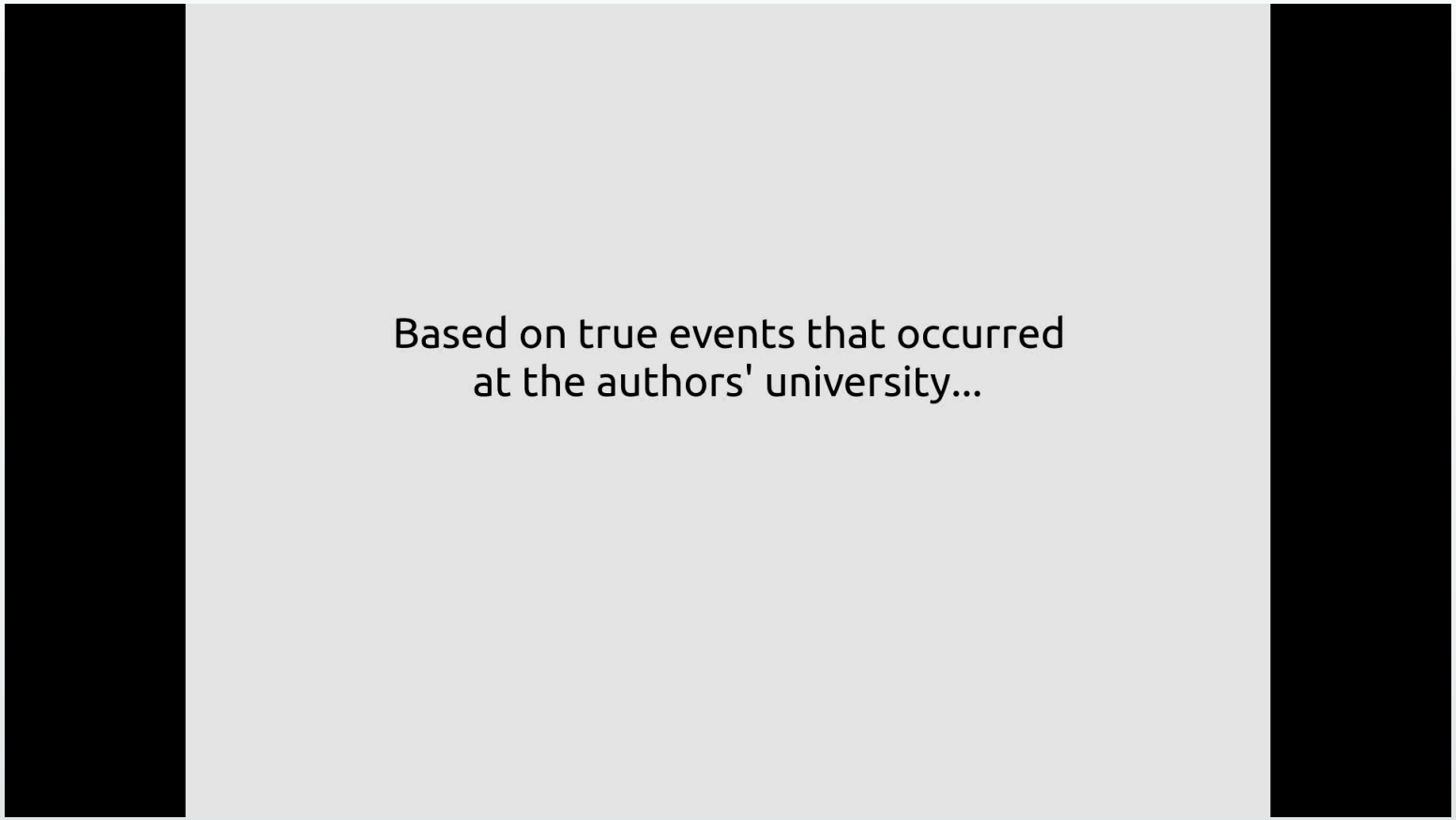
DSCRETE

Automatic Rendering of Forensic Information from Memory Images via Application Logic Reuse

Brendan Saltaformaggio, Zhongshu Gu, Xiangyu Zhang, and Dongyan Xu

Purdue University

A Cyber-Crime

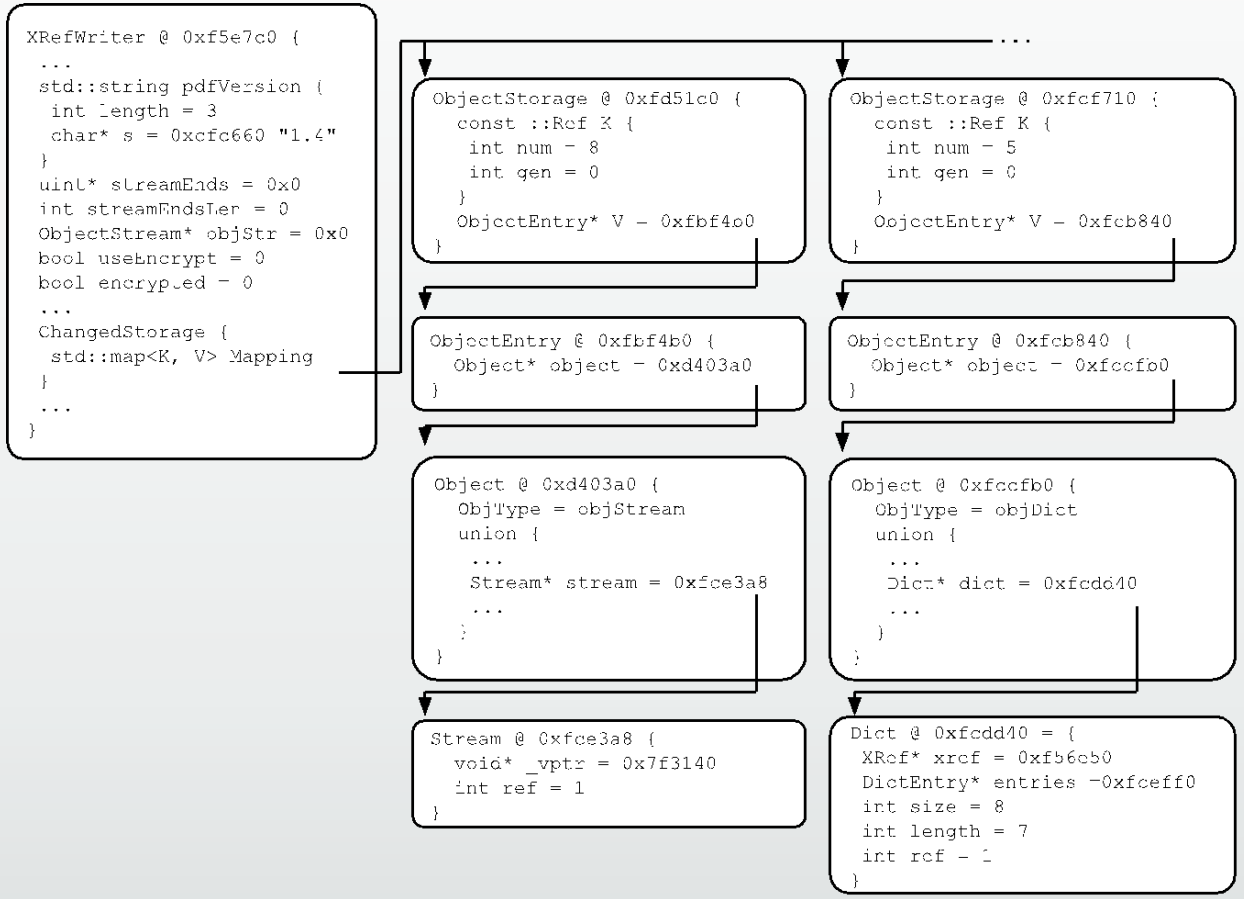


State of the Art … but Limited

Finds raw data structure instances in memory image

Still cannot understand the ***content*** of the data structure!

E.g., images, passwords, formatted/encoded data



Content Reverse Engineering

Observation: Application that defined the data structure contains printing/rendering logic for it too!

Let's call this logic the "P function"

Transforms data structure to formatted application output

Scanner+Renderer Engineering

Input: Data Structure Instance

Output: Formatted Content

Program Code

```
struct pdf* my_pdf;
my_pdf = load_pdf_file(…);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);
exit(0);
```

P Function

```
save_pdf_file(struct pdf* ptr)
{
  char* buf = format_pdf(ptr);
  fwrite(buf, …);
}
```

DSCRETE reuses P to build a scanner+renderer tool

"P Function"

Scanner+Renderer

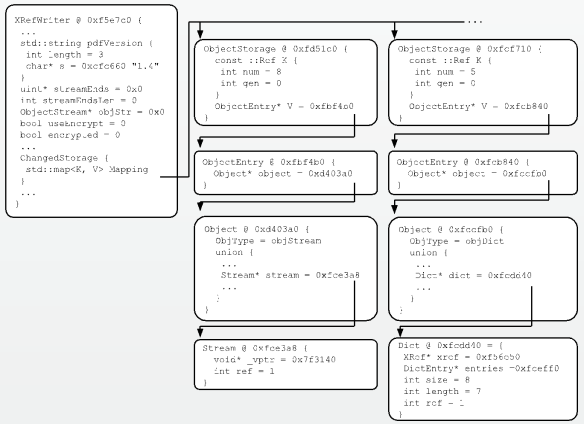
110100000101010
1111010010111000
110100101001010
010001001001111

P Function

```
save_pdf_file(struct pdf* ptr)
{
  char* buf = format_pdf(ptr);
  fwrite(buf, …);
}
```

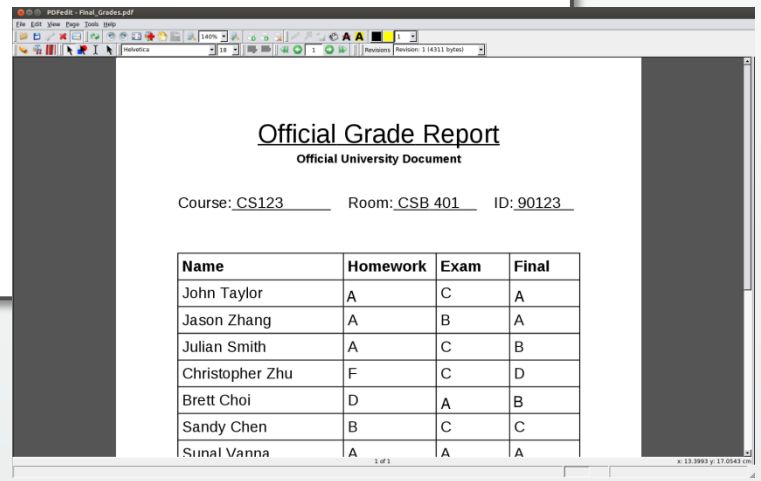
Intuition: Invalid input will crash P

Scanner+Renderer



P Function

```
save_pdf_file(struct pdf* ptr)
{
   char* buf =
   fwrite(buf,
}
```



Official Grade Report

Official University Document

Course: CS123 Room: CSB 401 ID: 90123

Name	Homework	Exam	Final
John Taylor	A	C	A
Jason Zhang	A	B	A
Julian Smith	A	C	B
Christopher Zhu	F	C	D
Brett Choi	D	A	B
Sandy Chen	B	C	C
Supal Vanna	A	A	A

Present every offset of a memory image to P

Valid output is reported

Binary to Scanner+Renderer

In the Forensics Lab, investigators recover the binary from the suspects computer

Based on dynamic binary analysis, DSCRETE then builds a scanner+renderer tool in 2 steps

The resulting scanner+renderer tool can be reused in all future investigations of that application

Step 1: Find the P Function

Execute the binary from the suspect's computer

Slicing techniques find printing/rendering component

- Select which output functions emit the evidence
- E.g. fwrite(…) that saved PDF file

DSCRETE saves a memory snapshot during output function(s)

Step 2: Isolate P's Entry Point

DSCRETE finds "candidates" for the entry point

Candidates must:

1. Take a heap pointer as input
2. All selected output/rendering functions must depend on it

Execute the binary again &

Use **Cross-State Execution** to find correct candidates

Cross-State Execution

Identified Candidate

Program Code

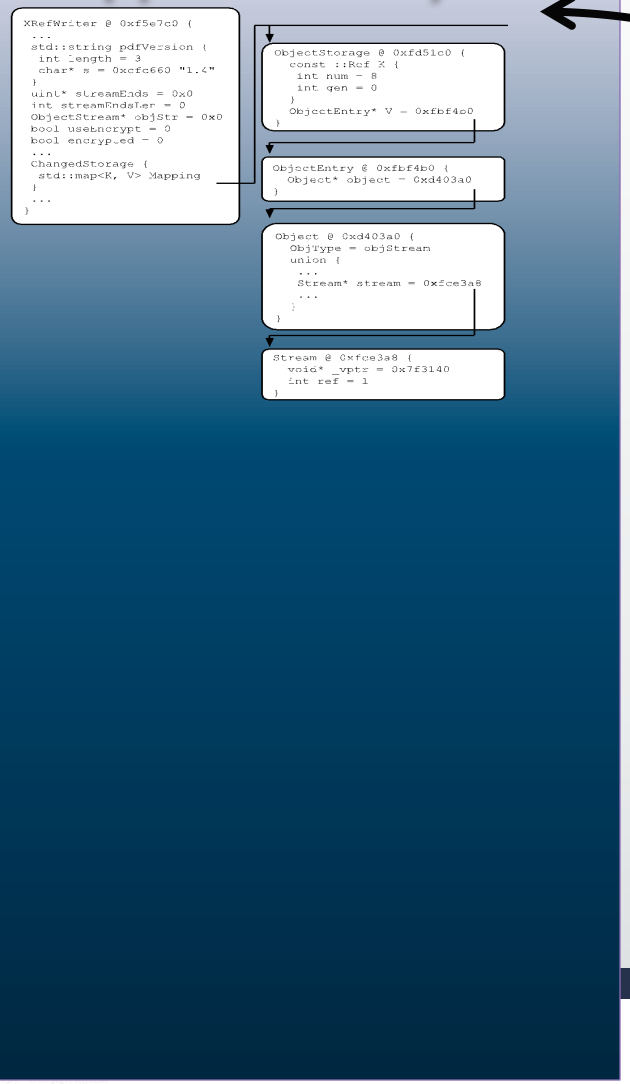
```
struct pdf* my_pdf;
my_pdf = load_pdf_file(…);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);

exit(0);


save_pdf_file(struct pdf* ptr)
{
  char* buf = format(ptr);
  fwrite(buf, …);
}
```

Cross-State Execution

App's Memory



Program Code

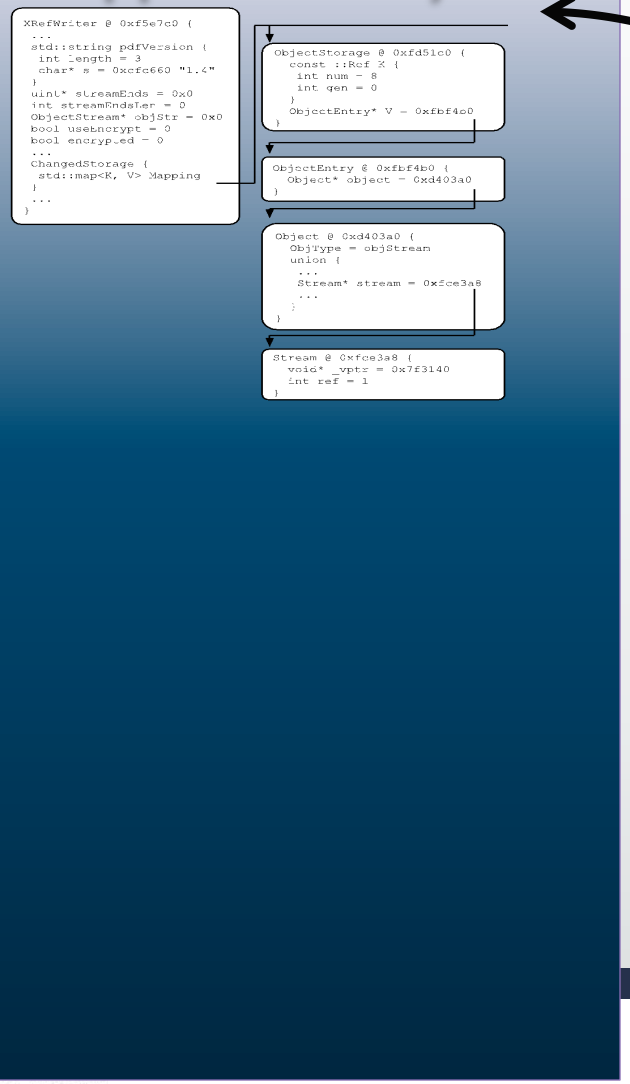
```
struct pdf* my_pdf;
my_pdf = load_pdf_file(…);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);

exit(0);


save_pdf_file(struct pdf* ptr)
{
  char* buf = format(ptr);
  fwrite(buf, …);
}
```

Cross-State Execution

App's Memory



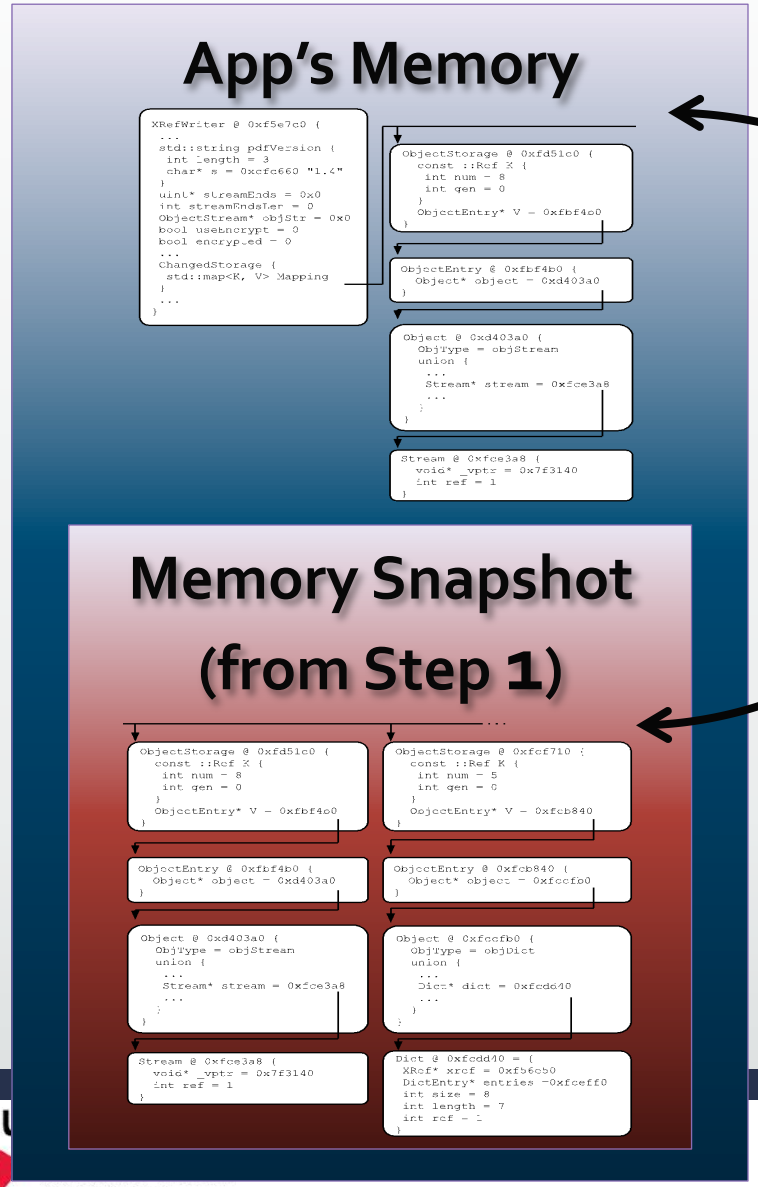
Program Code

```
struct pdf* my_pdf;
my_pdf = load_pdf_file(…);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);

exit(0);


save_pdf_file(struct pdf* ptr)
{
  char* buf = format(ptr);
  fwrite(buf, …);
}
```

Cross-State Execution



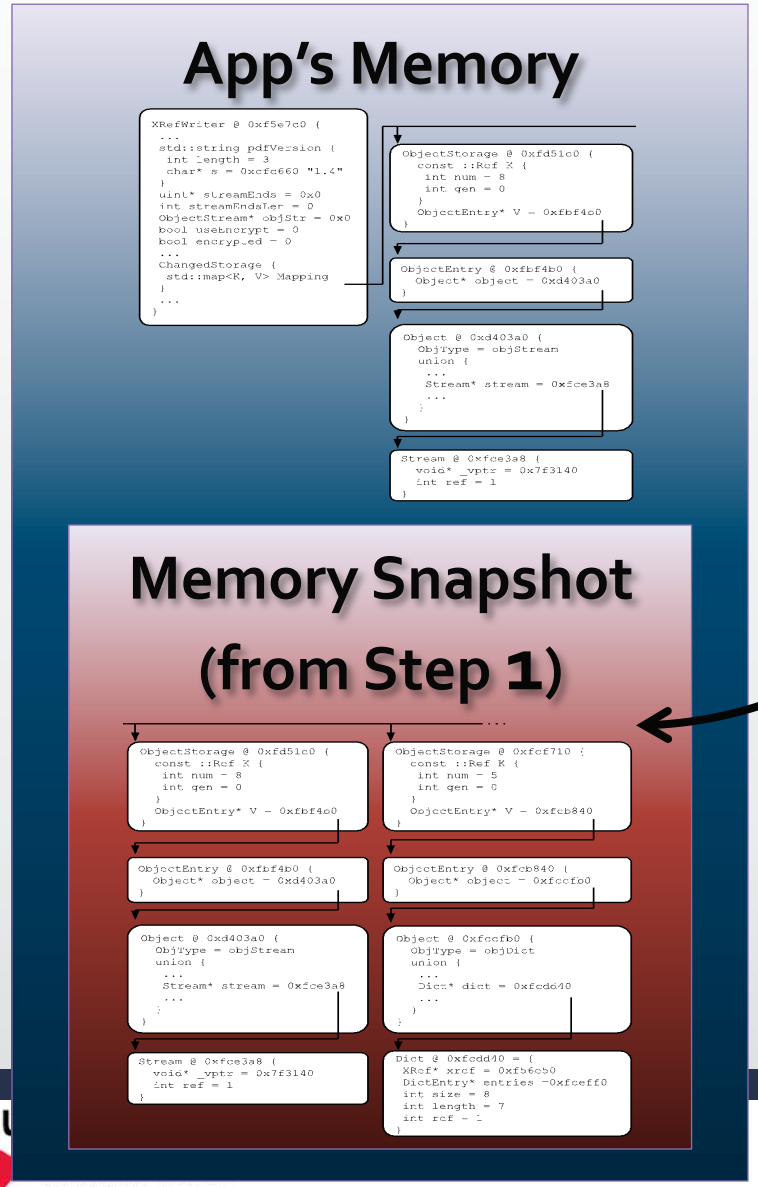
Program Code

```
struct pdf* my_pdf;
my_pdf = load_pdf_file(…);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);
```

Begin Cross-State Execution!

1. Map in memory snapshot
2. Swap my_pdf pointer

Cross-State Execution



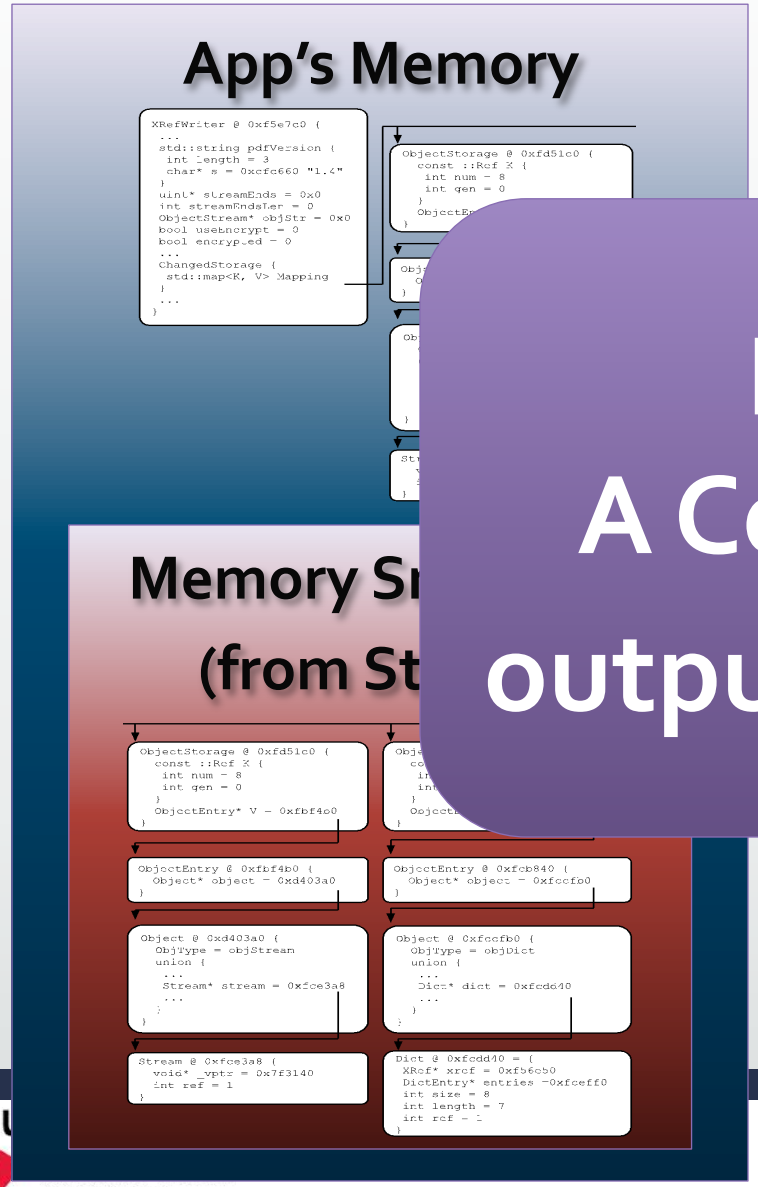
Program Code

```
struct pdf* my_pdf;
my_pdf = load_pdf_file(…);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);

exit(0);


save_pdf_file(struct pdf* ptr)
{
   char* buf = format(ptr);
   fwrite(buf, …);
}
```

Cross-State Execution



Program Code

```
struct pdf* my_pdf;
                                    …);
                                    ser edits PDF


                                    f* ptr)

  char* buf = format(ptr);
  fwrite(buf, …);
}
```

Key Observation:
A Correct Candidate will output the PDF from Step 1

Reused Application Logic

Correct candidate is packed into scanner+renderer tool

Presents each offset in suspect's memory image to P

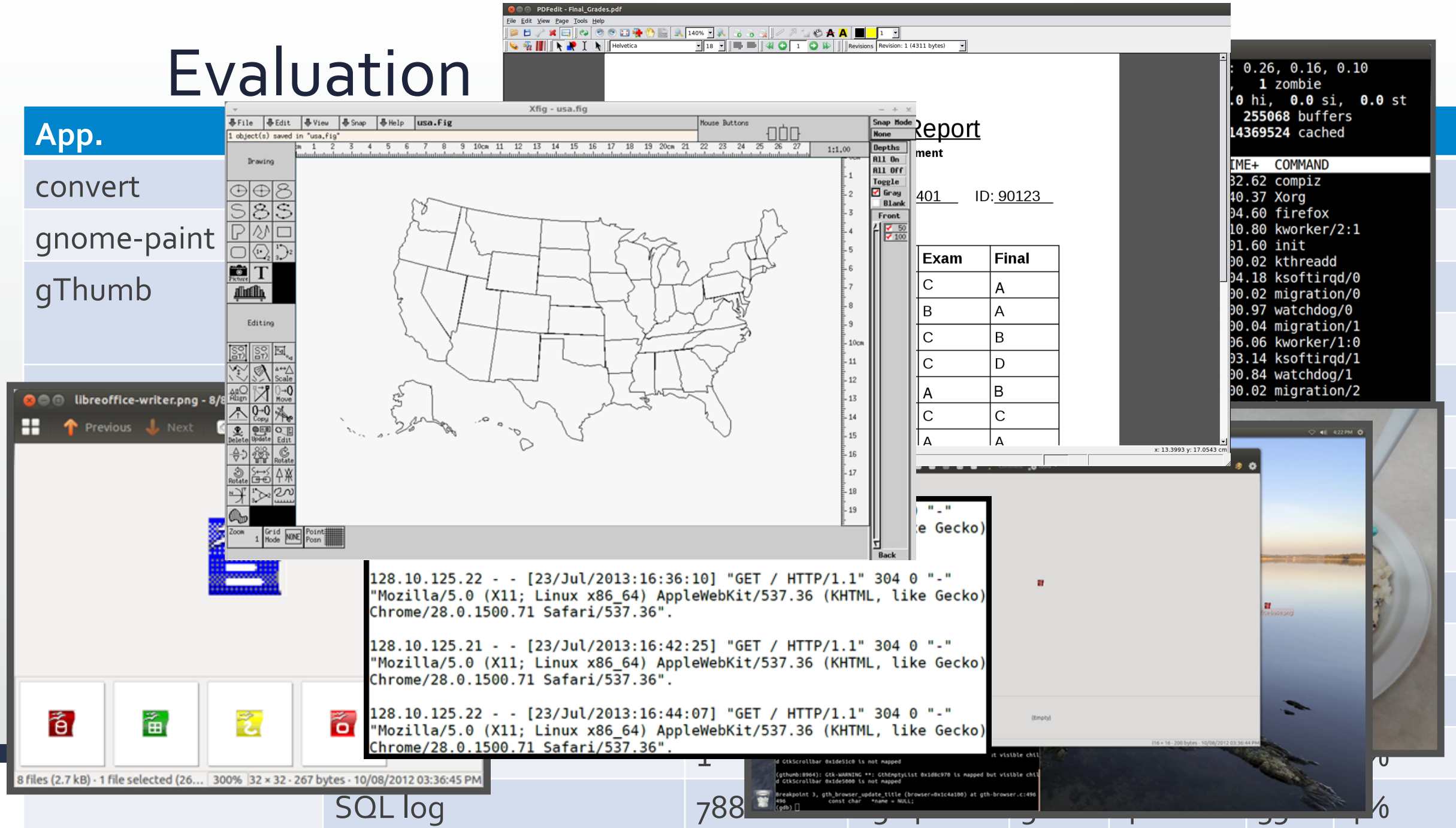
- Reports natural application output as evidence

This tool can be used in all future investigations of this app.

Let’s catch that criminal…

Evaluation

App.
convert
gnome-paint
gThumb
SQL log



Conclusion

Identified the Content Reverse Engineering problem in forensics

DSCRETE leverages binary logic reuse to automatically locate data structures in memory images and reverse engineer content

Highly effective at recovering many forms of digital evidence

Lots of opportunities to improve and apply DSCRETE, and many future research directions!

Thank you!

Questions?

Brendan Saltaformaggio
bsaltafo@cs.purdue.edu