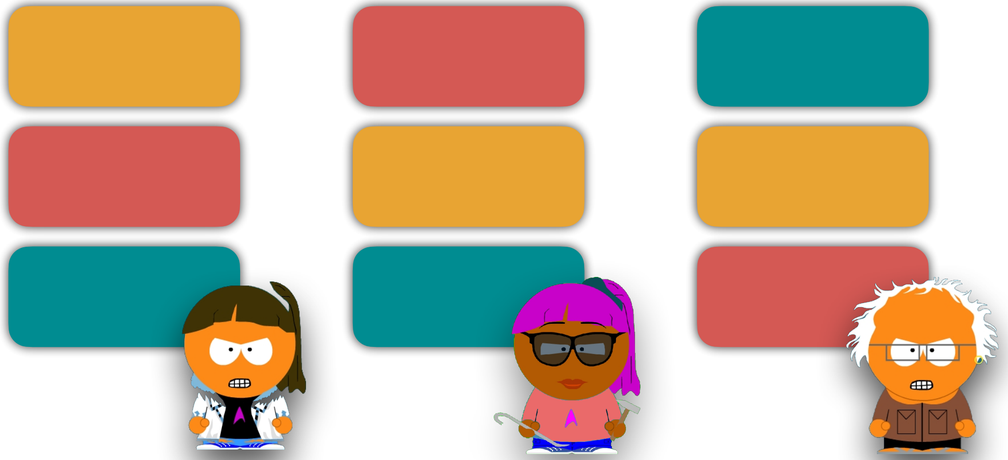
Permutation Watermarks

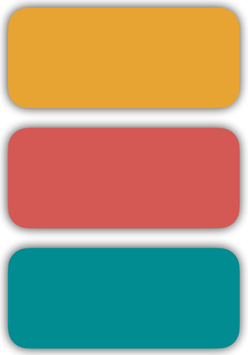


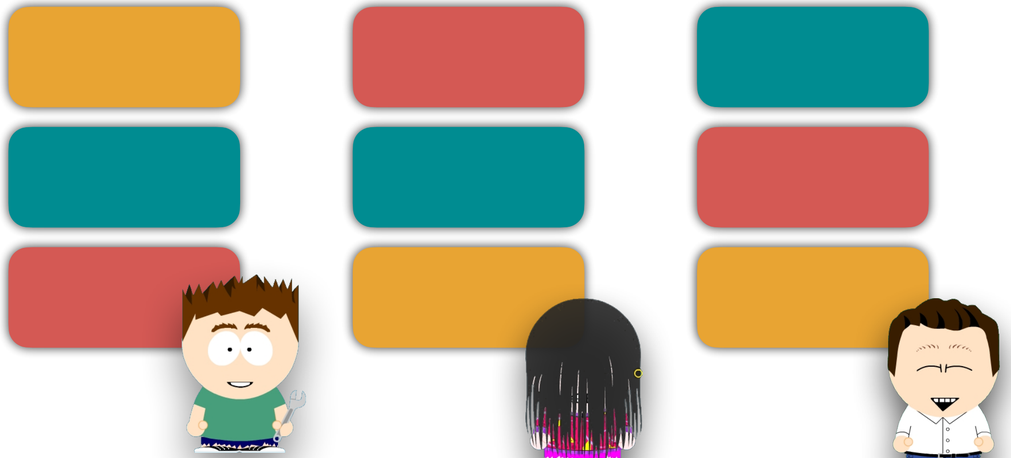
Permutation Watermarks



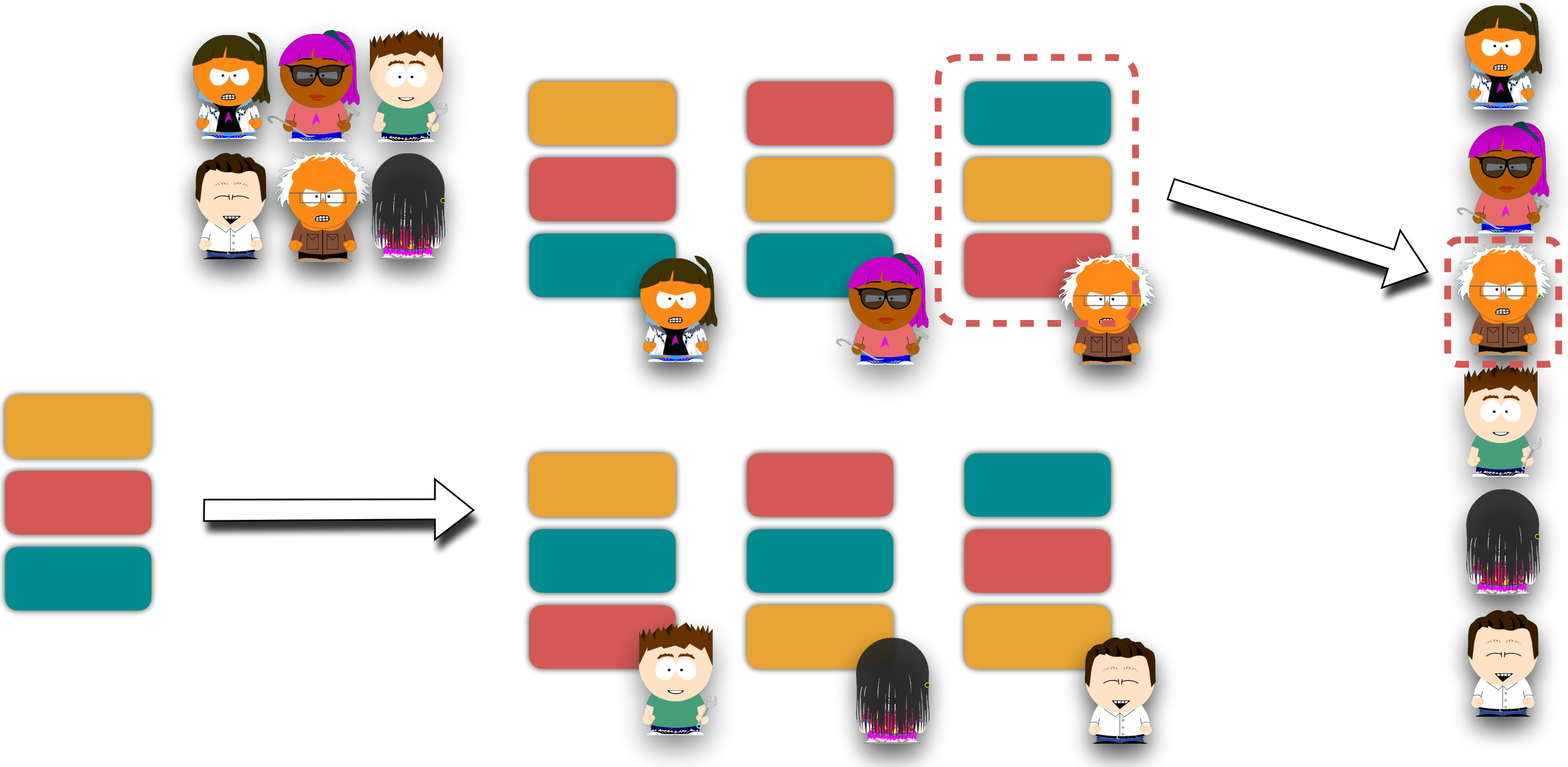
Permutation Watermarks







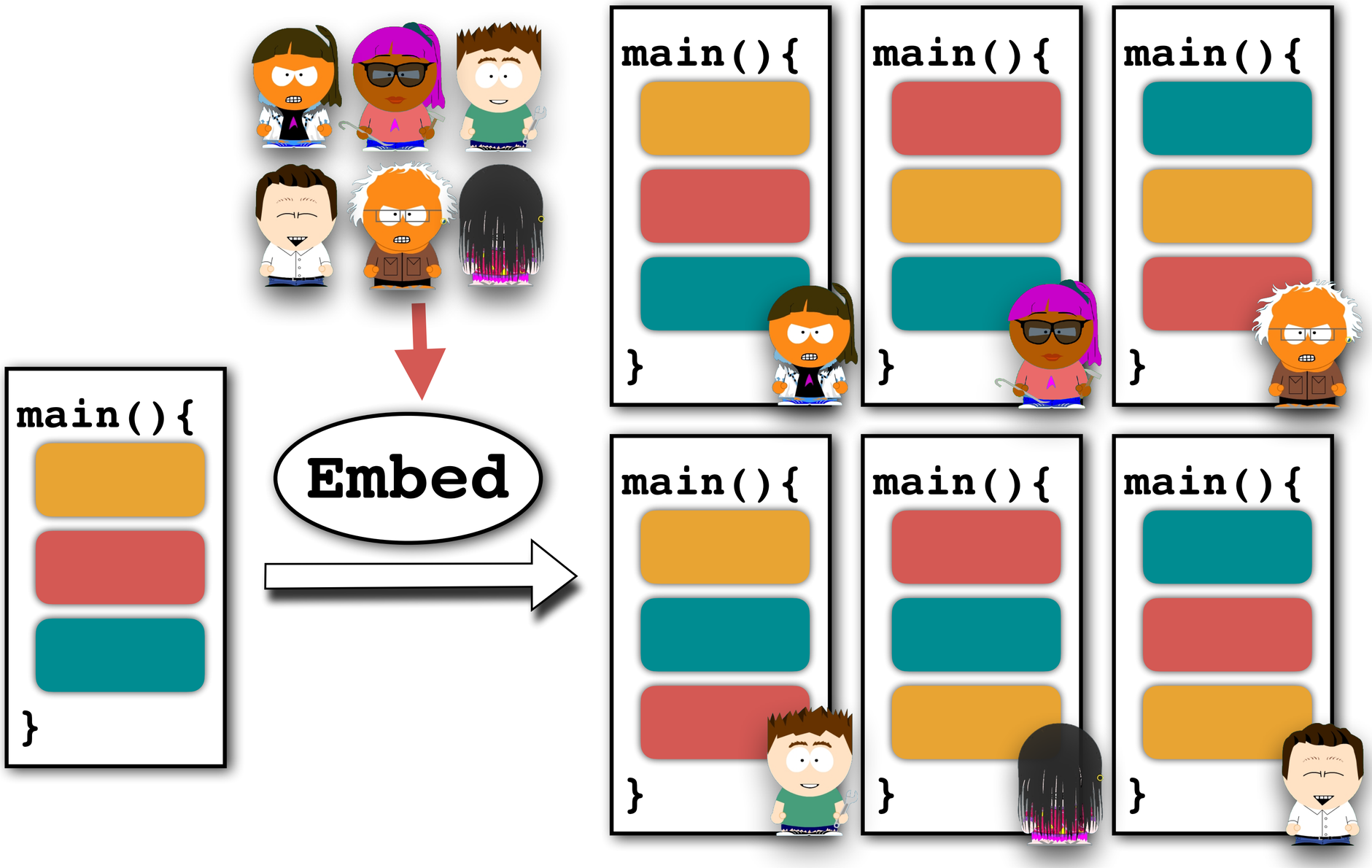
Permutation Watermarks



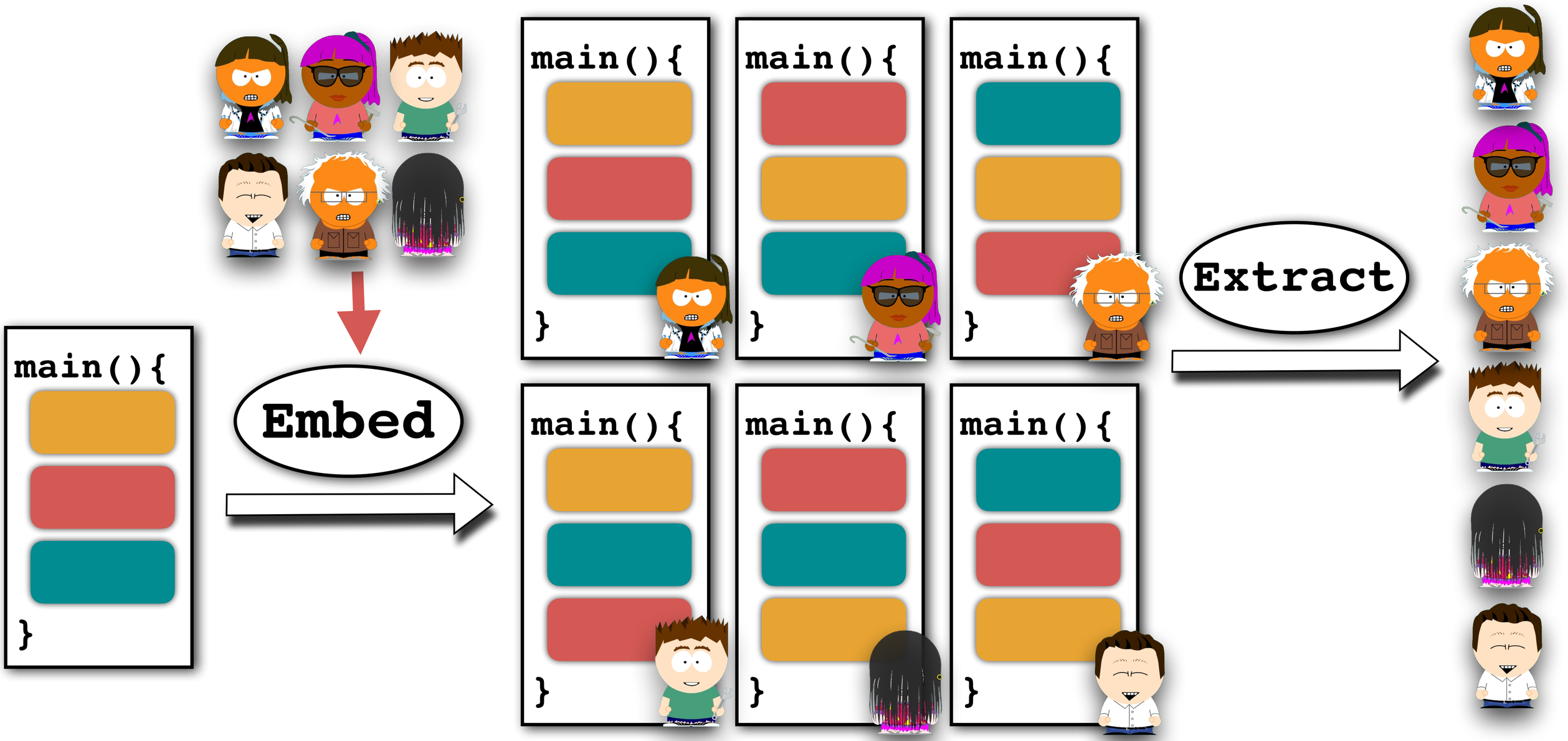
Permuting Statements

```
main(){
  [ ]
  [ ]
  [ ]
}
```

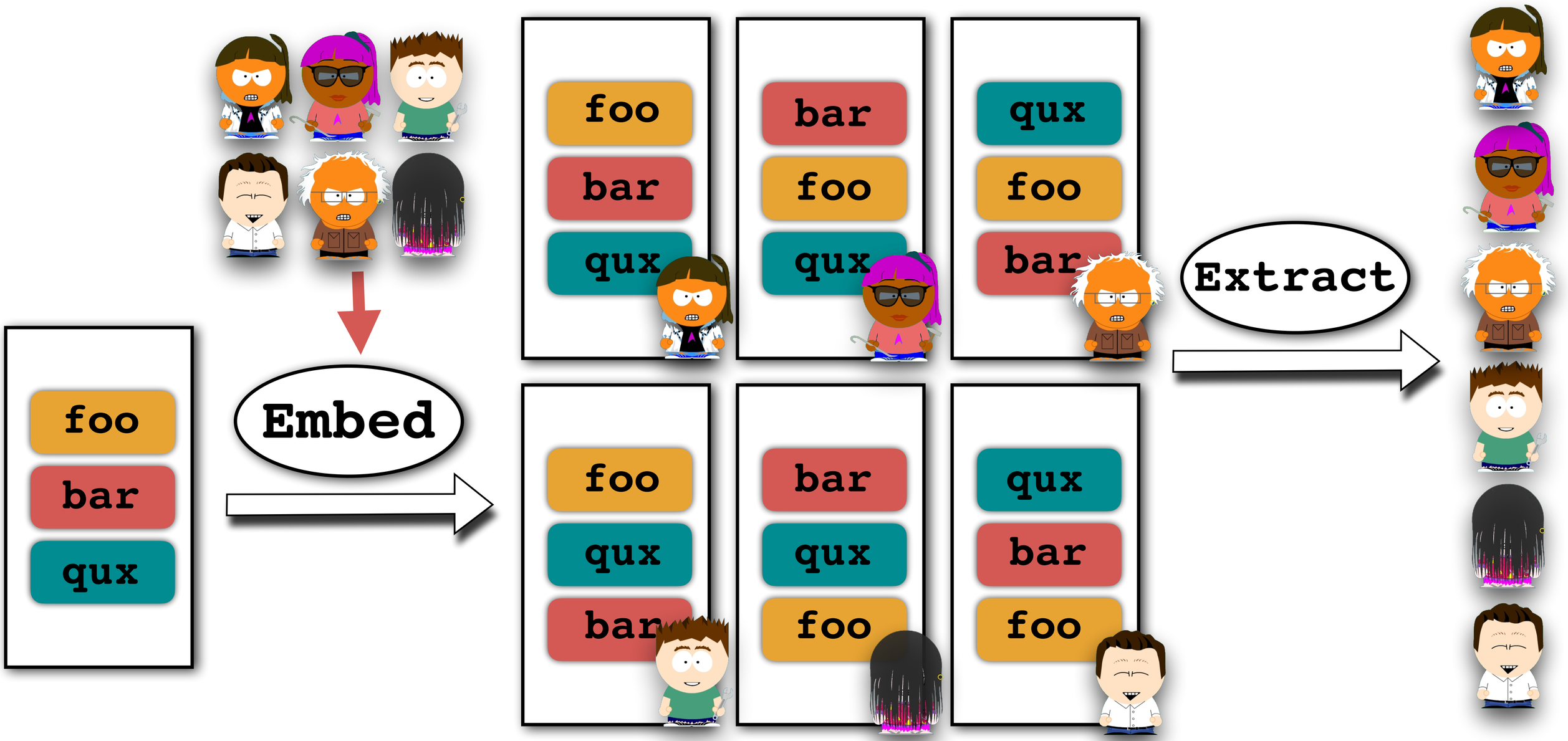
Permuting Statements



Permuting Statements



Permuting Functions



Case Study 1:

Projector

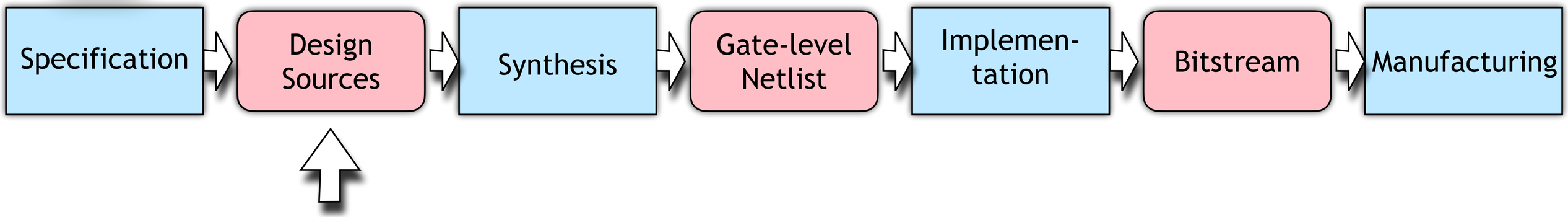
Hardware

Supply Chains

Case Study 1: Protecting Hardware Supply Chains

Hardware Design Flow

Hardware Design Flow

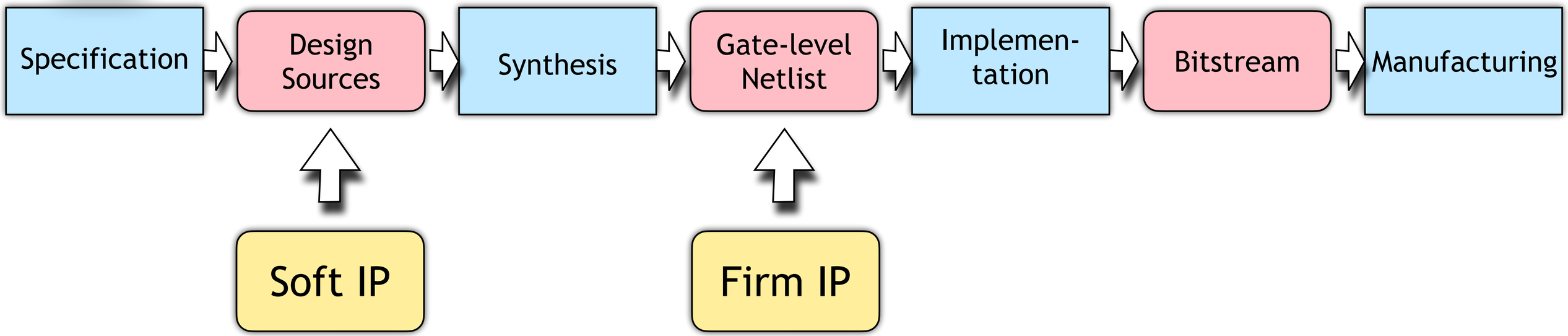


Soft IP

```
26   process (aclr, clk)
27   begi
28     (a lr = '1') then
29       q  <= others => '0');
30     el f r ing ed (clk) then
31       q_s <  (' &si ed(a)) + (
32     end if, -- clk'
33   end process;
34
35 end signed_adder_arch;
```

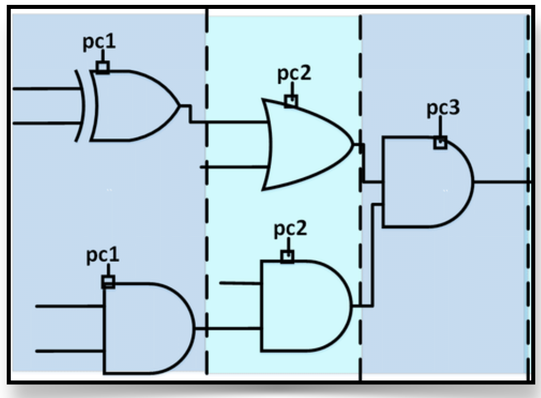
VHDL

Hardware Design Flow




```
26   process (aclr, clk)
27   begi
28     (a lr = '1') then
29       q  <= others => '0');
30     el f r ing ed (clk) then
31       q_s <  (' &si ed(a)) + (
32     end if, -- clk'
33   end process;
34
35 end signed_adder_arch;
```

VHDL



Hardware Design Flow

Specification → Design Sources → Synthesis → Gate-level Netlist → Implementation → Bitstream → Manufacturing

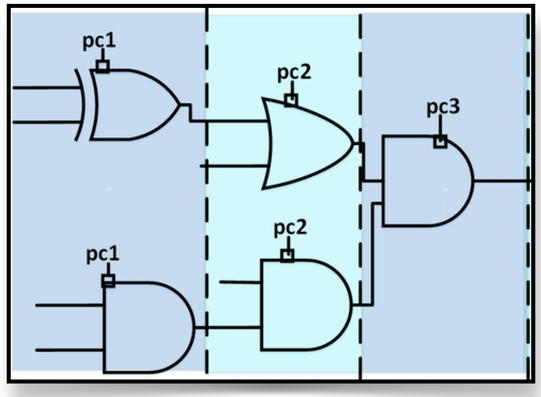
Soft IP

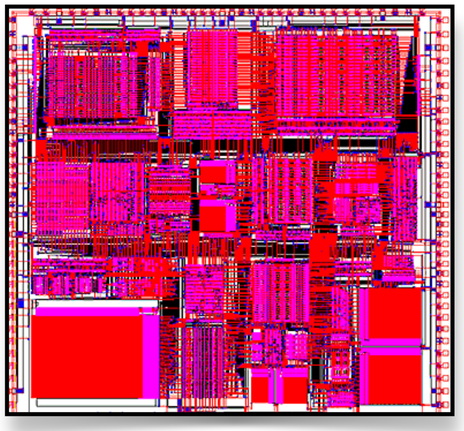
Firm IP

Hard IP

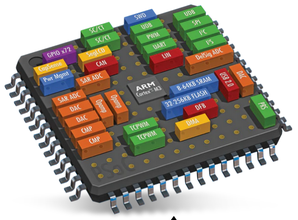
```
26   process (aclr, clk)
27   begi
28     (aclr = '1') then
29       q   <= others => '0');
30     elsif rising_edge(clk) then
31       q_s <= ('0' &si ed(a)) + (
32     end if; --clk'
33   end process;
34
35 end signed_adder_arch;
```

VHDL





Hardware Design Flow



SoC,
FPGA, ASIC

Specification → Design Sources → Synthesis → Gate-level Netlist → Implementation → Bitstream → Manufacturing

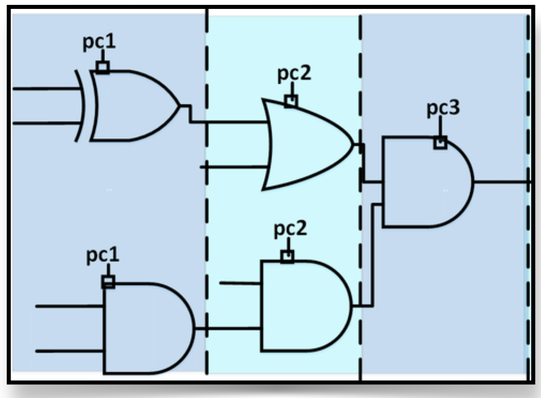
Soft IP

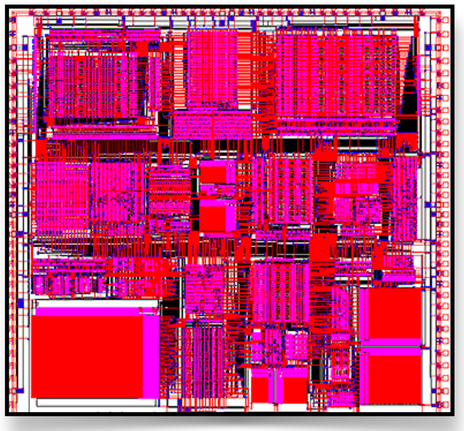
Firm IP

Hard IP

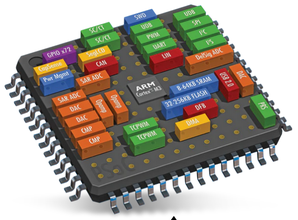
```
26   process (aclr, clk)
27   begi
28     (aclr = '1') then
29       q <= others => '0');
30     elsif rising_edge(clk) then
31       q_s <= ('0' &signed(a)) + (
32     end if; -- clk'
33   end process;
34
35 end signed_adder_arch;
```

VHDL





EDA Tools

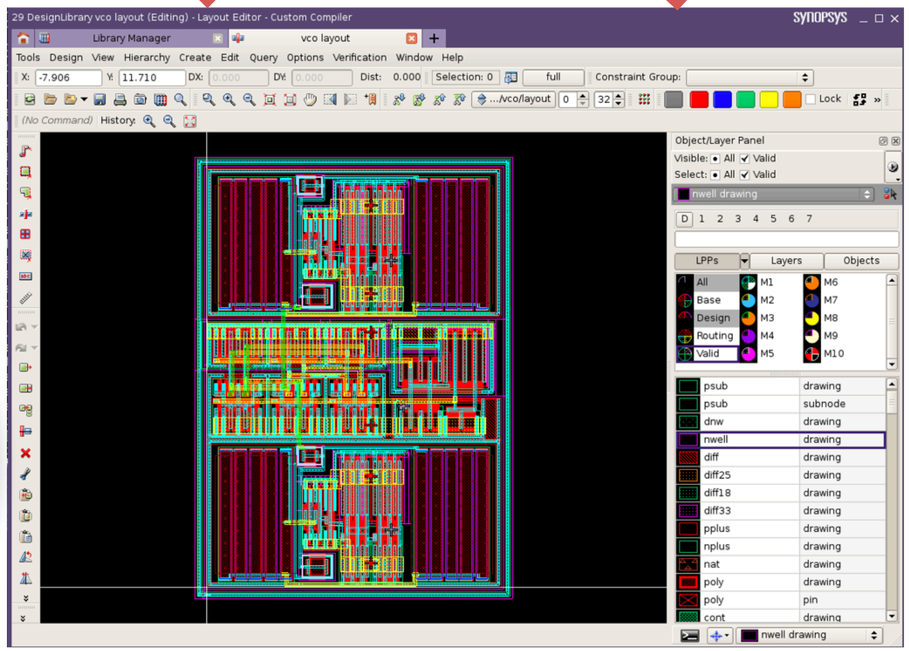


SoC,
FPGA, ASIC

Specification → Design Sources → Synthesis → Gate-level Netlist → Implemen-tation → Bitstream → Manufacturing

Electronic Design and Automation Tool Vendors

- Cadence
- Synopsys
- Xilinx
- Siemens



3rd Party Developers

Specification → Design Sources → Synthesis → Gate-level Netlist → Implemen-tation → Bitstream → Manufacturing

IP Core Author

3rd Party Developers

IP Core Author

Specification → Design Sources → Synthesis → Gate-level Netlist → Implemen-tation → Bitstream → Manufacturing

Soft IP Core → Design Sources

3rd Party Developers

Specification → Design Sources → Synthesis → Gate-level Netlist → Implementation → Bitstream → Manufacturing

Soft IP Core → Design Sources

Firm IP Core → Gate-level Netlist

IP Core Author

3rd Party Developers

Specification → Design Sources → Synthesis → Gate-level Netlist → Implemen-tation → Bitstream → Manufacturing

Soft IP Core → Design Sources

Firm IP Core → Gate-level Netlist

Hard IP Core → Bitstream

IP Core Author

3rd Party Developers

Specification → Design Sources → Synthesis → Gate-level Netlist → Implemen-tation → Bitstream → Manufacturing

IP Core Author

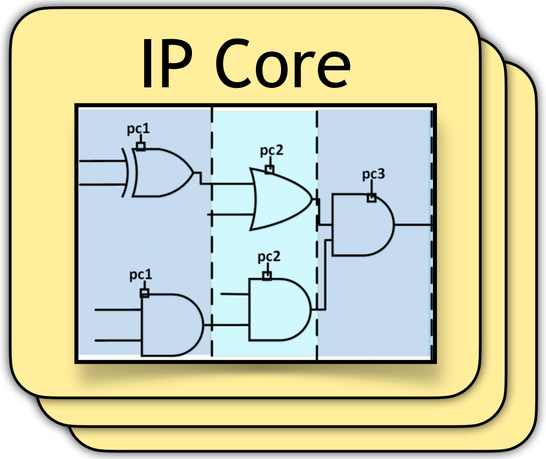
Soft IP Core

Firm IP Core

Hard IP Core

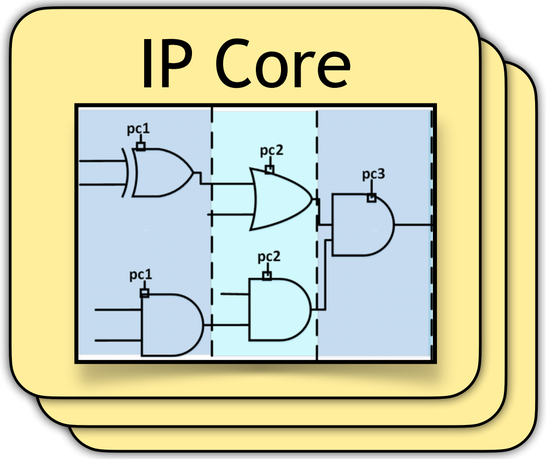
Supply Chain Stakeholders

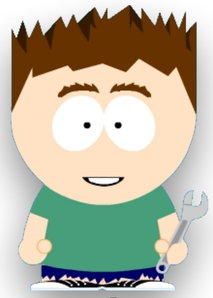
IP Core Author



Supply Chain Stakeholders

IP Core Author



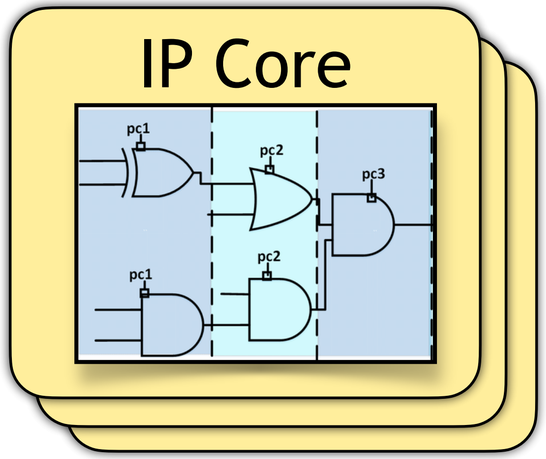


Developer

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

Supply Chain Stakeholders

IP Core Author

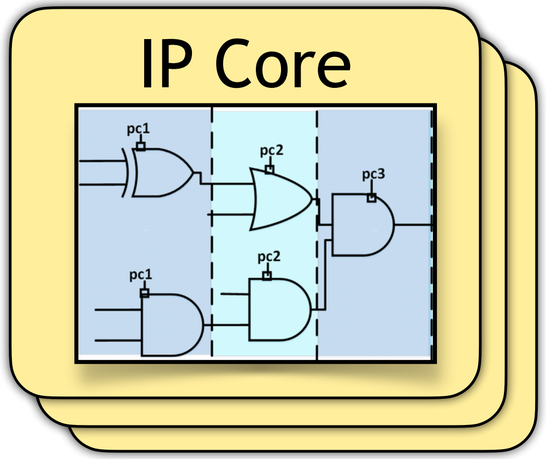


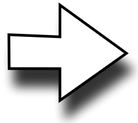
```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

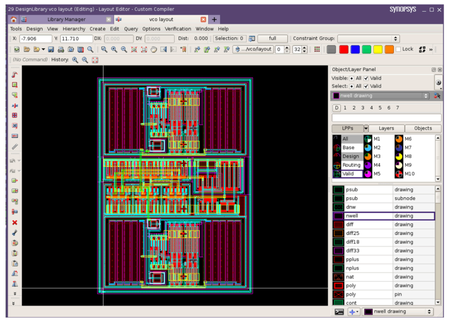
Developer

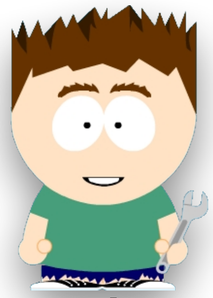
Supply Chain Stakeholders

IP Core Author









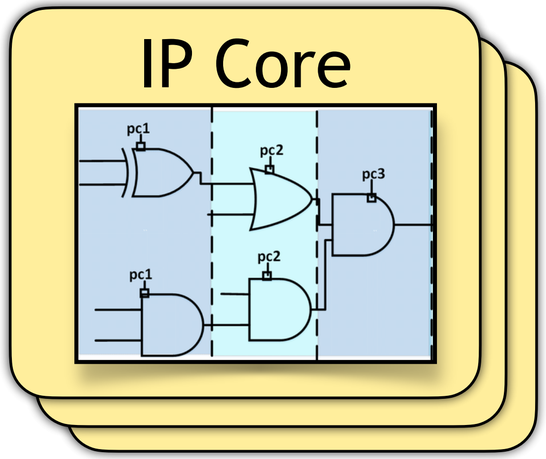
Developer

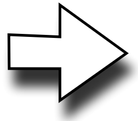
```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

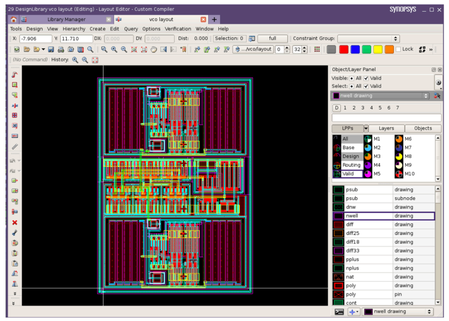
EDA Tool Vendor

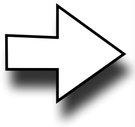
Supply Chain Stakeholders

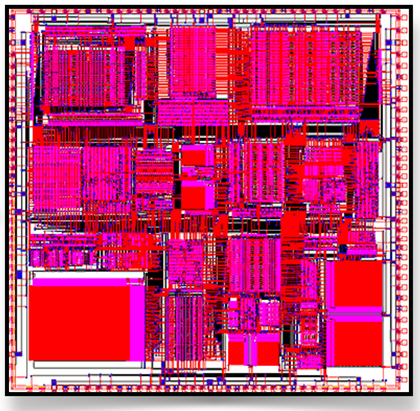
IP Core Author











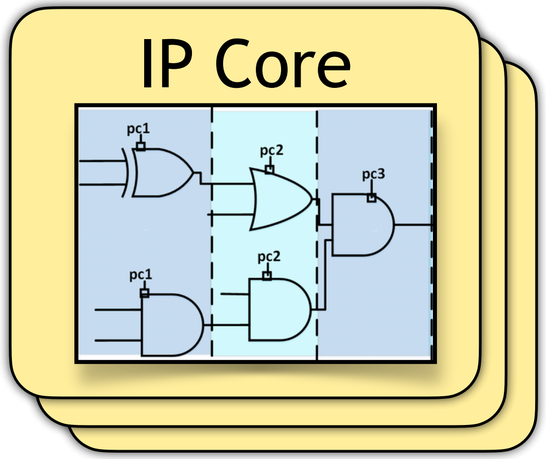
Developer

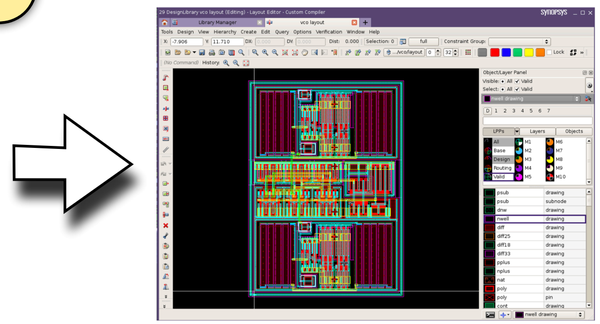
```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

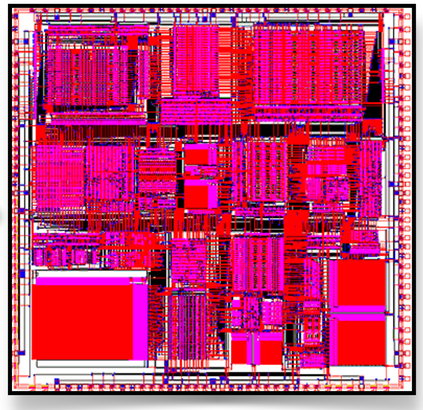
EDA Tool Vendor

Supply Chain Stakeholders

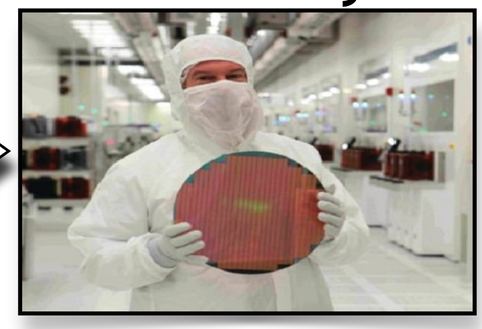
IP Core Author







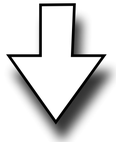
Foundry



```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

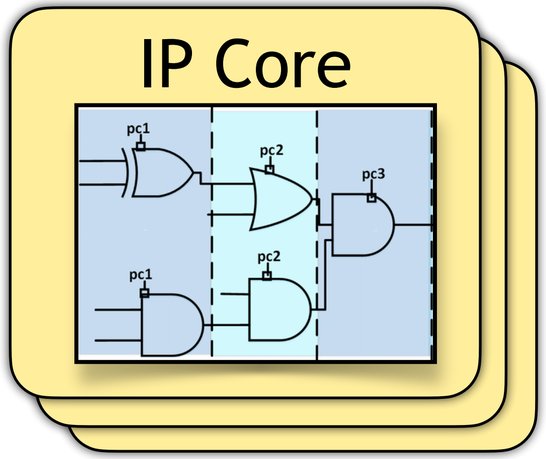
Developer

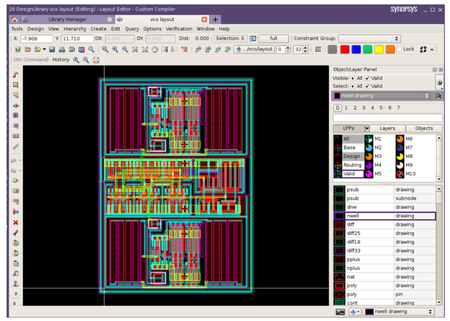
EDA Tool Vendor

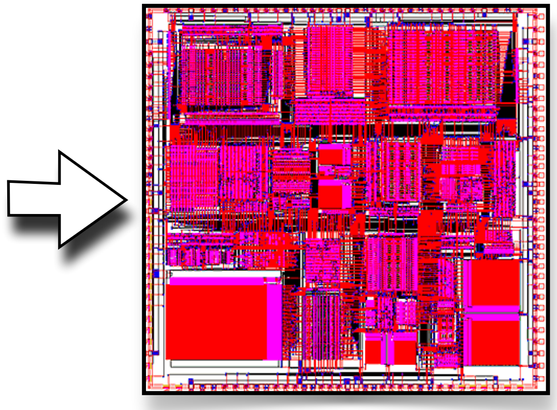


Supply Chain Attacks

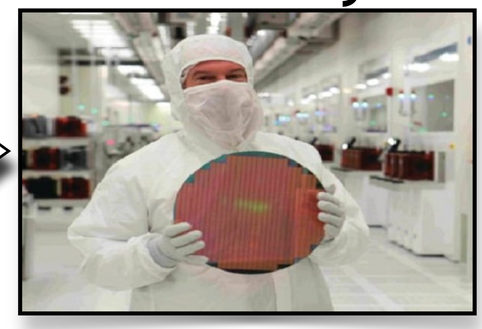
IP Core Author

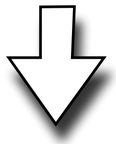






Foundry



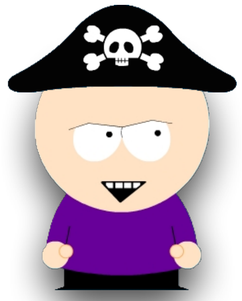


Developer

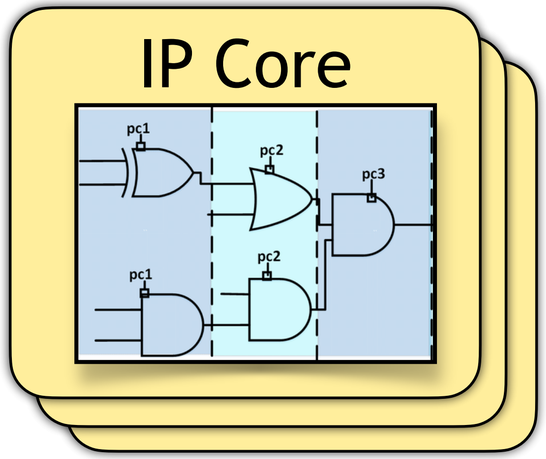
```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

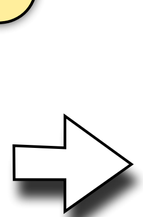
EDA Tool Vendor

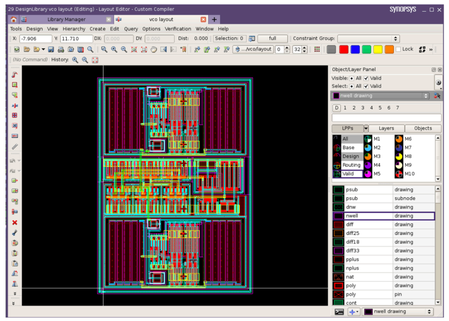
Supply Chain Attacks

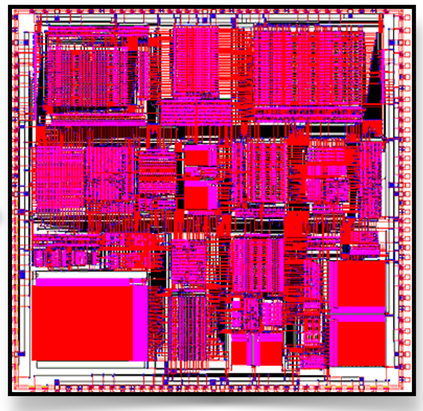


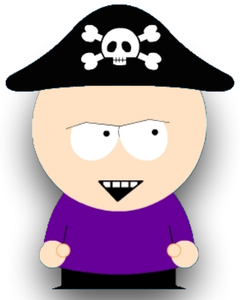
IP Core Author



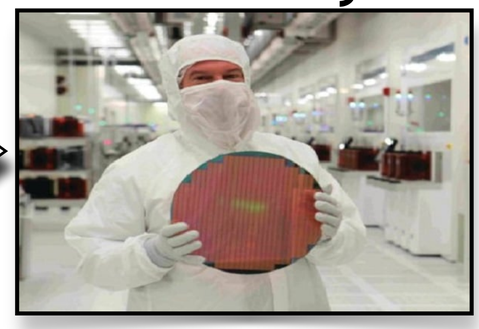


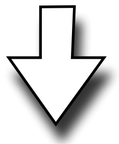


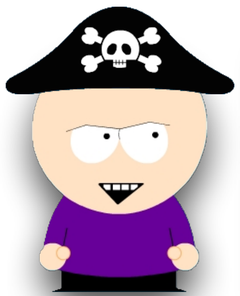




Foundry

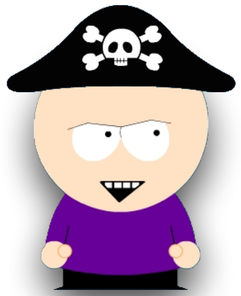






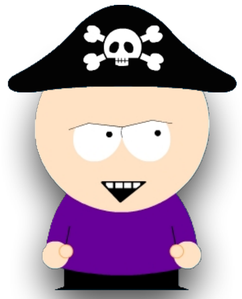
Developer

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

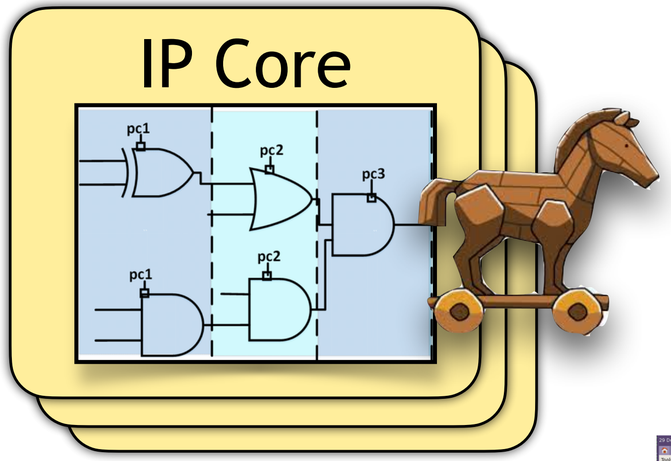


EDA Tool Vendor

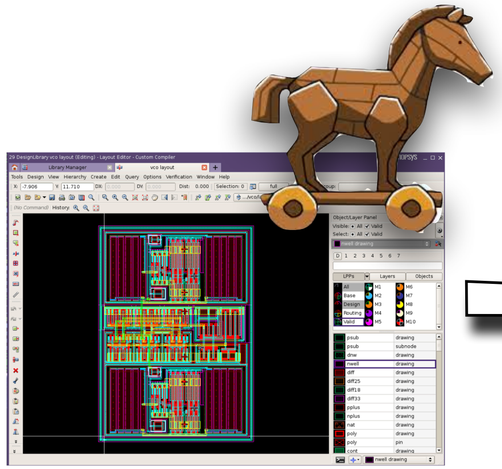
Supply Chain Attacks

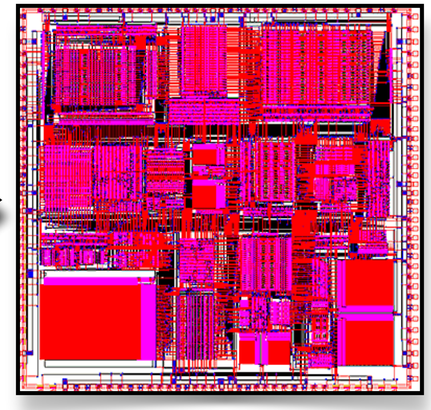


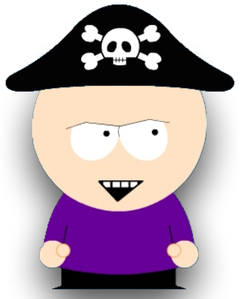
IP Core Author



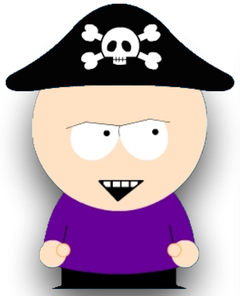
License Agreement





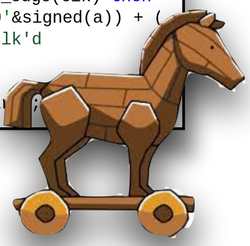


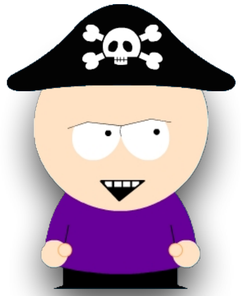
Foundry



Developer

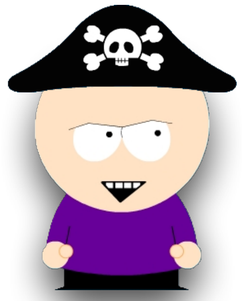
```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_a
```



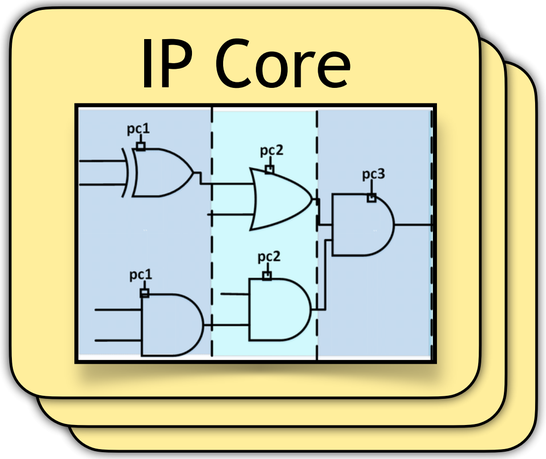


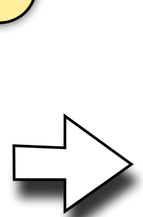
EDA Tool Vendor

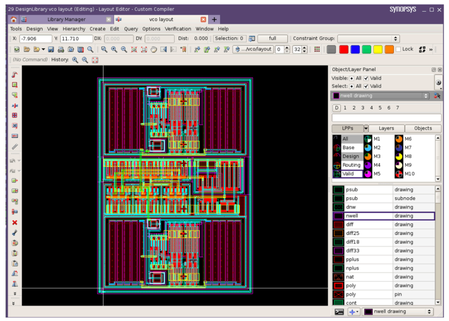
Supply Chain Attacks

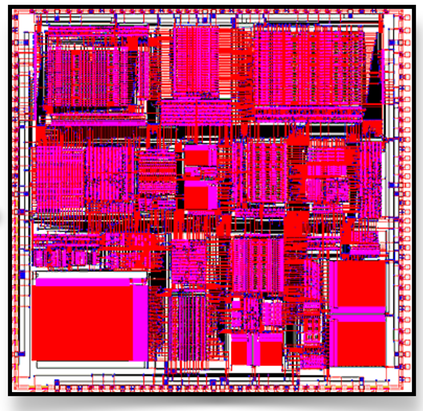


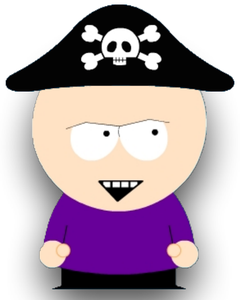
IP Core Author



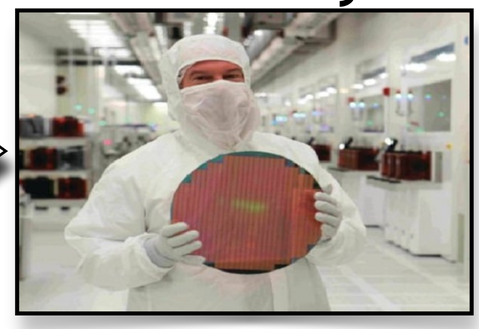


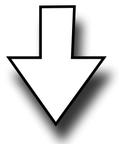


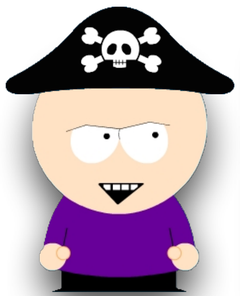




Foundry

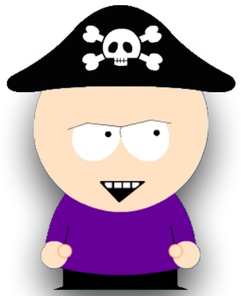






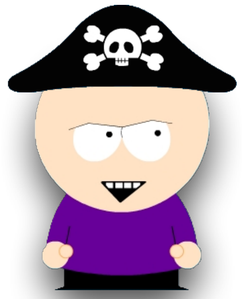
Developer

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

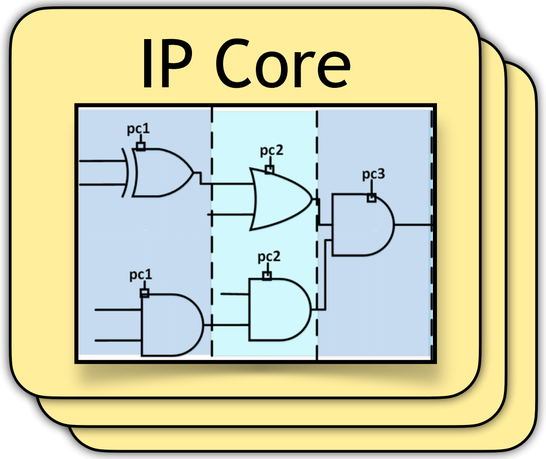


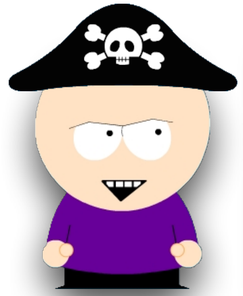
EDA Tool Vendor

Supply Chain Attacks

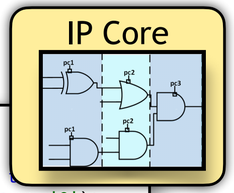


IP Core Author

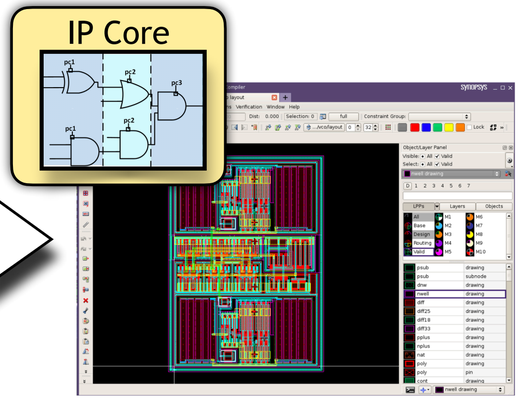


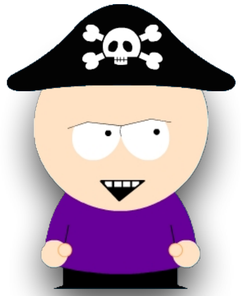


Developer

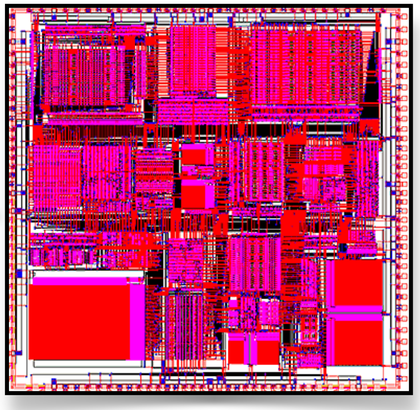


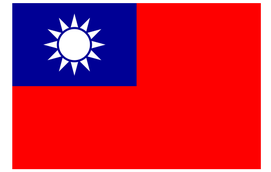

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1')
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

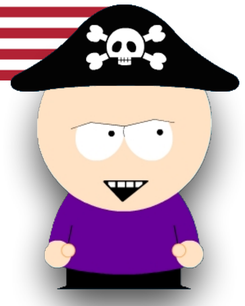




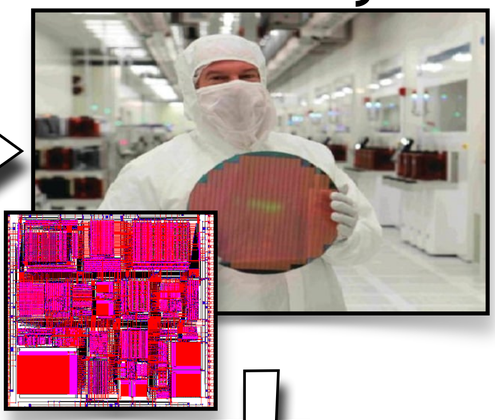
EDA Tool Vendor



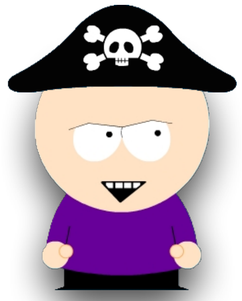




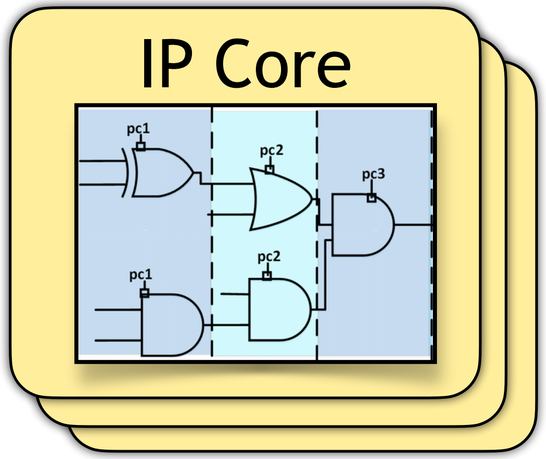
Foundry

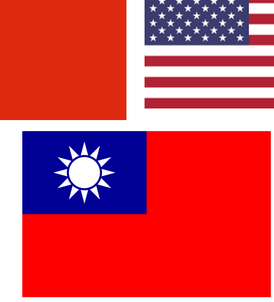


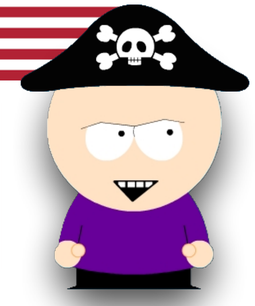
Supply Chain Attacks

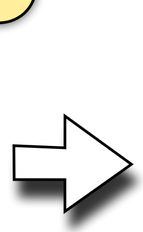


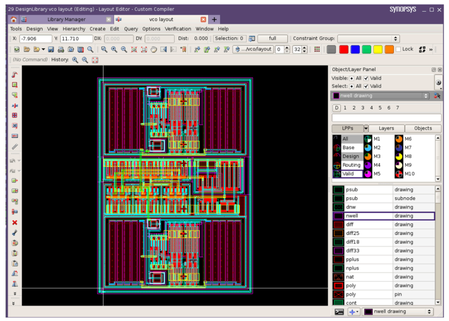
IP Core Author

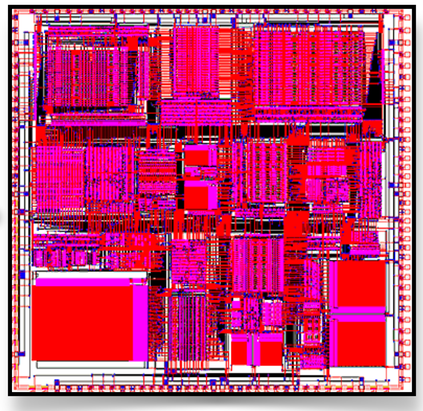


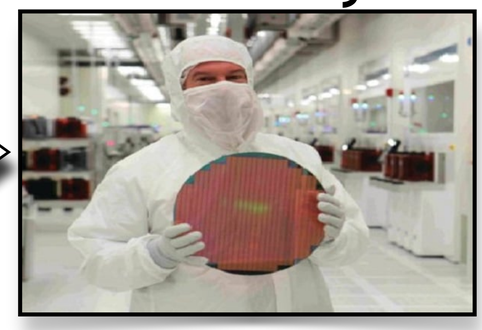




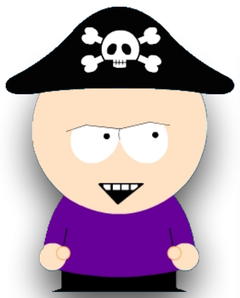






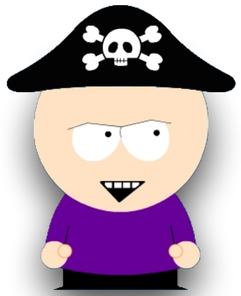


Foundry



Developer

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```



EDA Tool Vendor

IEEE Standard 1735

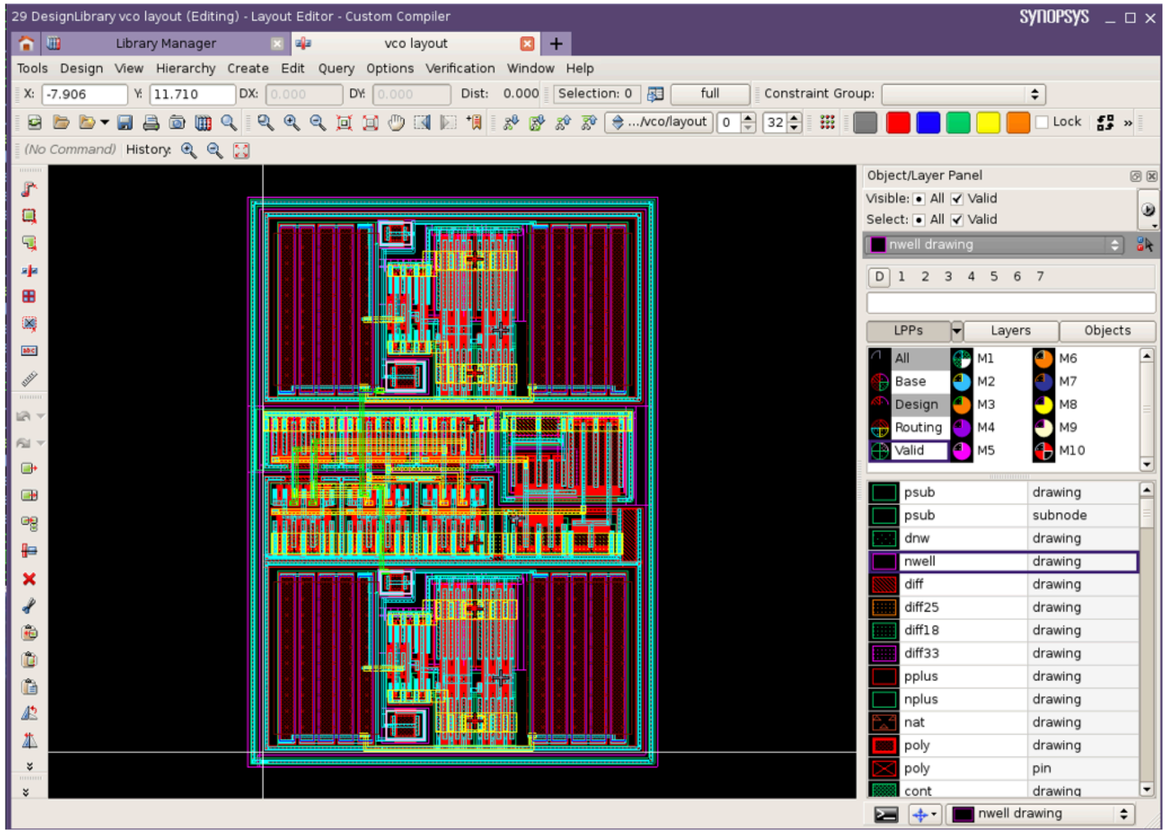
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

Cryptolope

Developer



IEEE Standard 1735

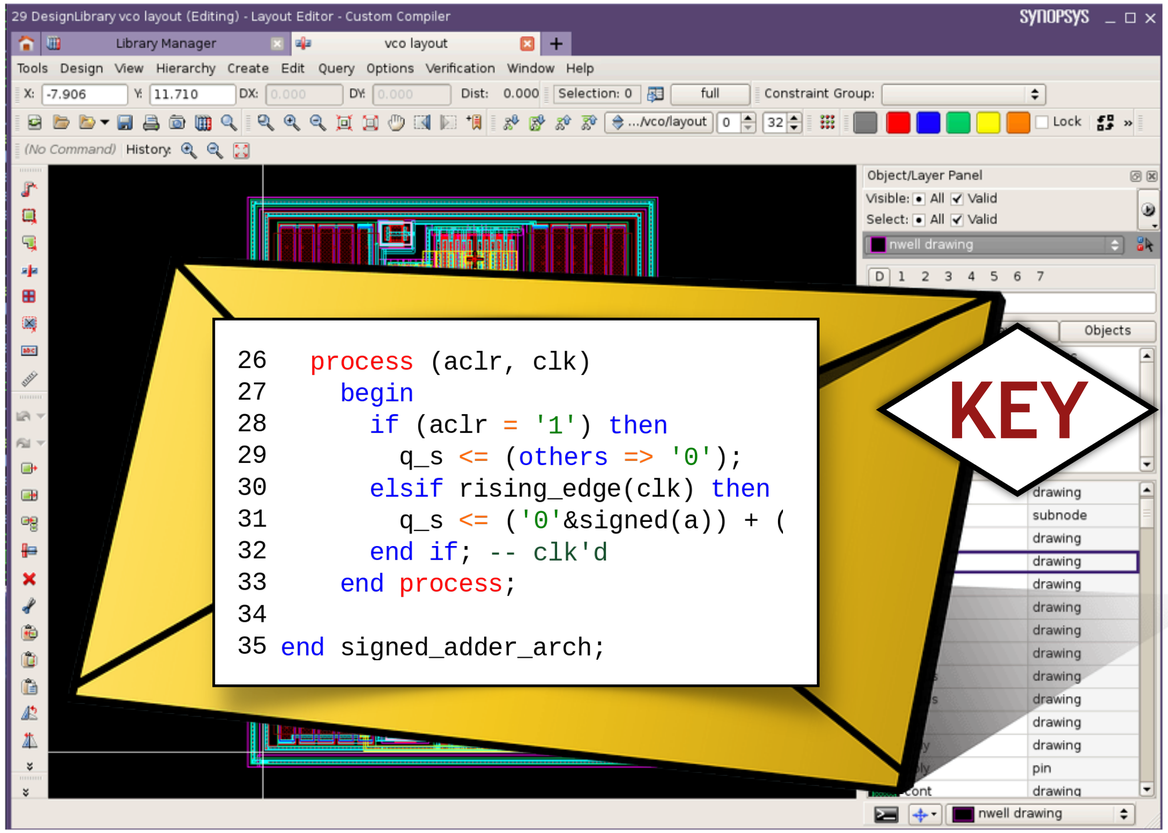
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

Cryptolope

Developer




```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

IEEE Standard 1735

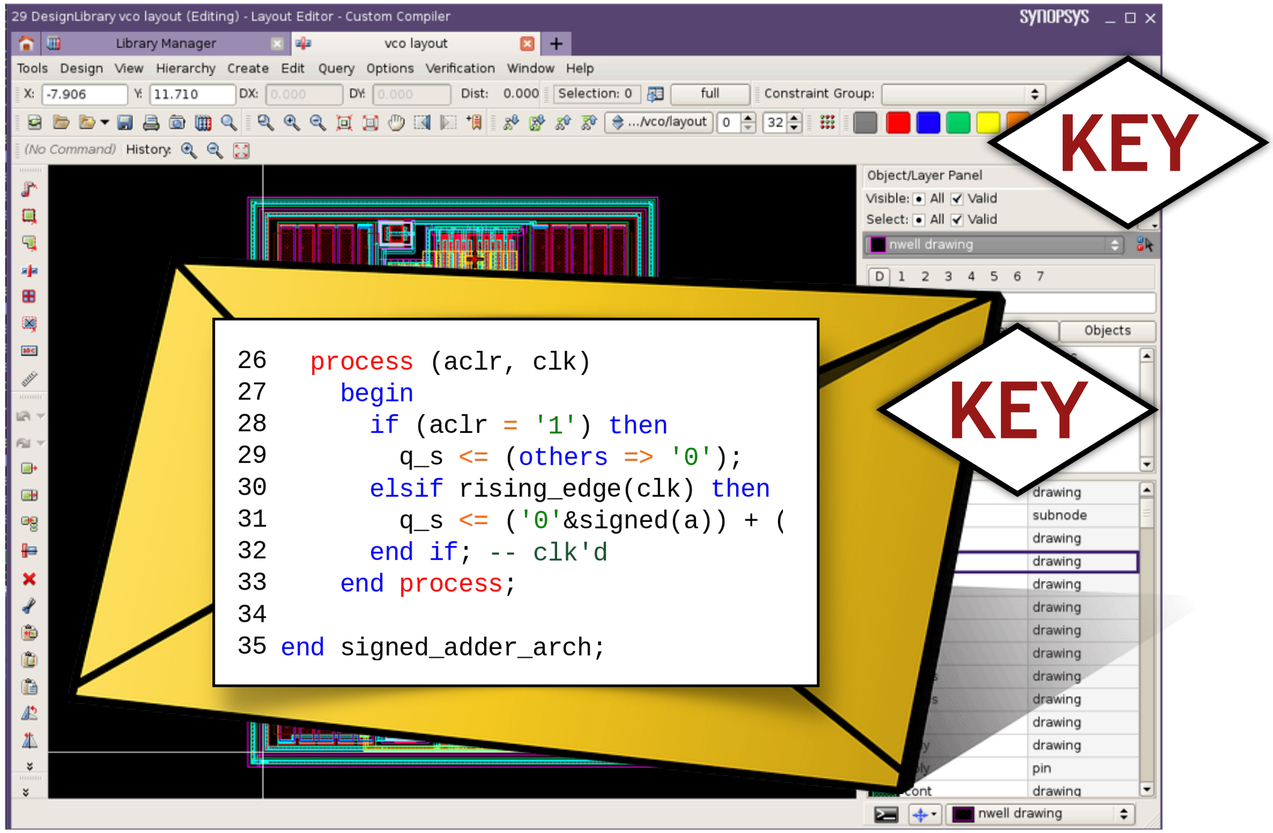
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

Cryptolope

Developer



IEEE Standard 1735

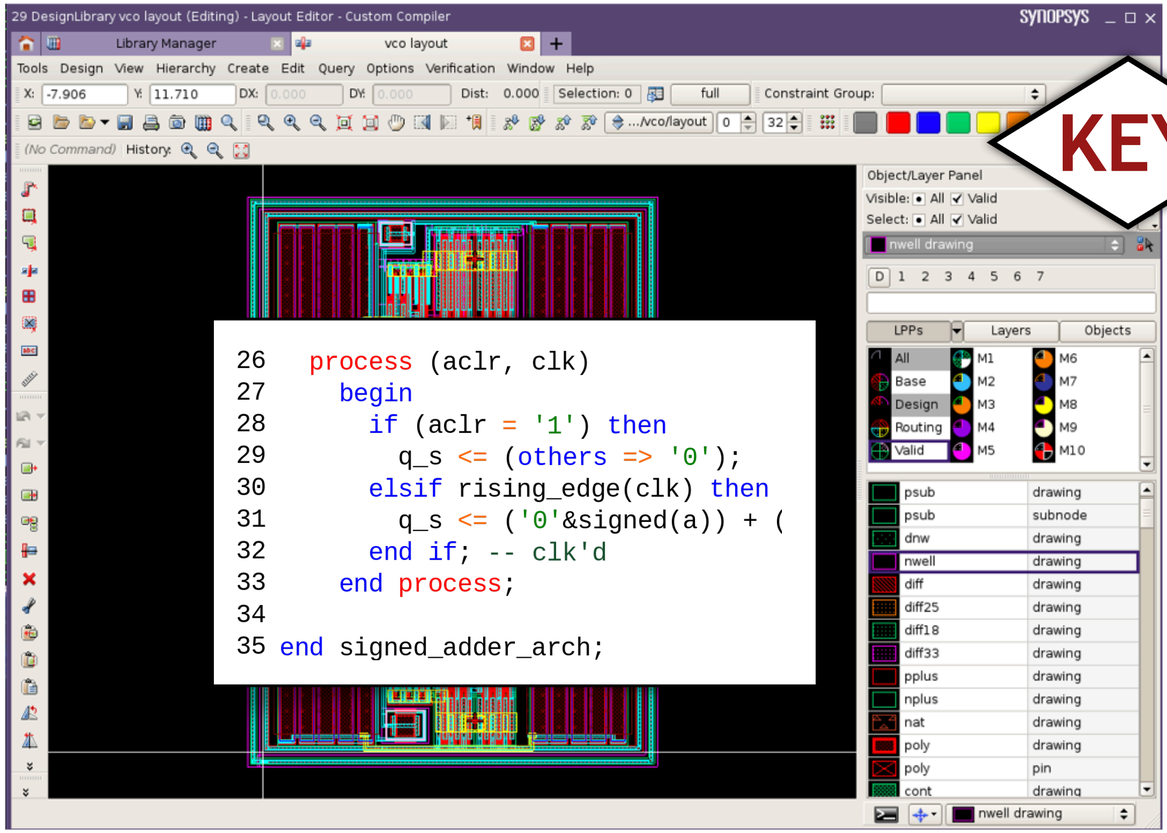
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

Cryptolope

Developer



IEEE Standard 1735

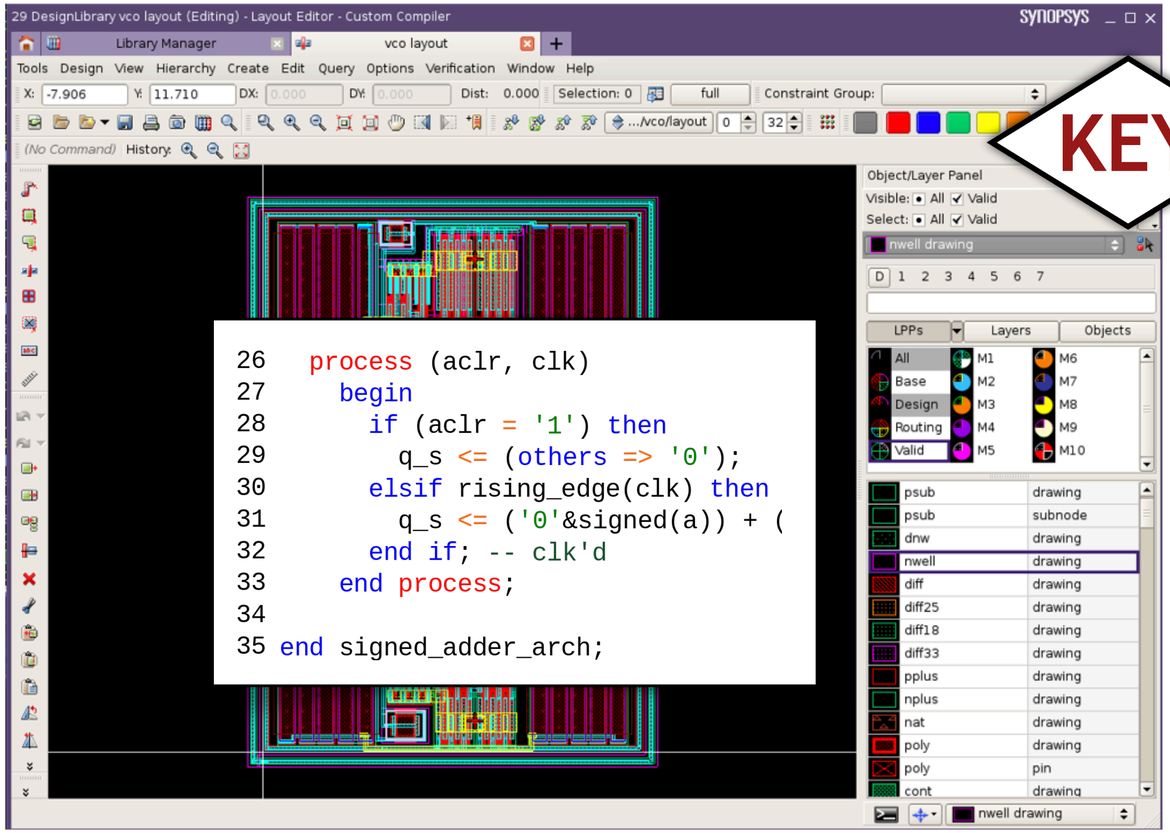
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

Cryptolope

Developer



KEY

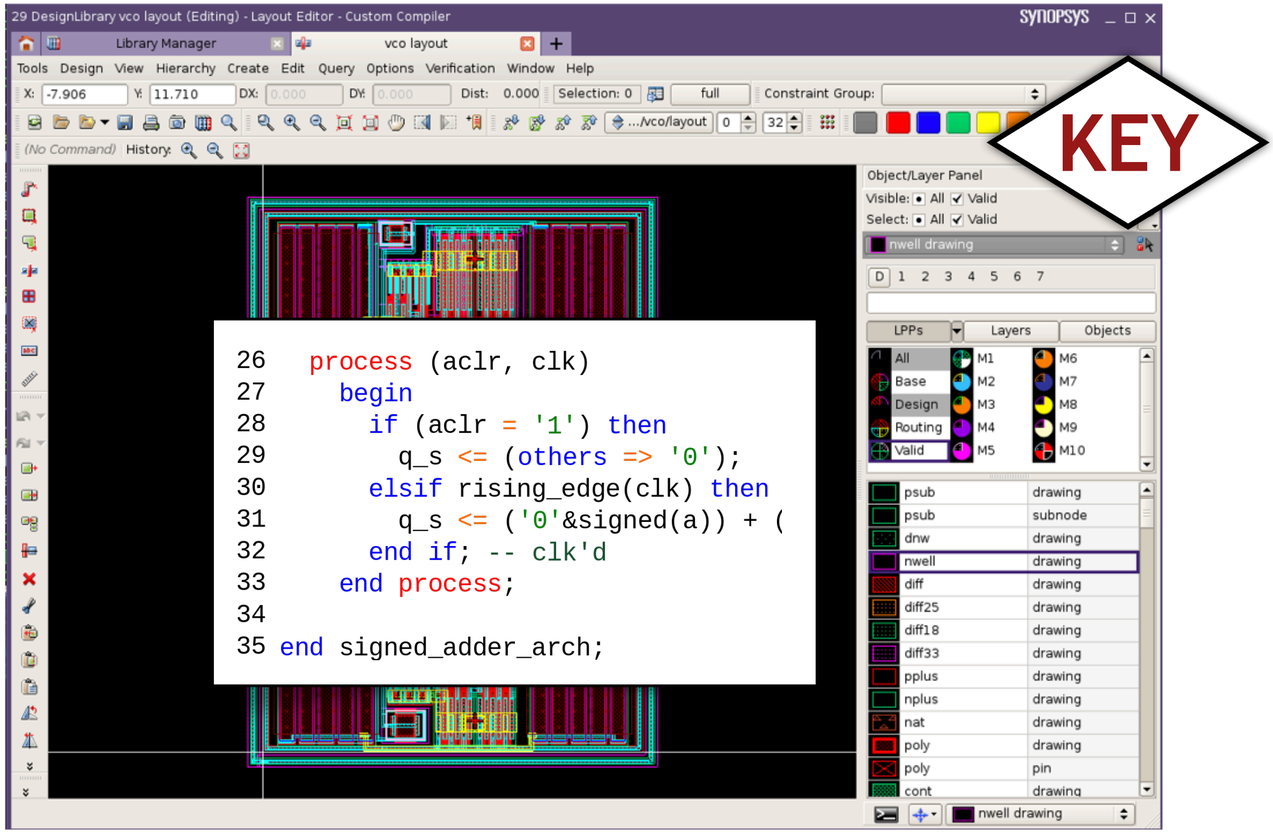
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

How not to Protect Your IP - An Industry-Wide Break of IEEE 1735 Implementations, IEEE S&P'22

Developer



KEY

IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

SIEMENS

Microsemi

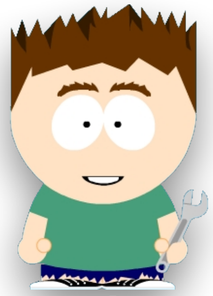
LATTICE SEMICONDUCTOR

XILINX

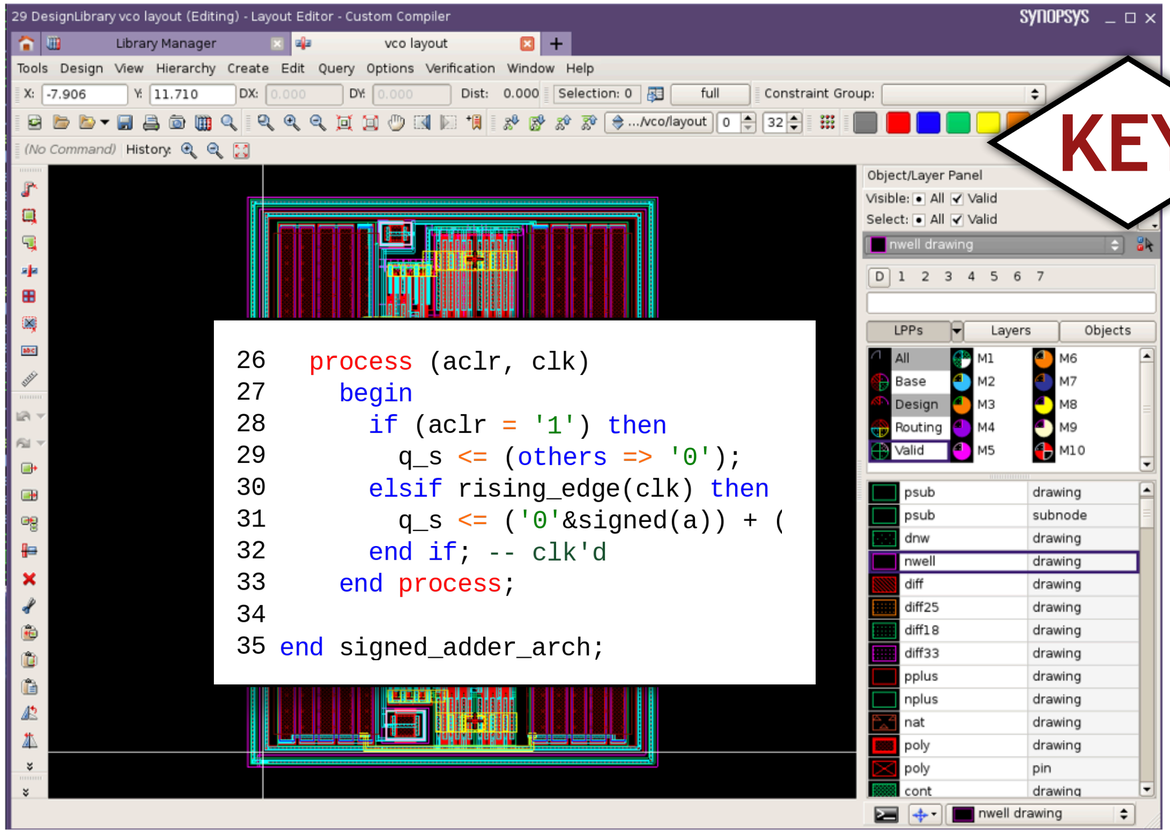
cadence®

QUARTUS PRIME DESIGN SOFTWARE

How not to Protect Your IP - An Industry-Wide Break of IEEE 1735 Implementations, IEEE S&P'22



Developer



KEY

IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

SIEMENS

Microsemi

LATTICE SEMICONDUCTOR

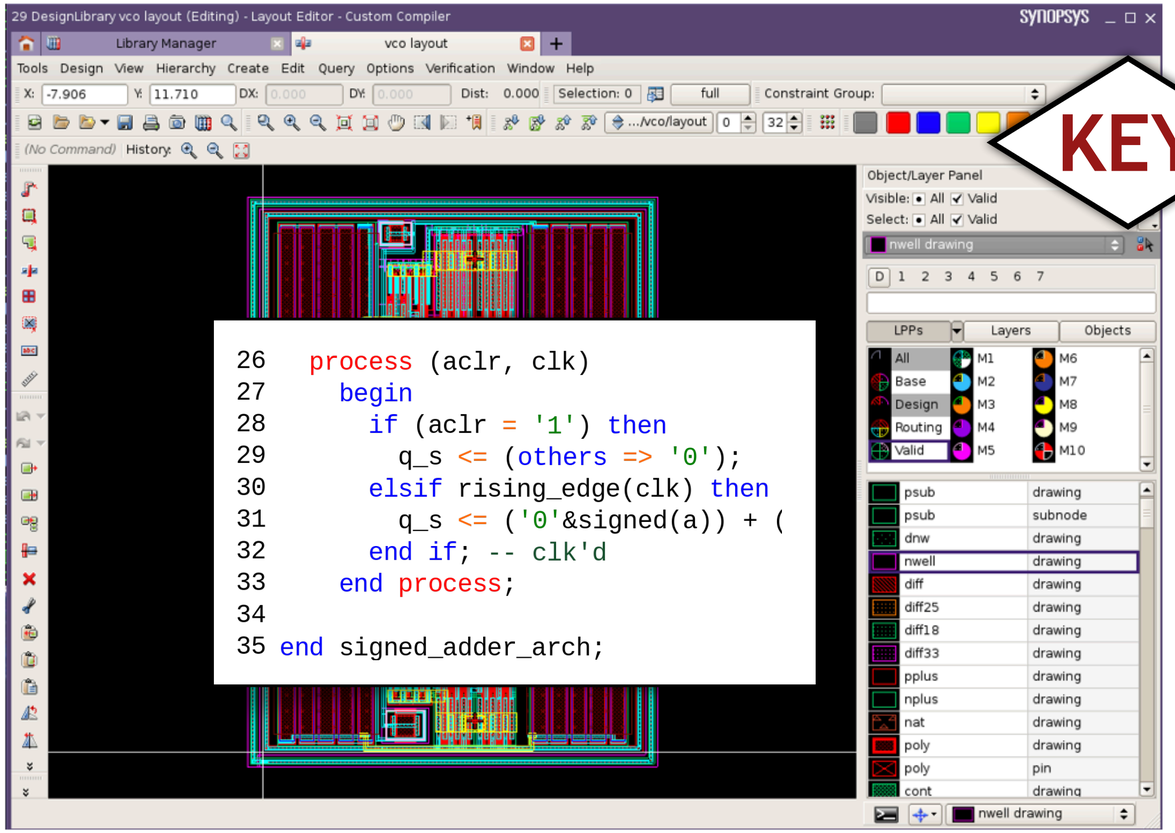
XILINX

cadence®

QUARTUS PRIME DESIGN SOFTWARE

Each tool vendor needs to decide what, if any, anti-debug, anti-tamper, and anti-key-discovery technology they will use.
— IEEE 1735

Developer



KEY

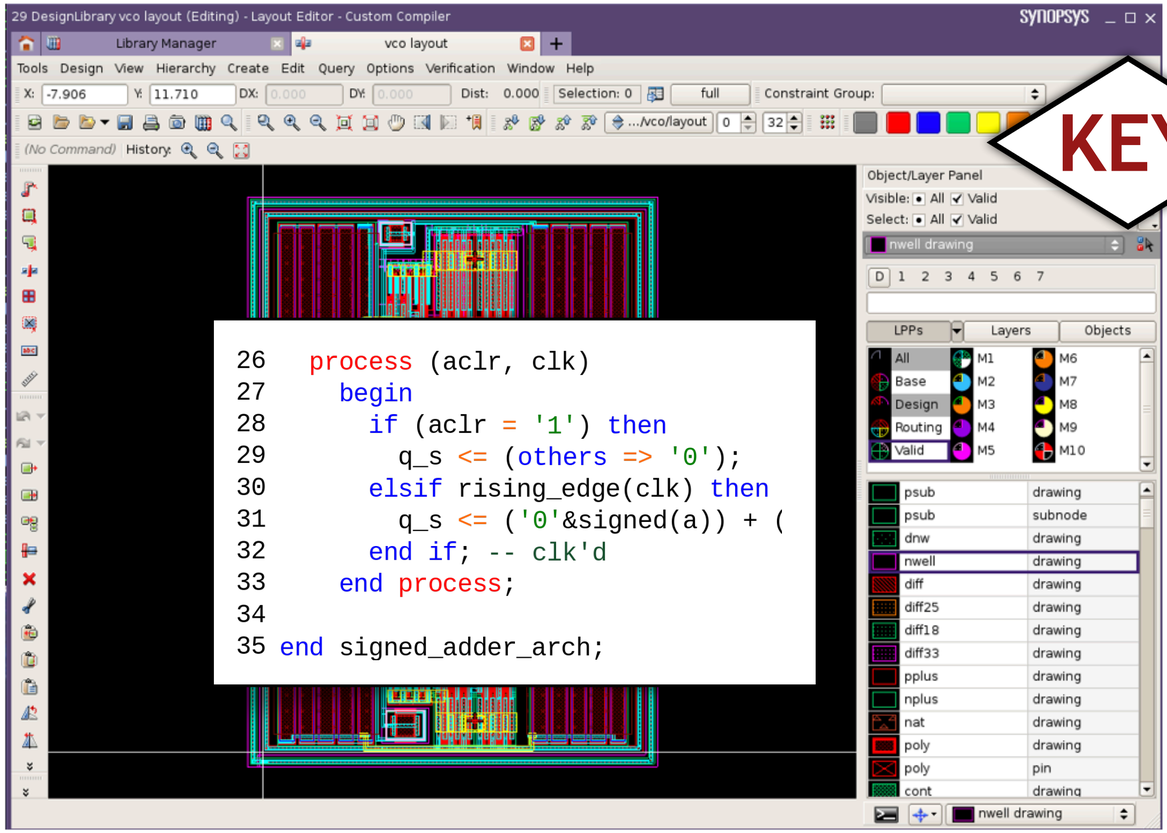
IP Core Author

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

KEY

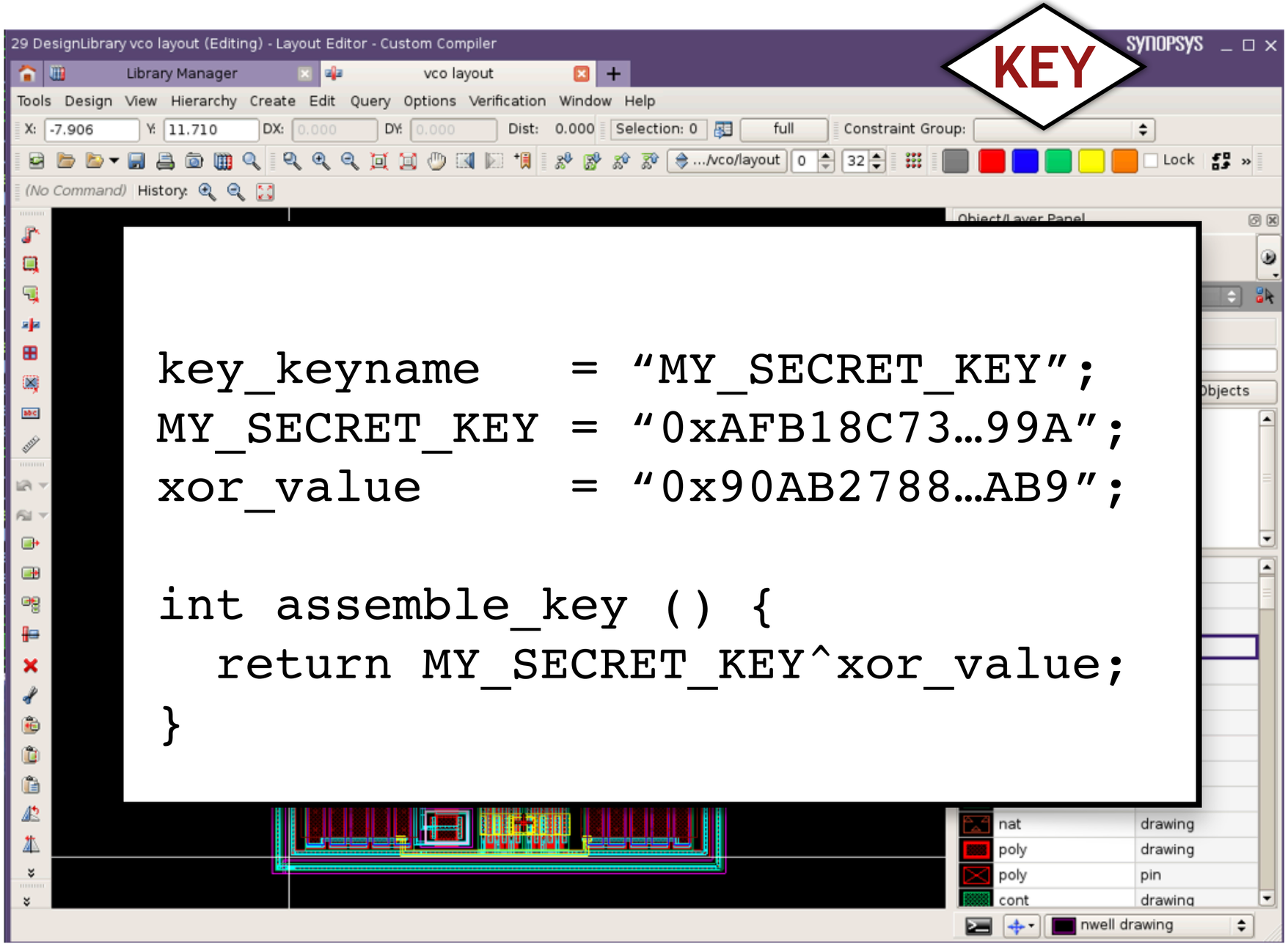
Each tool vendor needs to decide what, if any, anti-debug, anti-tamper, and anti-key-discovery technology they will use.
— IEEE 1735

Developer



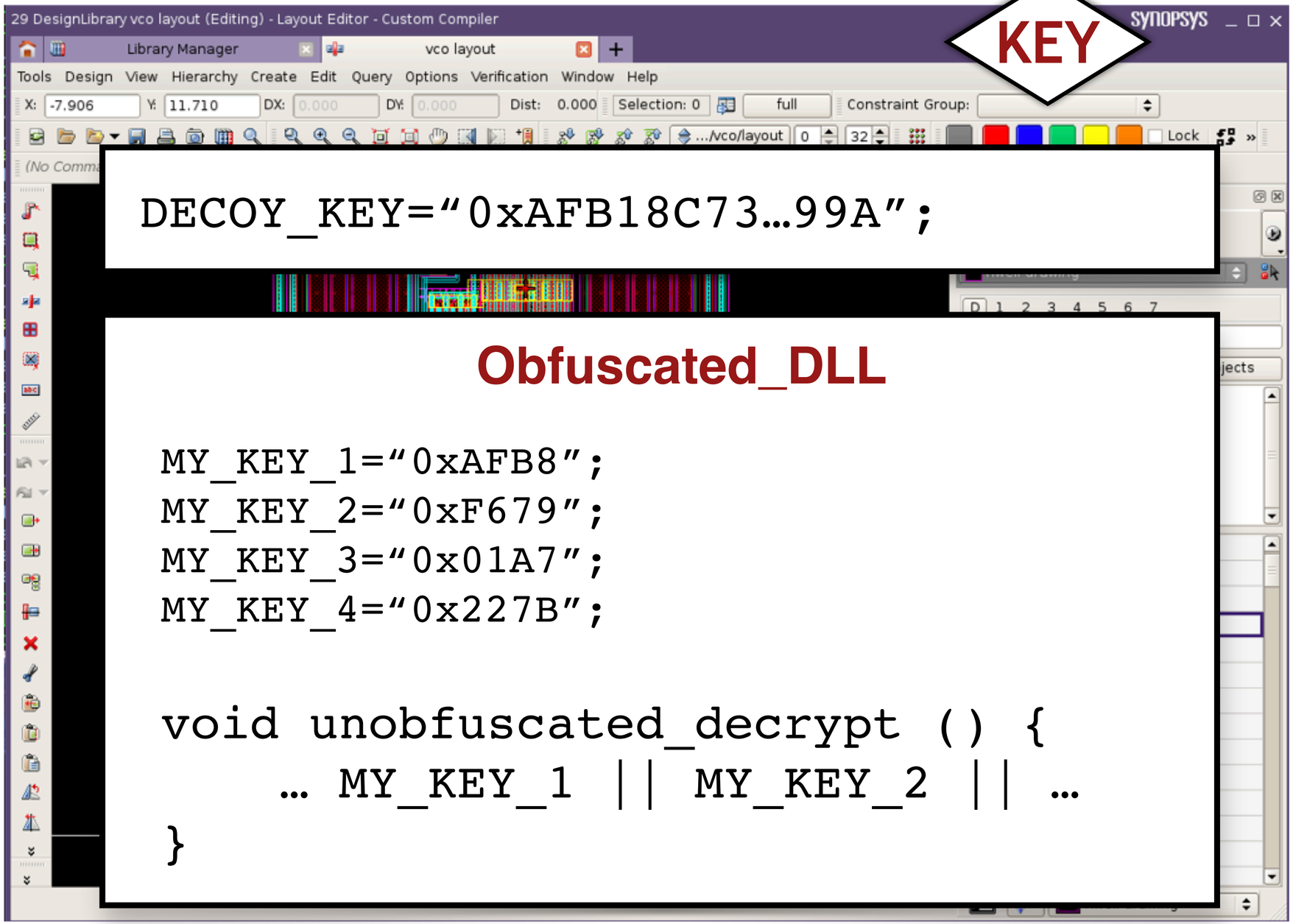
KEY

- Broken by searching for the “keyname” in the binary.
- Key trivially hidden by xor:ing with a constant.
- **Time to break: 1h**

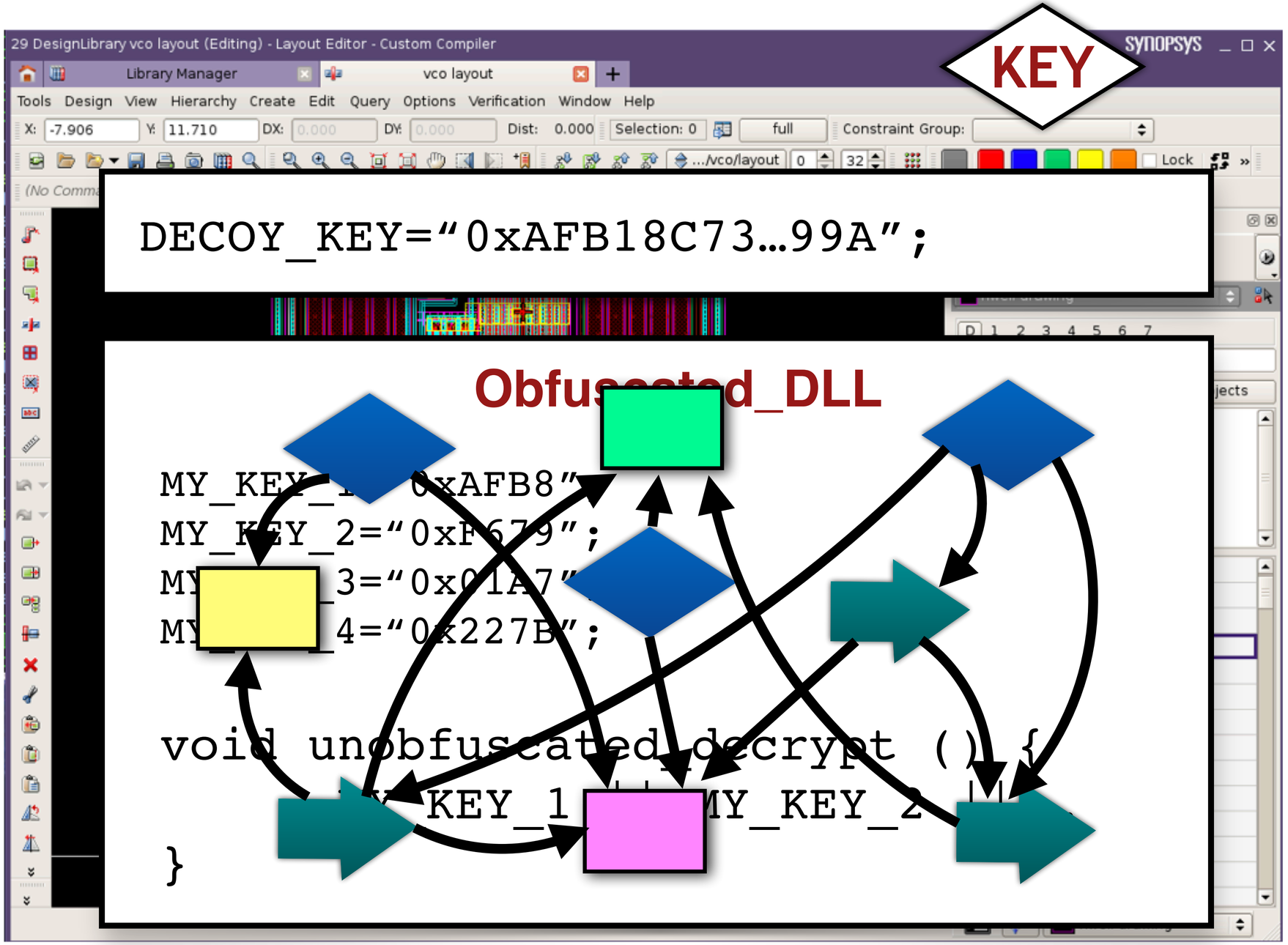


Microsemi Libero System-on-Chip

- DLL is obfuscated - suspicious!
- DLL has anti-debugging!
- DLL has integrity checks!
- Key is split in pieces.
- Break by debugging.
- **Time to break: 1 week**



- DLL is obfuscated - suspicious!
- DLL has anti-debugging!
- DLL has integrity checks!
- Key is split in pieces.
- Break by debugging.
- **Time to break: 1 week**



Conclusion

- In our case studies, potent **software obfuscation proves to be the one countermeasure that significantly increases attack time**.
- **Cryptography […] must always be combined with strong software obfuscation** techniques to suppress any attempt on static or dynamic analysis.

How not to Protect Your IP - And Industry-Wide Break of IEEE 1735 Implementations, IEEE S&P'22

Case Study 2:

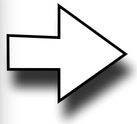
Decimorpmoc

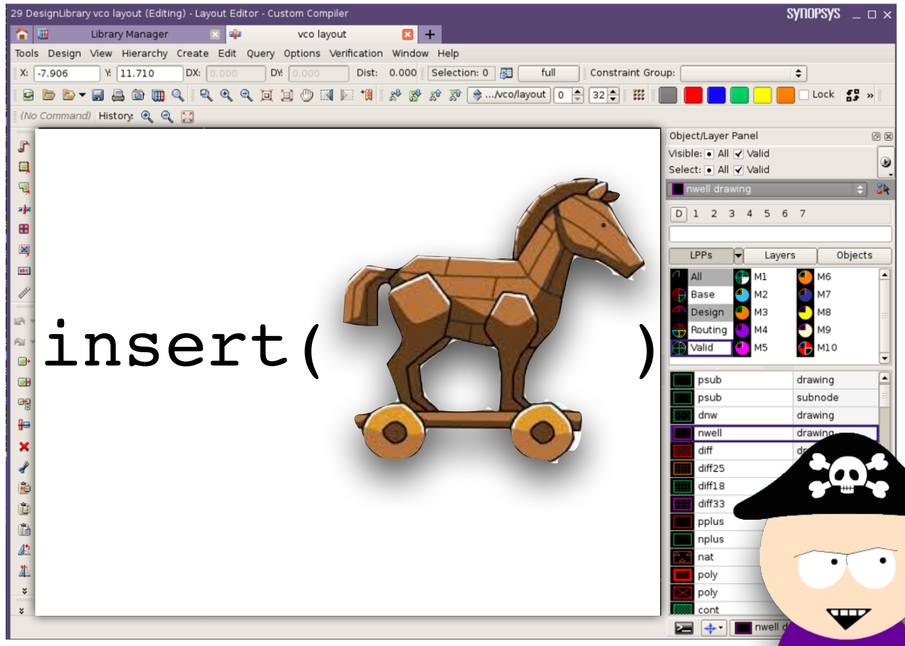
slooT

Case Study 2: Compromised Tools

EDA Supply Chain Attacks

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

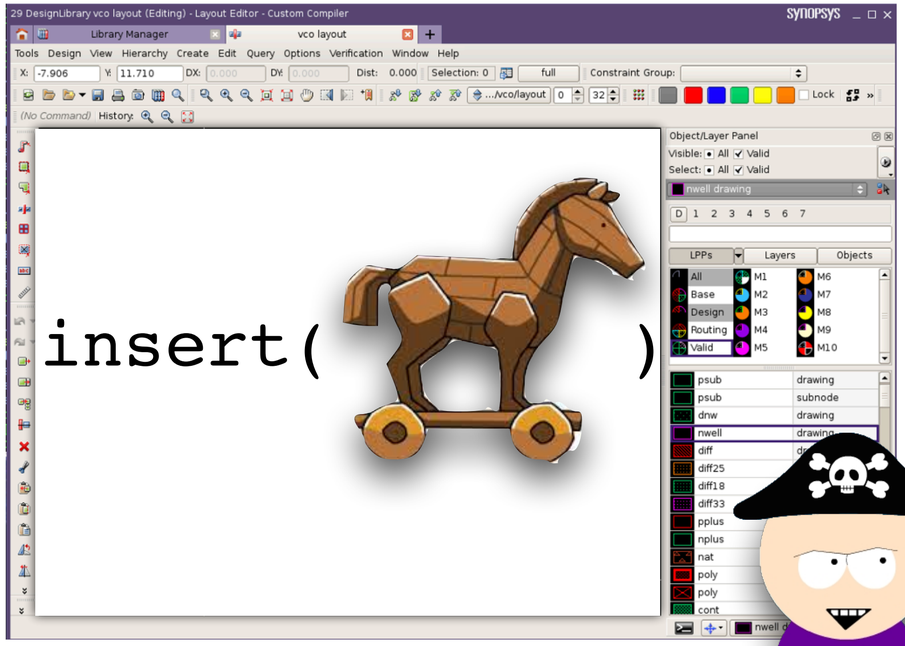




Evil EDA
Tool Vendor

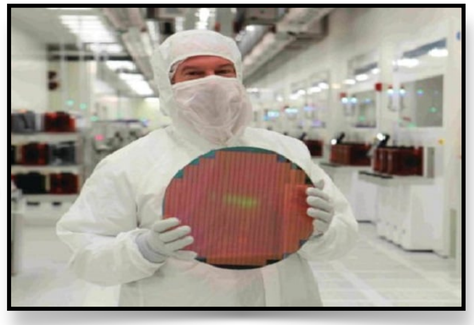
EDA Supply Chain Attacks

```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```



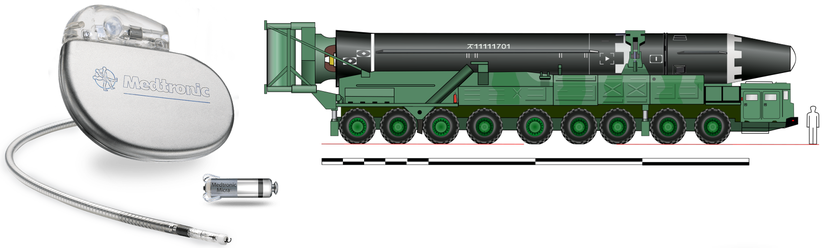
insert( )

Evil EDA
Tool Vendor

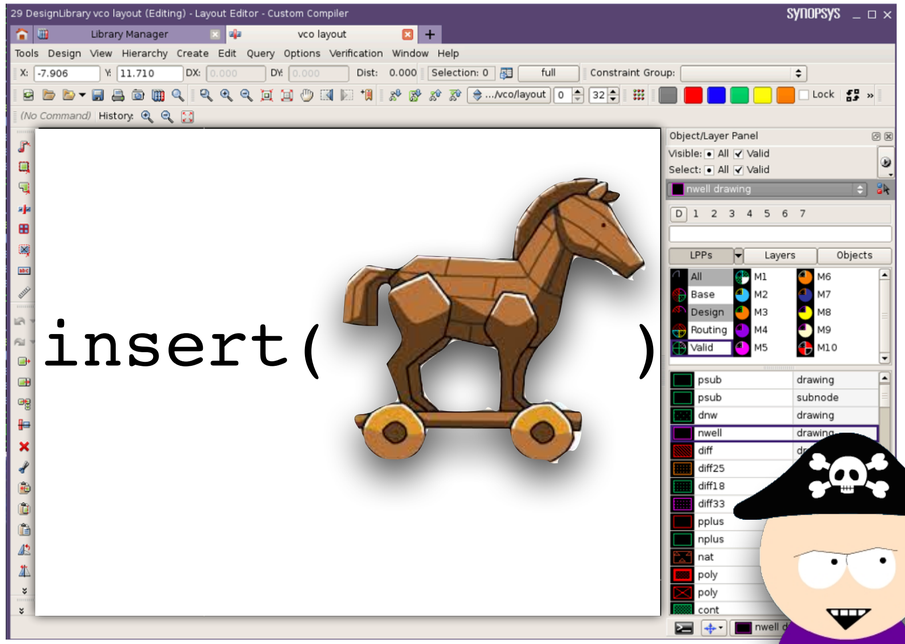


Foundry

EDA Supply Chain Attacks

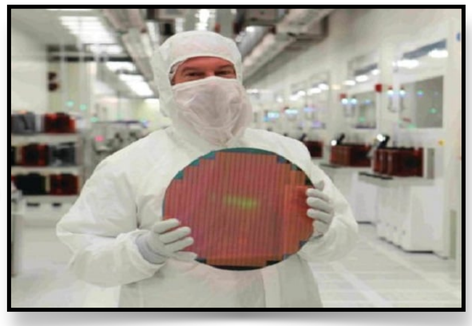


```
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + (
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

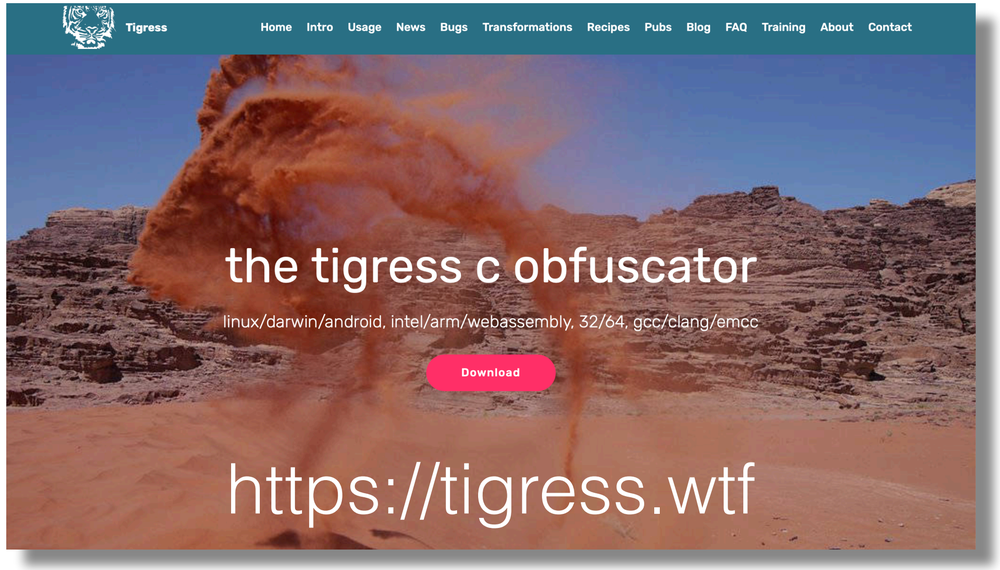


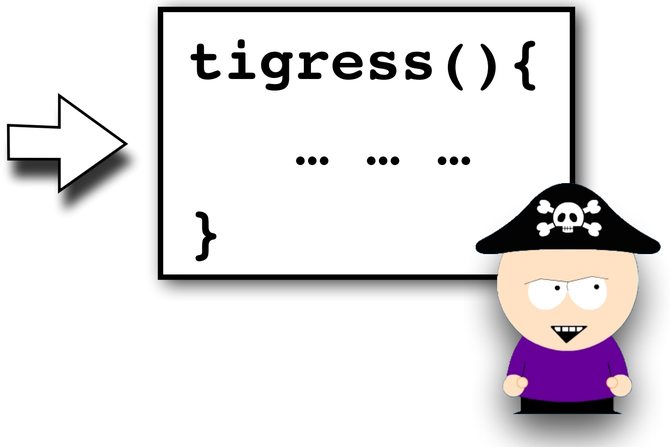
insert( )

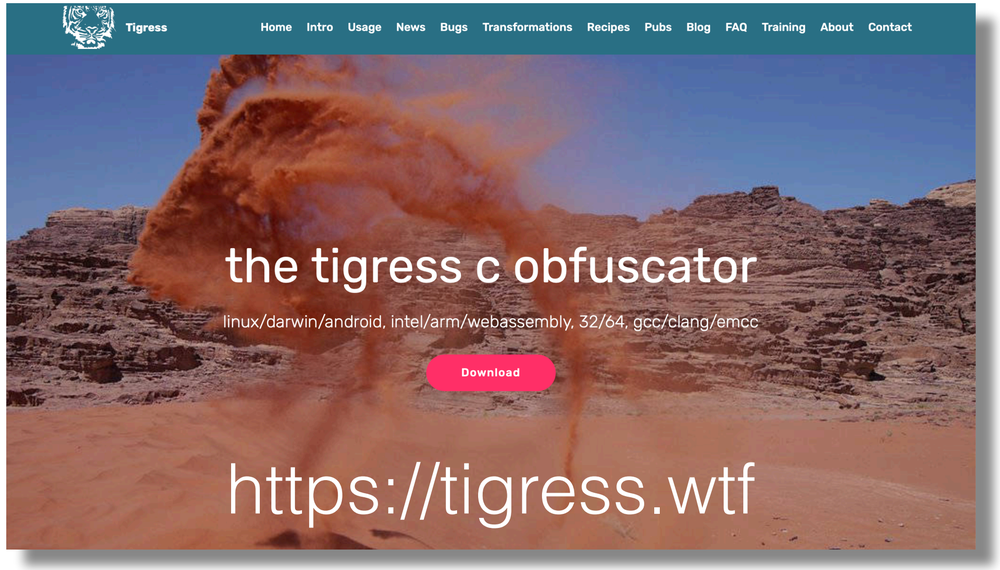
Evil EDA
Tool Vendor

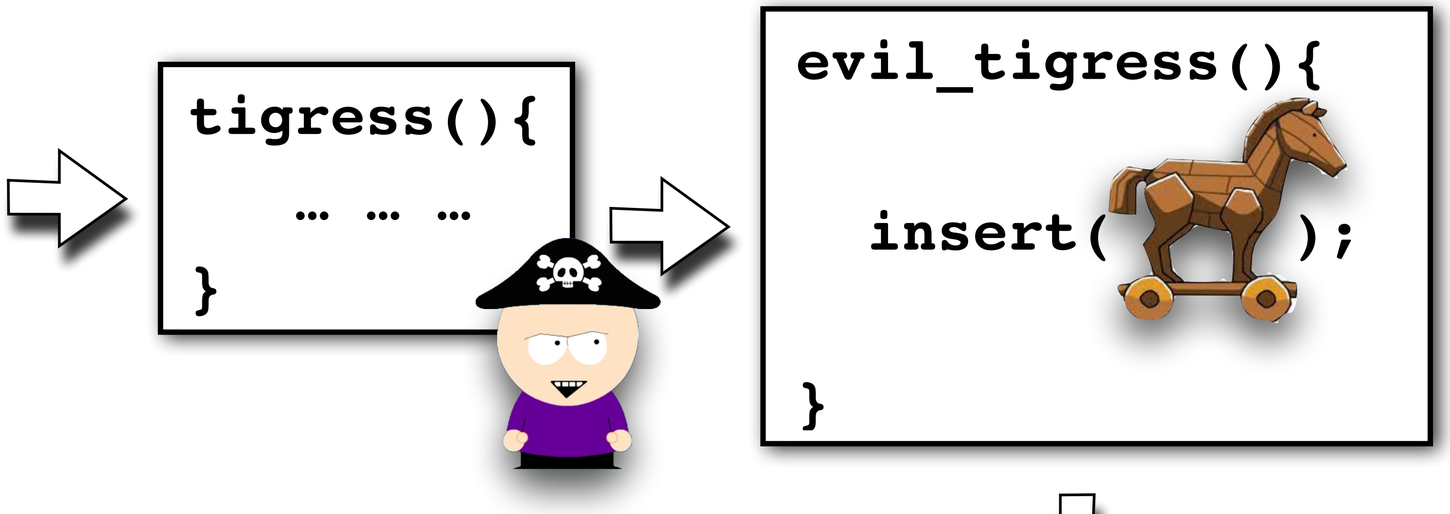


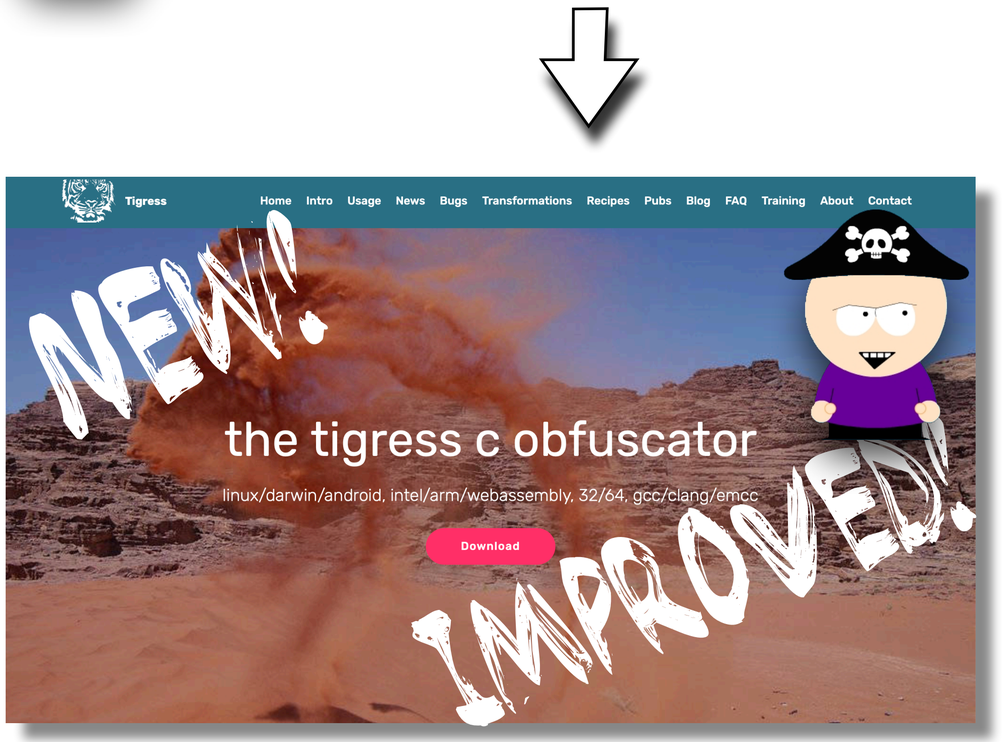
Foundry


Tigress
Home Intro Usage News Bugs Transformations Recipes Pubs Blog FAQ Training About Contact
the tigress c obfuscator
linux/darwin/android, intel/arm/webassembly, 32/64, gcc/clang/emcc
Download
https://tigress.wtf

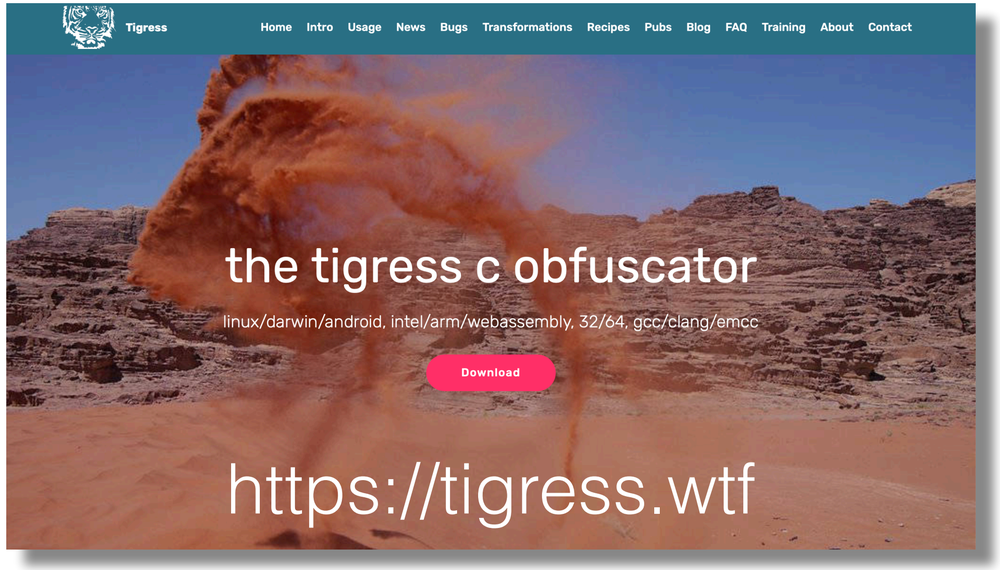

tigress(){
… … …
}

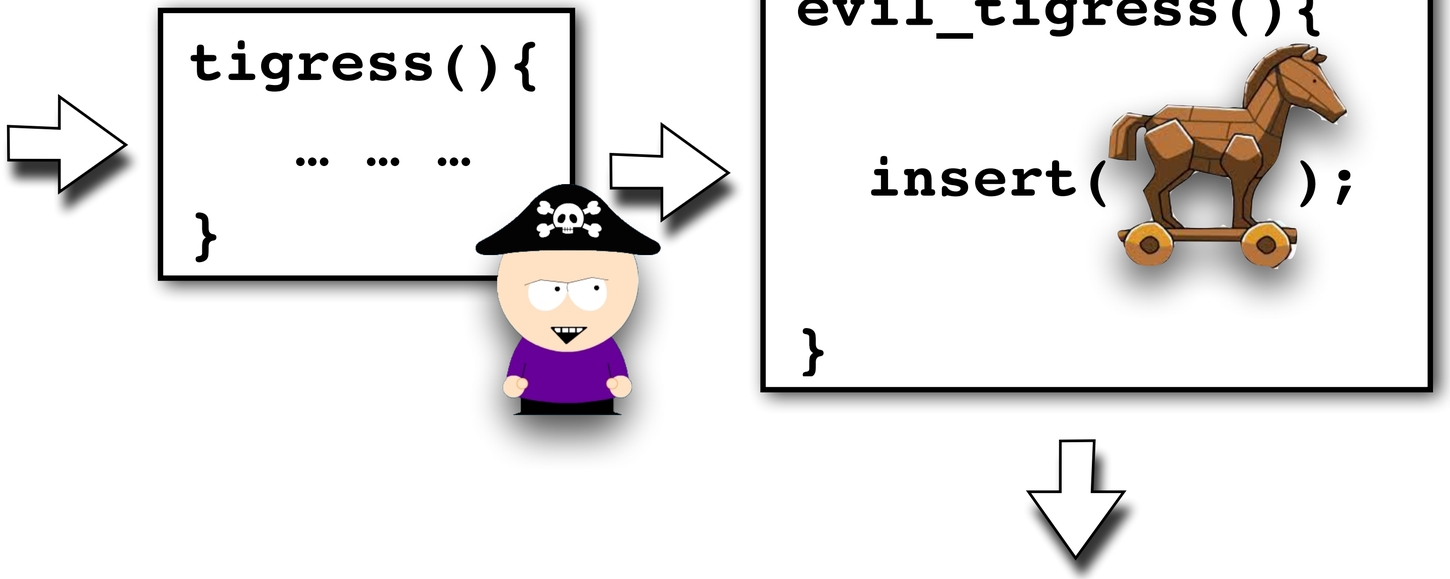


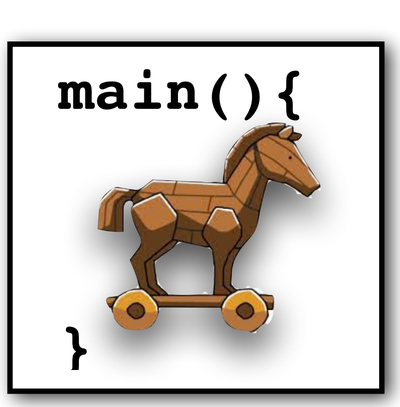




https://evil_tigress.com

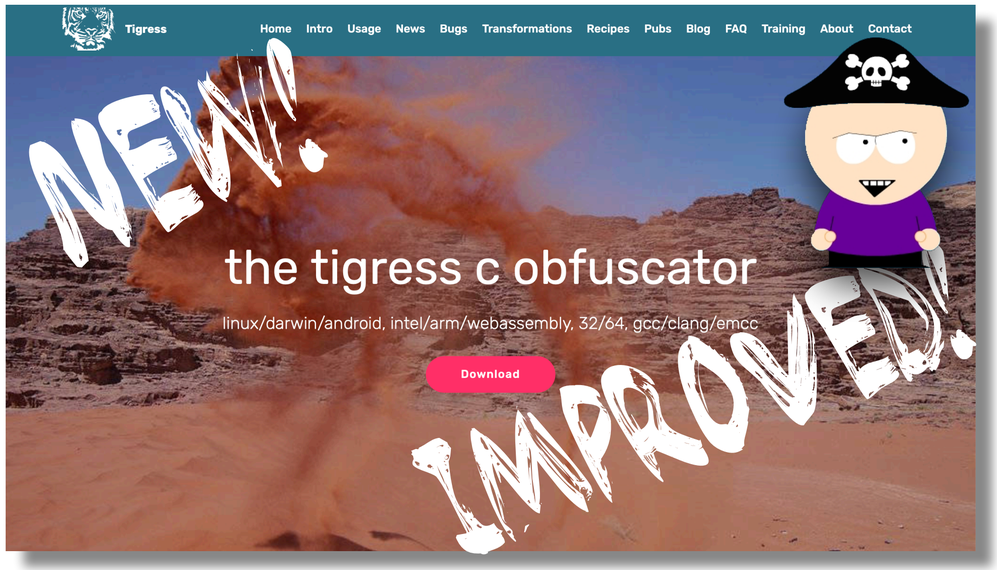






Evil-Tigress

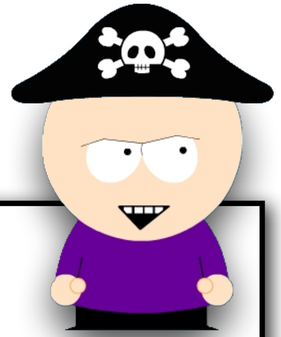
Developer



https://evil_tigress.com

```
foo(){
}
```

Developer



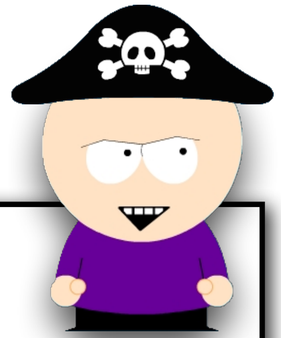
```
evil_tigress(){

}
```

Obfuscate

```
main(){

}
```

```
foo(){
}
```

Developer



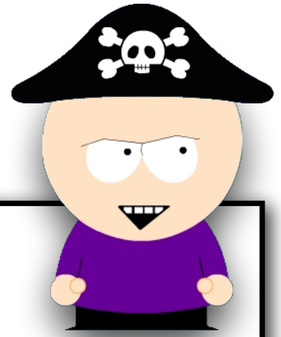
```
evil_tigress(){

}
```

Obfuscate

```
main(){

    foo(){
    }

}
```

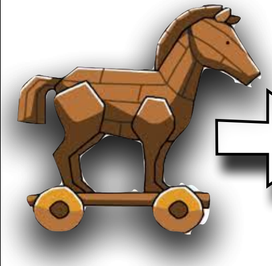
```
foo(){
}
```

Developer



```
evil_tigress(){
```

Obfuscate



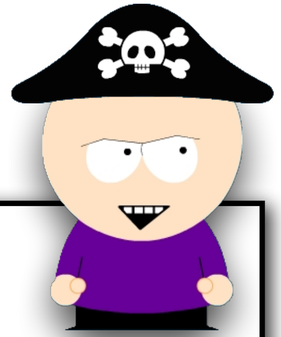
Watermark → Obfuscate → Tamper proof

```
}
```

```
main(){
    foo(){
    }
}
```

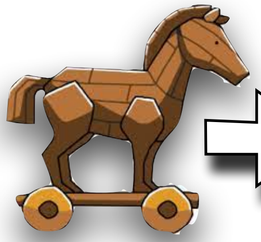
```
foo(){
}
```

Developer



```
evil_tigress(){
```

Obfuscate



Watermark

Obfuscate

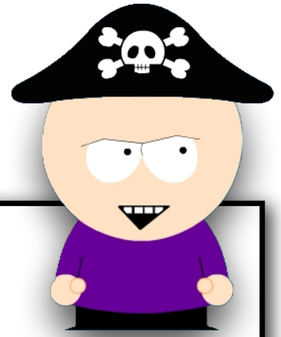
Tamper proof

```
}
```

```
main(){
    foo(){
    }
}
```

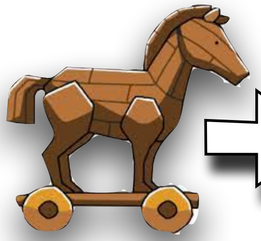
```
foo(){
}
```

Developer



```
evil_tigress(){
```

Obfuscate



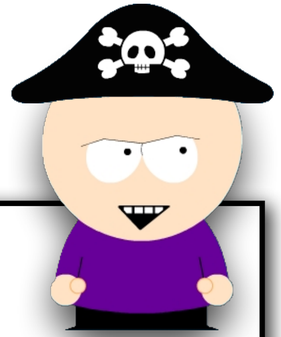
Watermark → Obfuscate → Tamper proof

```
}
```

```
main(){
    foo(){
    }
}
```

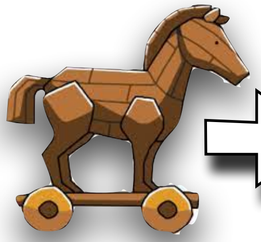
```
foo(){
}
```

Developer



```
evil_tigress(){
```

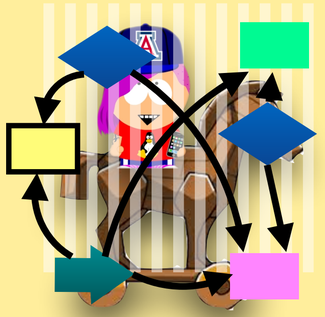
Obfuscate



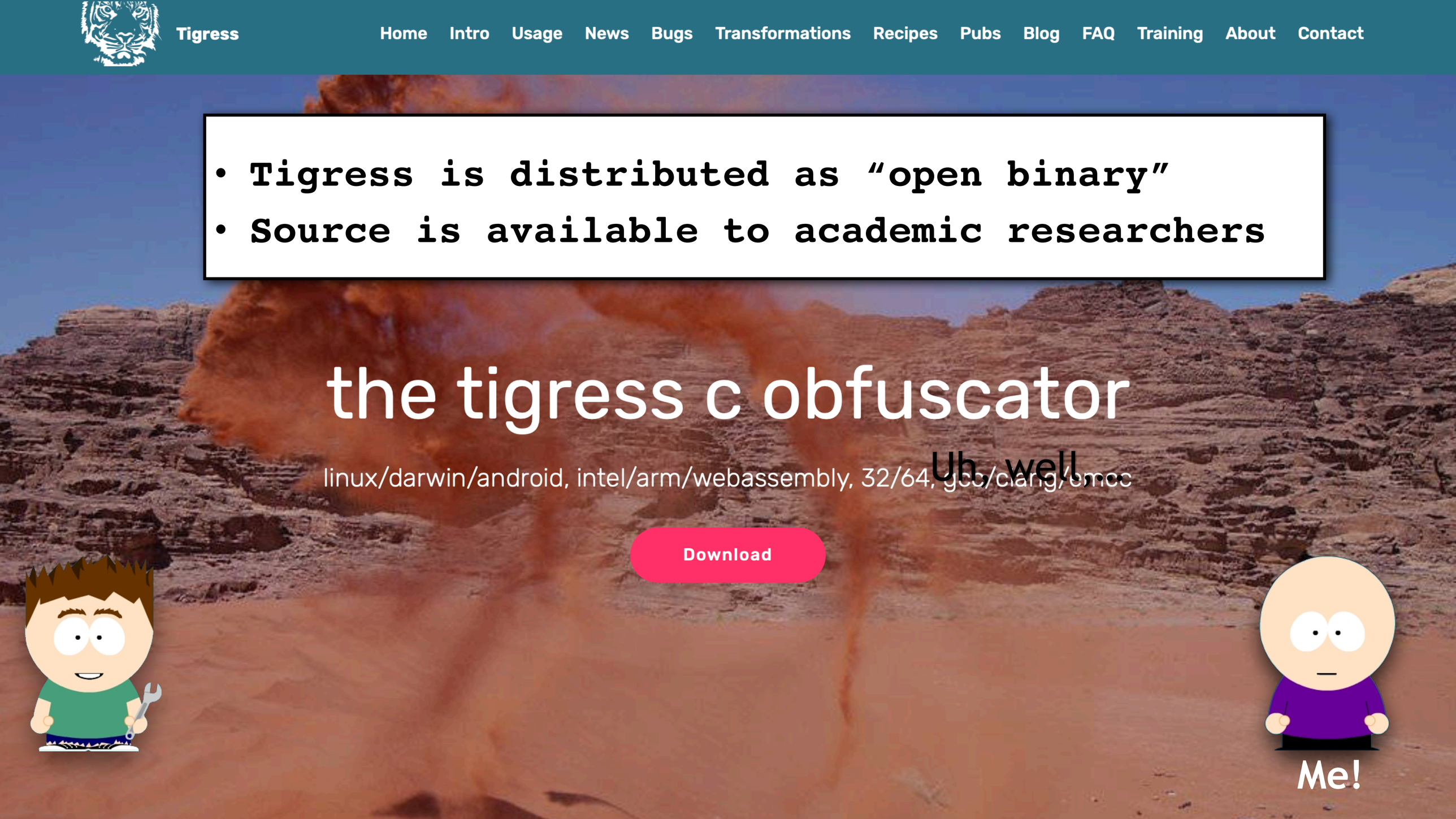
Watermark → Obfuscate → Tamper proof

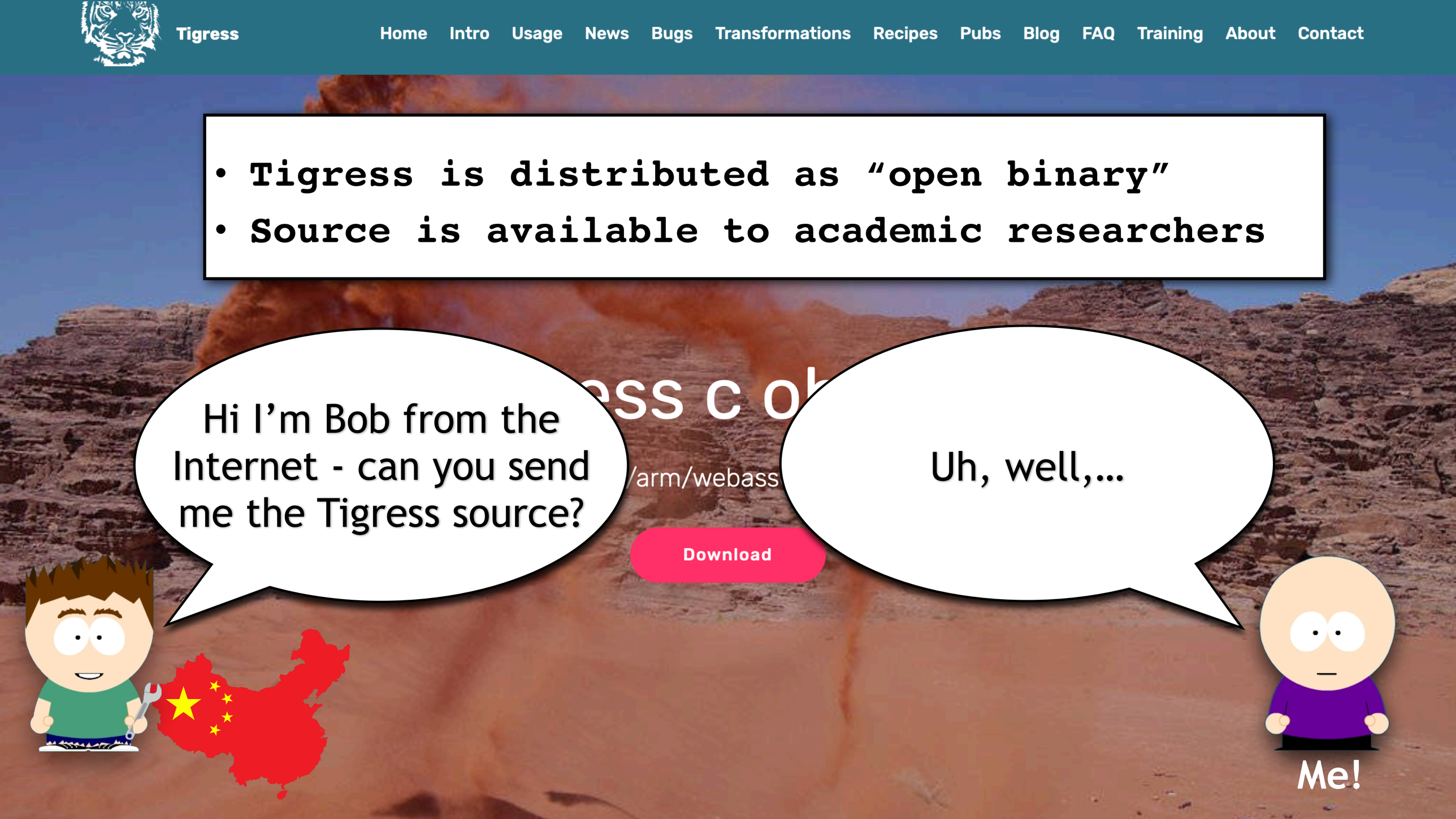
```
}
```

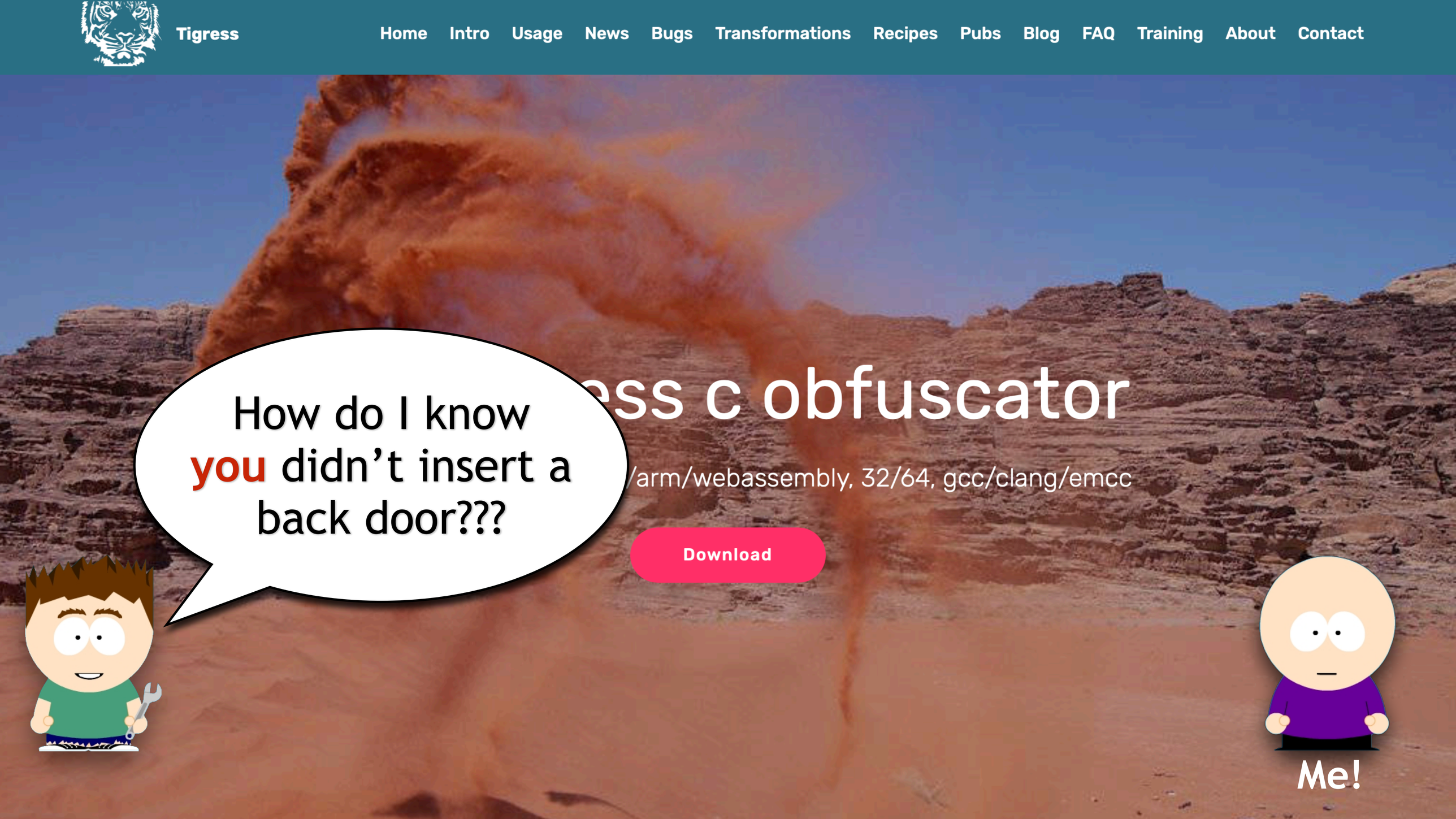
```
main(){
  foo(){
  }
```



```
}
```


Tigress
Home Intro Usage News Bugs Transformations Recipes Pubs Blog FAQ Training About Contact
• Tigress is distributed as “open binary”
• Source is available to academic researchers
the tigress c obfuscator
linux/darwin/android, intel/arm/webassembly, 32/64, gcc/clang/emcc
Download
Uh, well...
Me!


Tigress
Home Intro Usage News Bugs Transformations Recipes Pubs Blog FAQ Training About Contact
• Tigress is distributed as “open binary”
• Source is available to academic researchers
Hi I’m Bob from the Internet - can you send me the Tigress source?
Uh, well,…
Download
Me!


Tigress
Home
Intro
Usage
News
Bugs
Transformations
Recipes
Pubs
Blog
FAQ
Training
About
Contact
ess c obfuscator
/arm/webassembly, 32/64, gcc/clang/emcc
Download
How do I know you didn't insert a back door???
Me!

ss c o

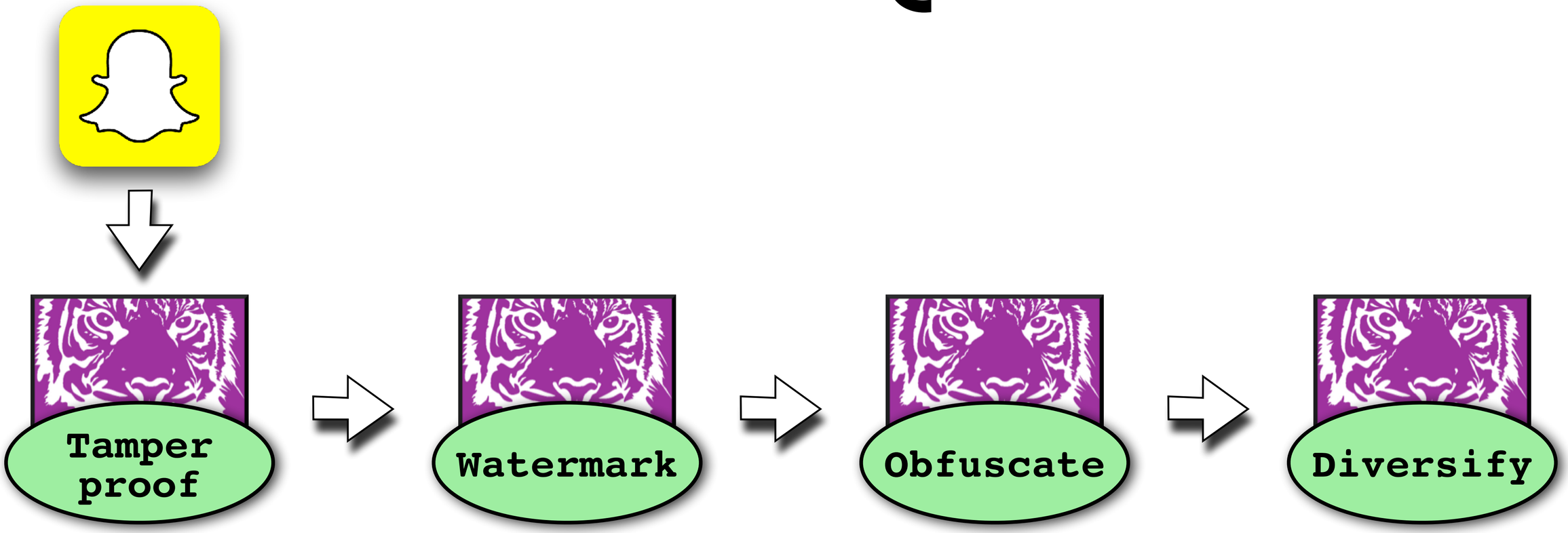
/arm/webass

Download

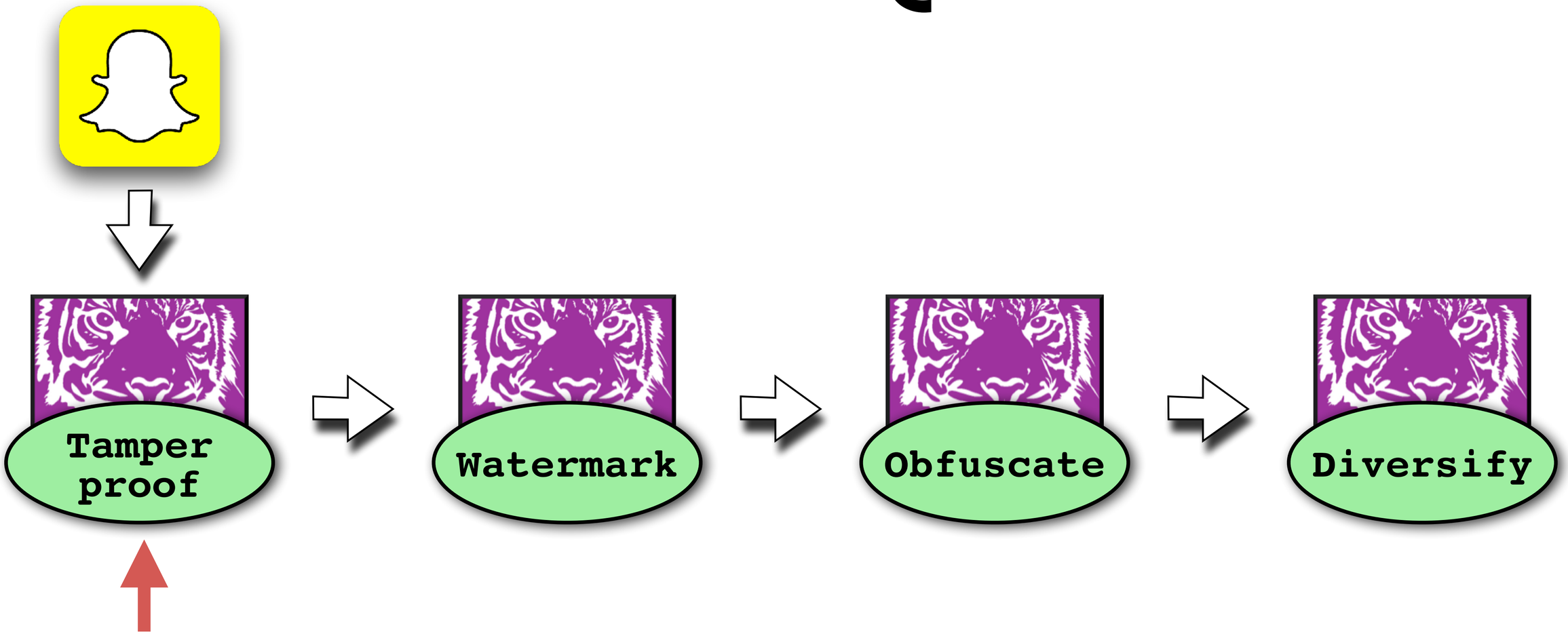
Reverse Program

Research Program

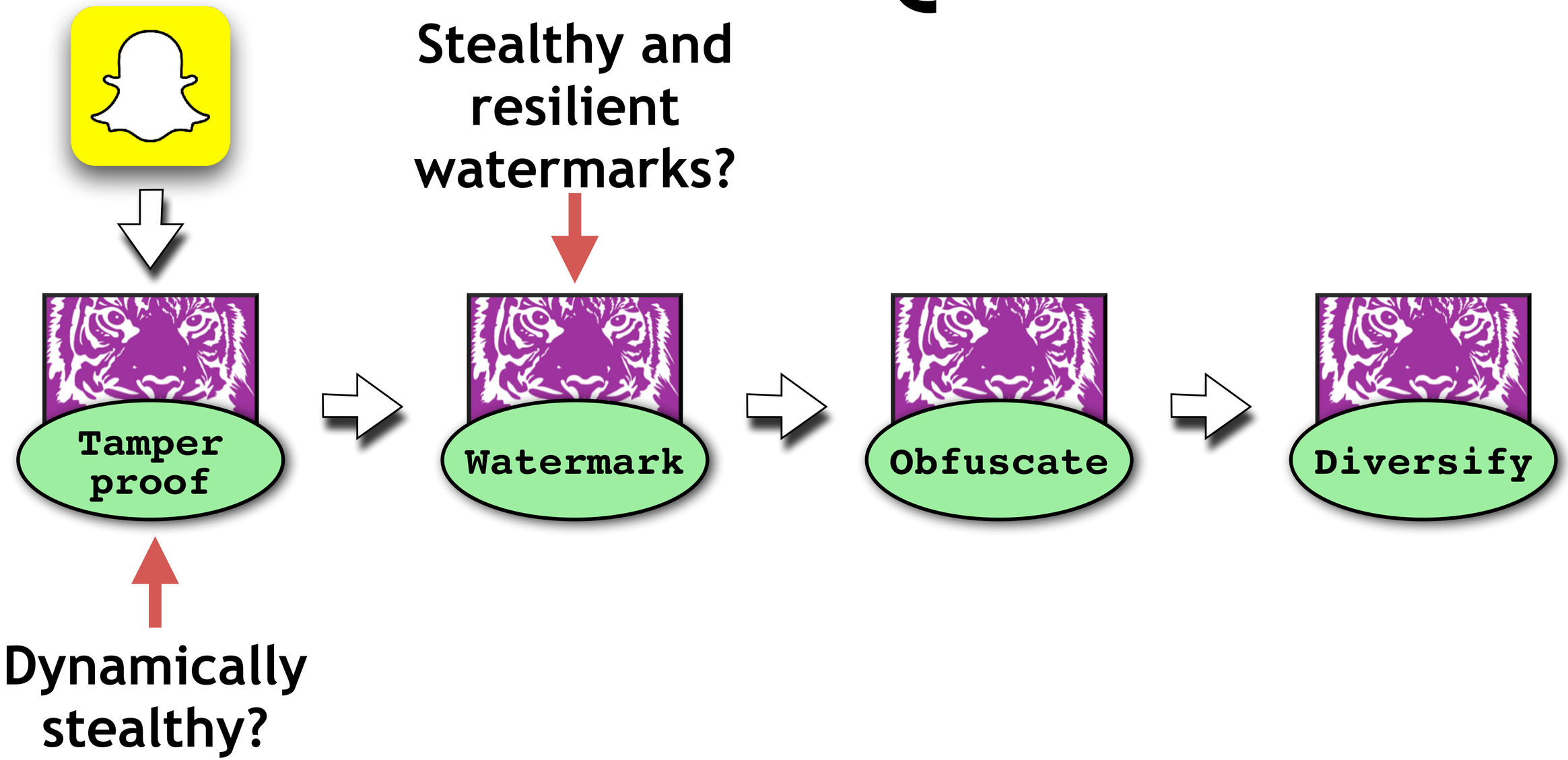
Research Questions



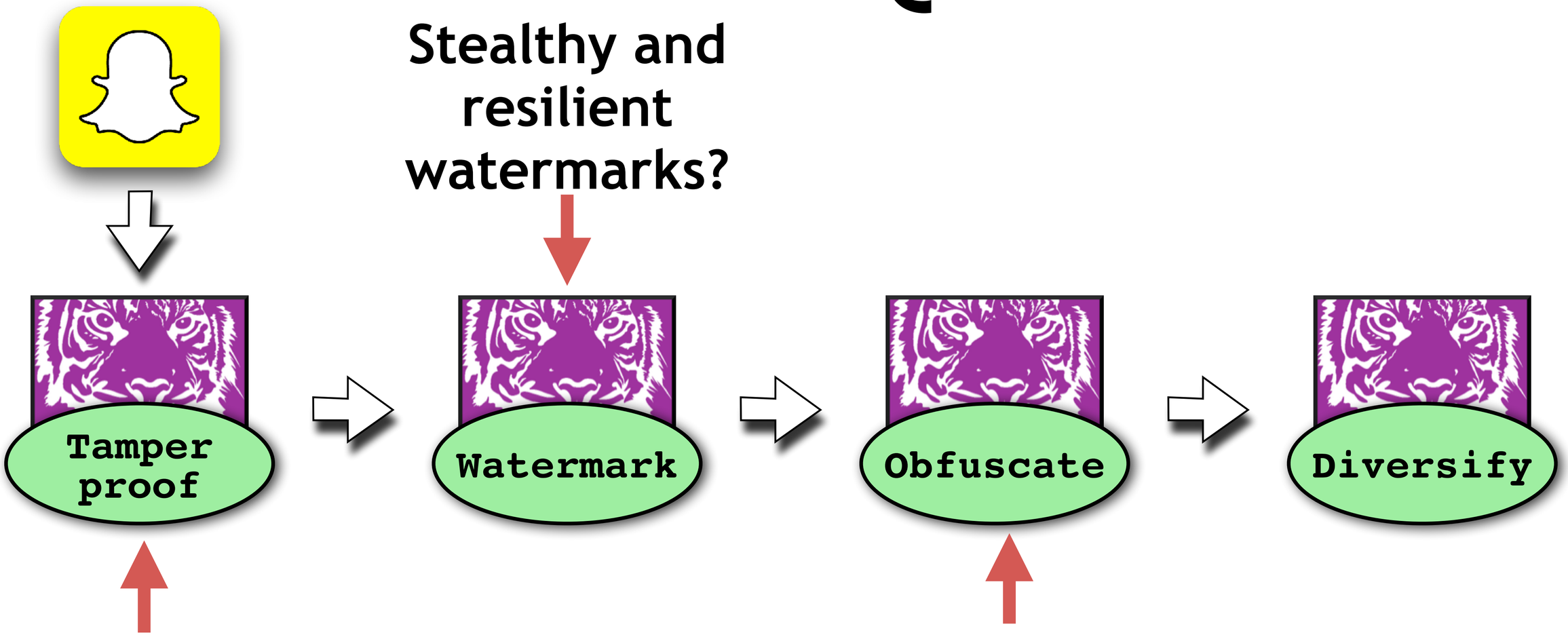
Research Questions



Research Questions



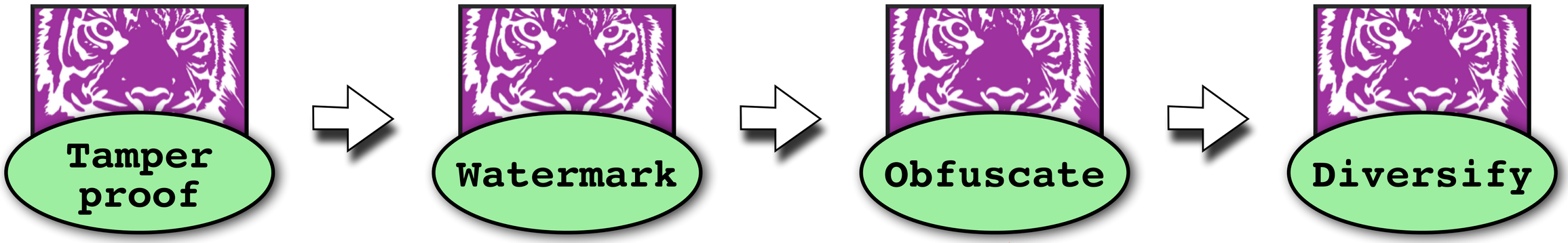
Research Questions



Research Questions

Stealthy and resilient watermarks?

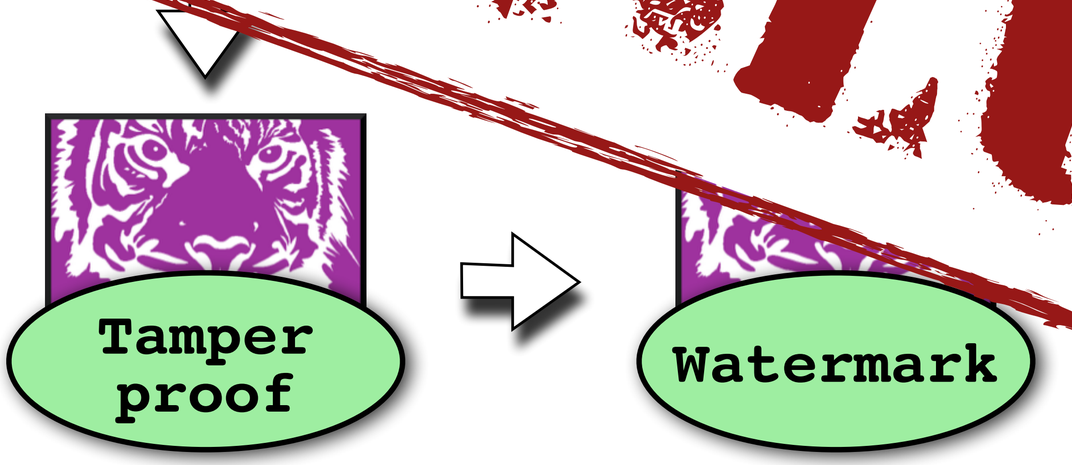
Diversity that actually prevents class attacks?



Dynamically stealthy?

Resilient to modern analysis tools (SMT), low overhead?

Research Questions



EVALUATION!

Grand Reverse Engineering Challenge $10,000 Prize Sum

The Grand Reverse Engineering Challenge runs in two rounds: **Round 1** starts **May 21** and **Round 2** starts **July 3.** The competition ends **midnight, anywhere on earth, July 18**. The total prize sum is USD 10,000. All individuals are welcome to participate. The only requirement is that you run our data collection framework in the background (typically within a Linux virtual machine) as you solve the challenges. The data collected will be published (anonymized, of course) and used to learn how reverse engineers solve problems in practice.

UPDATE: Congrats to Li Xusheng for winning the $5,000 1st price and to Varun Iyer for winning the $2,000 2nd price! Li Xusheng got 7 points for solving challenges 5, 7, 8, and 10, and Varun got 4 points for solving challenges 5, 8, and 10. The rest of the challenges remain unsolved.

Download Round 2

Grand Reverse Engineering Challenge

$10,000 Prize Sum

1. Recruit RE subjects worldwide.
2. Reward subjects with $$.
3. Collect fine-grained attack traces.
4. Analyze traces to derive models of RE behavior.

Download Round 2

Thank You!

collberg.cs.arizona.edu

tigress.wtf

grand-re-challenge.org

Supported by NSF CNS-1525820, SATC-2040206

Thank You!

collberg.cs.arizona.edu

tigress.wtf

grand-re-challenge.org

Supported by NSF CNS-1525820, SATC-2040206

Evaluating

End User

Attacks

Evaluating End User Attacks