Your Teammate Isn’t Human!

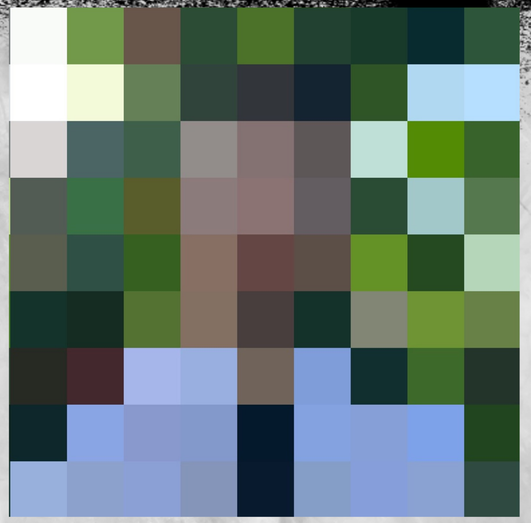
Mixing Decompilation and AI for Modern Reverse Engineering

COMMUNITY 23

@mahaloz

$ whoami

- Zion Leonahenahe Basque

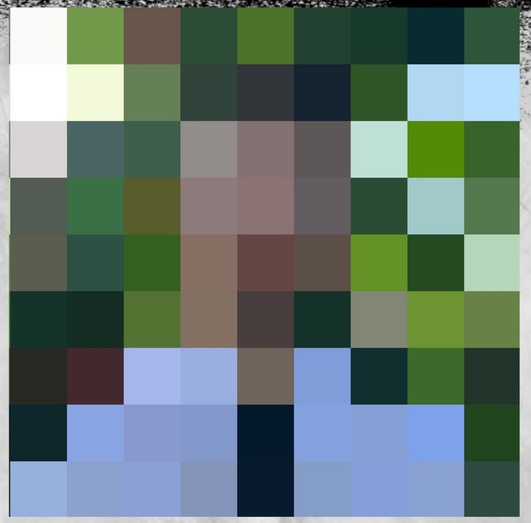


SHELLPHISH

sefcom
security engineering for future computing

$ whoami

- Zion Leonahenahe Basque
- Aka “mahaloz”

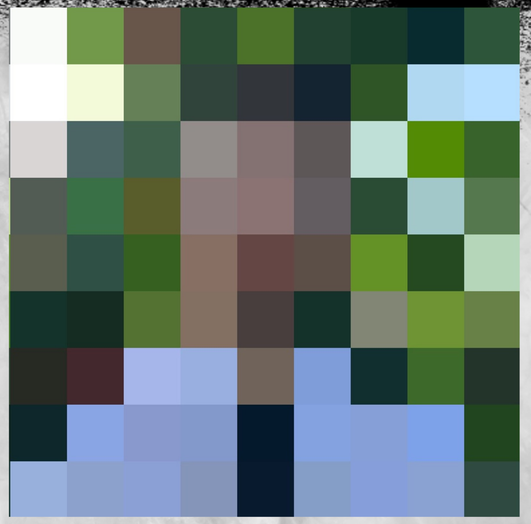


SHELLPHISH

sefcom
security engineering for future computing

$ whoami

- Zion Leonahenahe Basque
- Aka “mahaloz”
- Prev. Co-captain of Shellphish

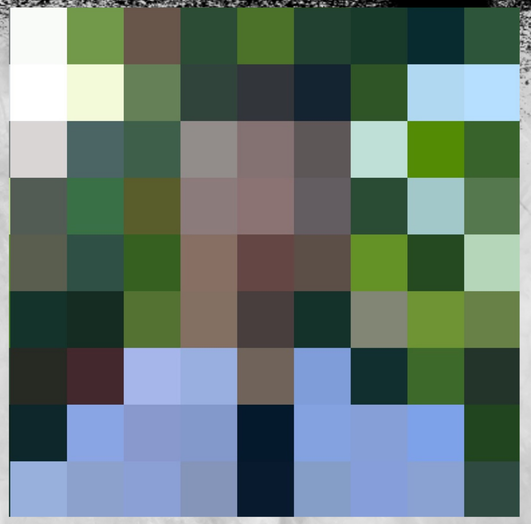


SHELLPHISH

sefcom
security engineering for future computing

$ whoami

- Zion Leonahenahe Basque
- Aka “mahaloz”
- Prev. Co-captain of Shellphish
- PhD Student @ASU SEFCOM
- Decompilers & RE research



SHELLPHISH

sefcom
security engineering for future computing

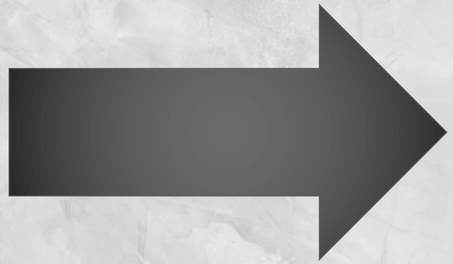
/outline

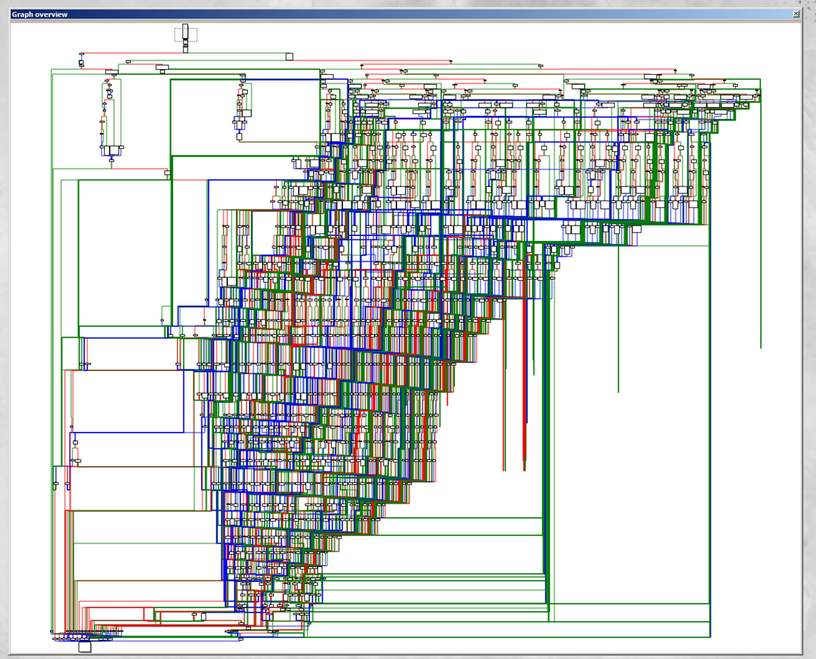
/reversing_binaries

(binary)

/reversing_binaries

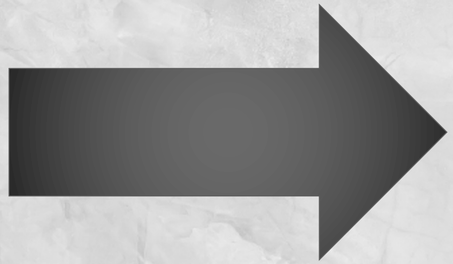
(binary)

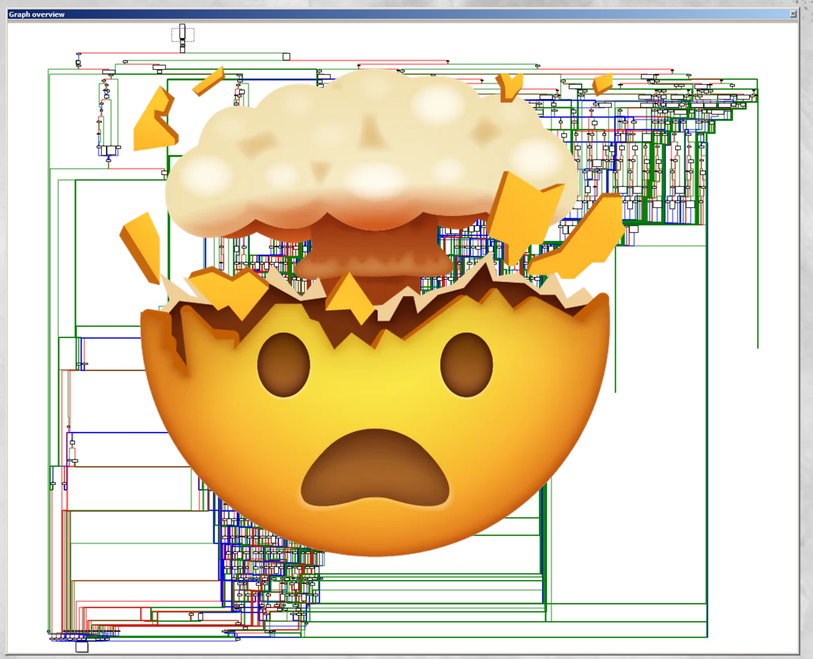




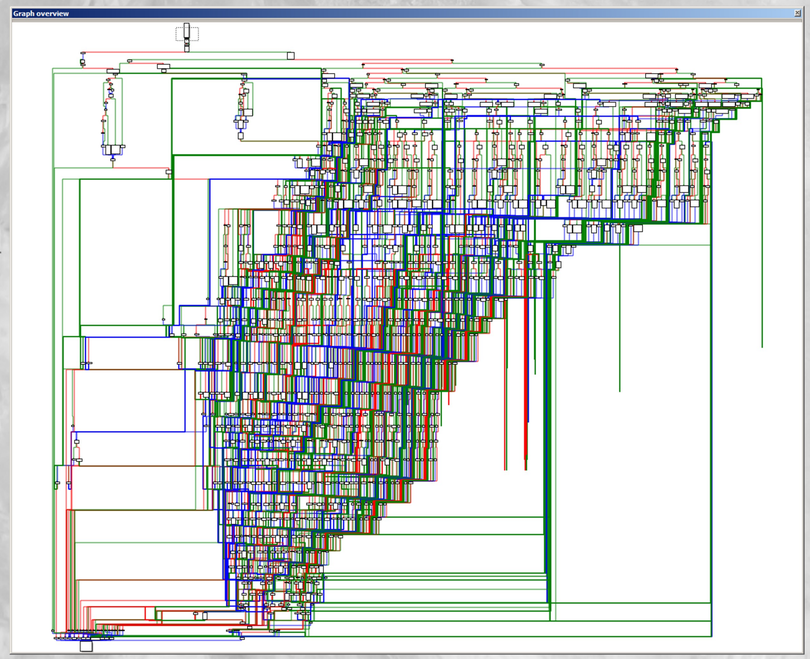
/reversing_binaries

(binary)





/reversing_binaries



```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
{
  v1 = a3;
  v4 = (char *)malloc(a3);
  if ( v4 )
  {
        v5 = v4;
        read(a1, a2, 4uLL);
        v6 = buf[0];
        if ( buf[0] )
        {
            v6 = v5;
            for ( i = 0LL; ; ++i )
            {
        // ...
```

/decompilers

/decompilation_flaws

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
{
  v1 = a3;
  v4 = (char *)malloc(a3);
  if ( v4 )
  {
    v5 = v4;
    read(a1, a2, 4uLL);
    v6 = buf[0];
    if ( buf[0] )
    {
        v6 = v5;
      for ( i = 0LL; ; ++i )
        {
    // ...
```

/decompilation_flaws

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
{
  v1 = a3;
  v4 = (char *)malloc(a3);
  if ( v4 )
  {
    v5 = v4;
    read(a1, a2, 4uLL);
    v6 = buf[0];
    if ( buf[0] )
    {
        v6 = v5;
      for ( i = 0LL; ; ++i )
        {
    // ...
```

/decompilation_flaws

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
{
  v1 = a3;
  v4 = (char *)malloc(a3);
  if ( v4 )
  {
    v5 = v4;
    read(a1, a2, 4uLL);
    v6 = buf[0];
    if ( buf[0] )
    {
        v6 = v5;
      for ( i = 0LL; ; ++i )
        {
    // ...
```

/decompilation_flaws

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
{
  v1 = a3;
  v4 = (char *)malloc(a3);
  if ( v4 )
  {
    v5 = v4;
    read(a1, a2, 4uLL);
    v6 = buf[0];
    if ( buf[0] )
    {
        v6 = v5;
      for ( i = 0LL; ; ++i )
        {
    // ...
```

/decompilation_flaws

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
{
  v1 = a3;
  v4 = (char *)malloc(a3);
  if ( v4 )
  {
    v5 = v4;
    read(a1, a2, 4uLL);
    v6 = buf[0];
    if ( buf[0] )
    {
        v6 = v5;
      for ( i = 0LL; ; ++i )
        {
    // ...
```

/compilation_losses

1. No function symbols
2. No variable symbols
3. No types
4. No comments
5. Obscure control flow structure

/decompilation_flaws

1. No comprehensible names
2. No summarizing info
3. Low-interactability
4. Hard to understand structure

/decompilation_flaws

1. No comprehensible names
2. No summarizing info
3. Low-interactability
4. Hard to understand structure

/fixing_decompilation

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
```

/fixing_decompilation

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
```

/fixing_decompilation

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
```

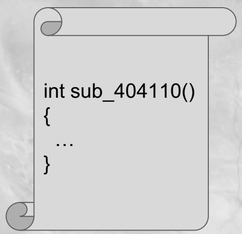
```
int __fastcall recv_data(int socket, char *buf, int size)
```

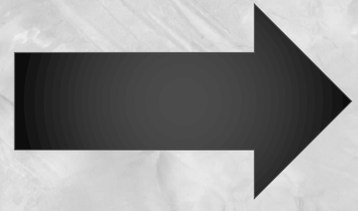
/fixing_decompilation

```
__int64 __fastcall sub_404110(int a1, void *a2, int a3)
```

```
int __fastcall recv_data(int socket, char *buf, int size)
```

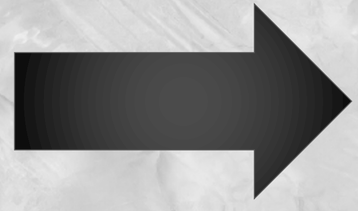
/previous_ai_approaches

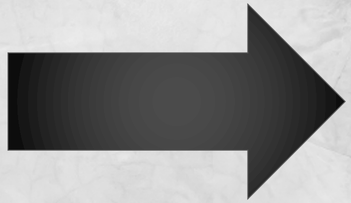




/previous_ai_approaches

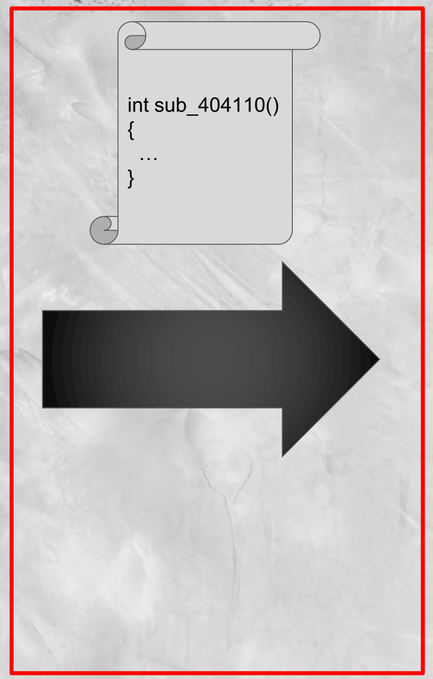
```
int sub_404110()
{
  …
}
```

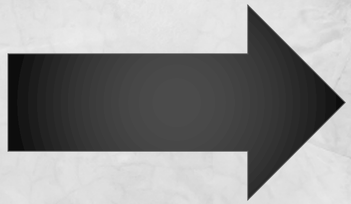




```
/* This code reads from socket and
 * stores the data…
 */
int get_data() { … }
```

/previous_ai_approaches

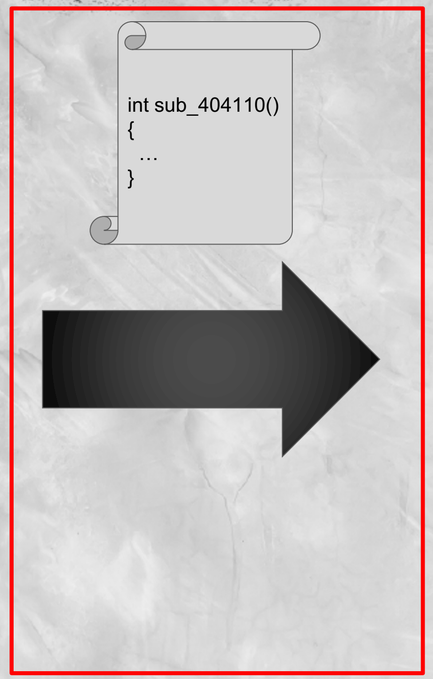




```
/* This code reads from socket and
 * stores the data…
 */
int get_data() { … }
```

Must be below ~80 lines of code

/previous_ai_approaches



Must be below ~80 lines of code

Only function names can be applied, else comment

/previous_ai_approaches

- Gepetto: https://github.com/JusticeRage/Gepetto
  - Easy-to-use, renames functions (and vars now)
- gpt-wpre: https://github.com/moyix/gpt-wpre
  - Scripted, avoids token limit with related summaries

/previous_ai_approaches_flaws

- Decompiler specific

/previous_ai_approaches_flaws

- Decompiler specific
- Non-interactable

/previous_ai_approaches_flaws

- Decompiler specific
- Non-interactable
- Non-scriptable (some)

/previous_ai_approaches_flaws

- Decompiler specific
- Non-interactable
- Non-scriptable (some)
- No variable renaming (some)

/introducing_DAILA

https://github.com/mahaloz/DAILA

/how_DAILA_works

/DAILA_prompts

/DAILA_interactability

DEMO

Questions?

Thanks for listening!