

# 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

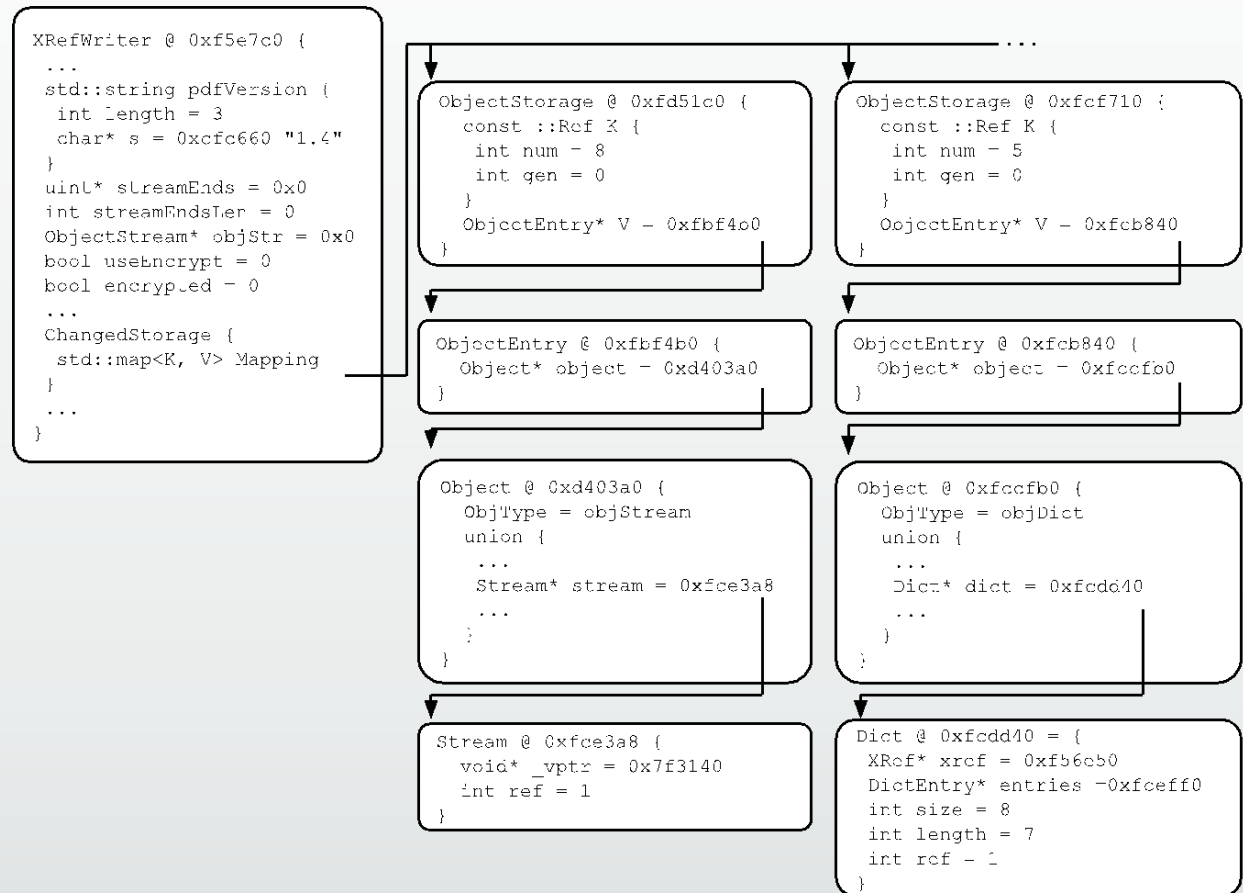
Based on true events that occurred  
at the authors' university...

# 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

# Content-Renderer Engineering

Input: Data Structure Instance

Output: Formatted Content

DSCRETE reuses P to build a scanner+renderer tool

## 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, ...);  
}
```

# Scanner+Renderer

110100000101010  
1111010010111000  
110100101001010  
010001001001111

## P Function

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



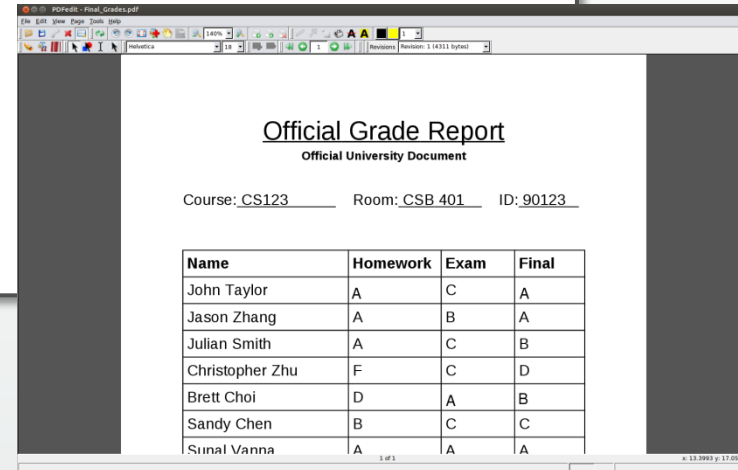
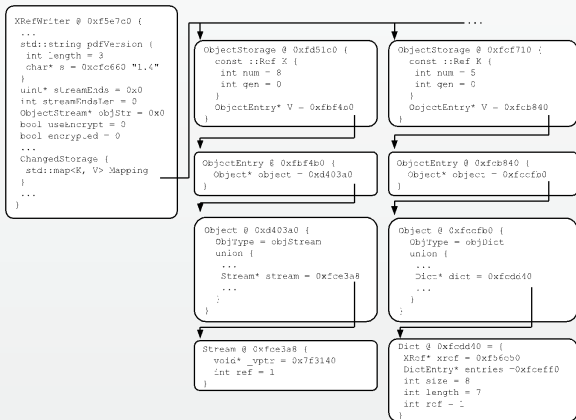
Intuition: Invalid input will crash P

# Scanner+Renderer

## P Function

```
save_pdf_file(struct pdf* ptr)
```

```
{  
    char* buf =  
    fwrite(buf,  
}
```



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

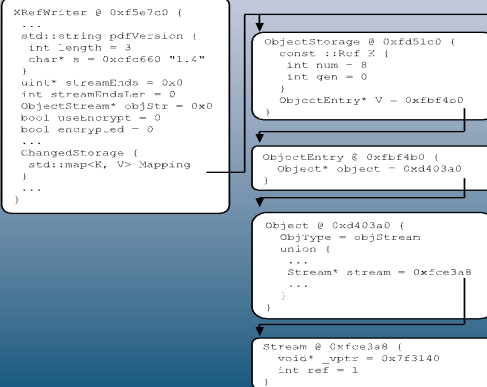
```
struct pdf* my_pdf;
my_pdf = load_pdf_file(...);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);

exit(0);

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

# Cross-State Execution

## App's Memory

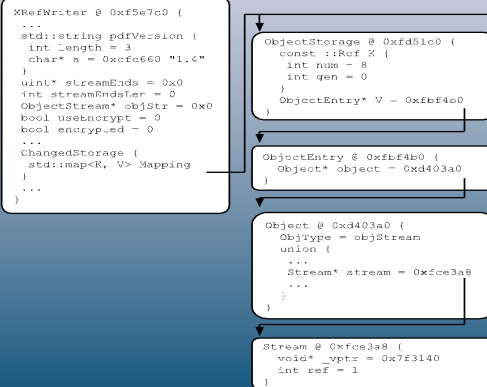


## Program Code

```
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

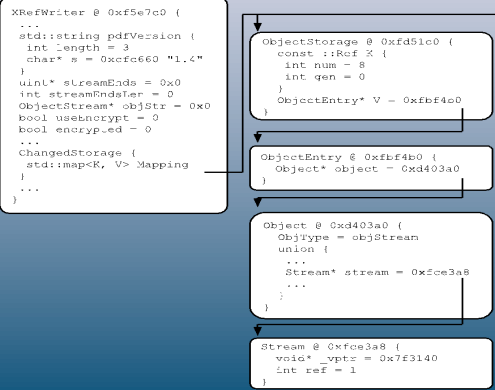


## Program Code

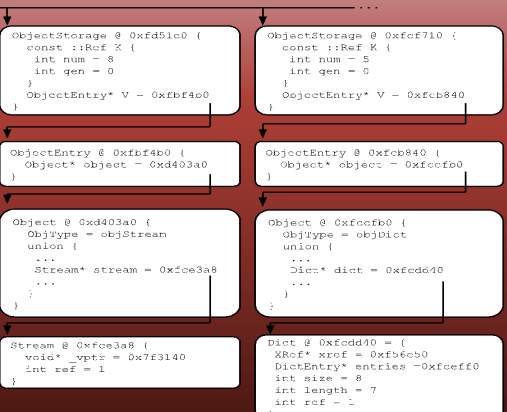
```
struct pdf* my_pdf;  
my_pdf = load_pdf_file(...);  
main_loop(my_pdf); // User edits PDF  
save_pdf_file(my_pdf);  
  
exit(0);  
  
save_pdf_file(struct pdf* ptr)  
{  
    char* buf = format(ptr);  
    fwrite(buf, ...);  
}
```

# Cross-State Execution

## App's Memory



## Memory Snapshot (from Step 1)



## Program Code

```

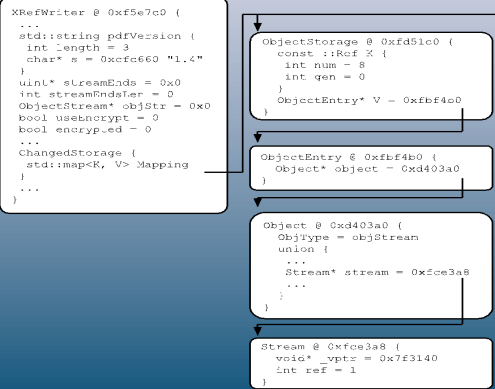
struct pdf* my_pdf;
my_pdf = load_pdf_file(...);
main_loop(my_pdf); // User edits PDF
save_pdf_file(my_pdf);
    
```

**Begin Cross-State Execution!**

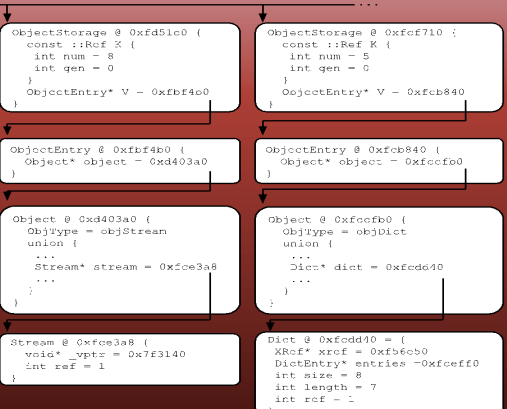
1. Map in memory snapshot
2. Swap my\_pdf pointer

# Cross-State Execution

## App's Memory



## Memory Snapshot (from Step 1)



## 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);

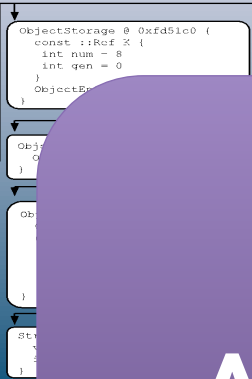
save_pdf_file(struct pdf* ptr)
{
  char* buf = format(ptr);
  fwrite(buf, ...);
}
    
```

# Cross-State Execution

## App's Memory

```

XRefWriter @ 0xf5e7c0 (
...
std::string pdfVersion (
int length = 3
char* s = 0xfcc6c0 "1.4"
)
uint* streamEnds = 0x0
int streamEndsLen = 0
ObjectStream* objStr = 0x0
bool useEncrypt = 0
bool encryptEd = 0
...
ChangedStorage (
std::map<R, V> Mapping
)
...
)
    
```



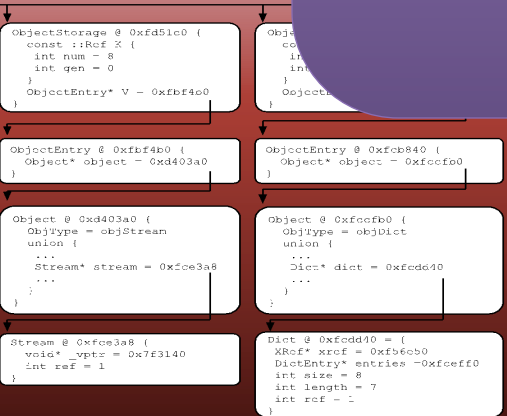
## Program Code

```

struct pdf* my_pdf;
...);
... user edits PDF
    
```

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

## Memory State (from Step 1)



```

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



# 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...

Back at the Forensics Lab...

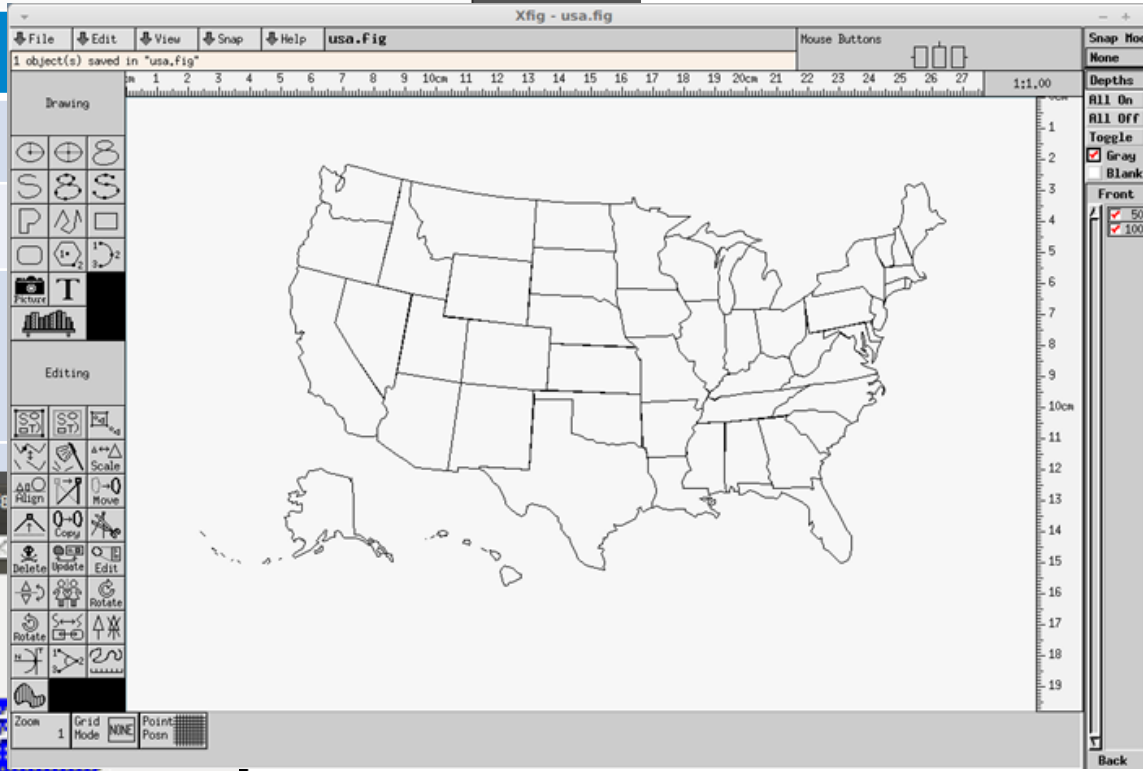
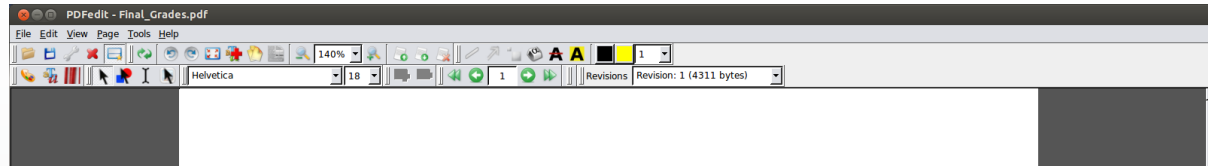
# Evaluation

## App.

convert

gnome-paint

gThumb



## Report

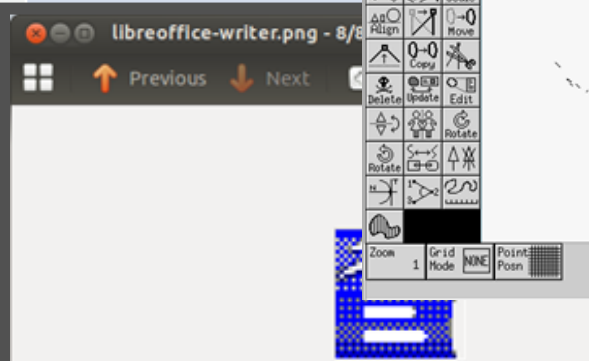
ment

401 ID: 90123

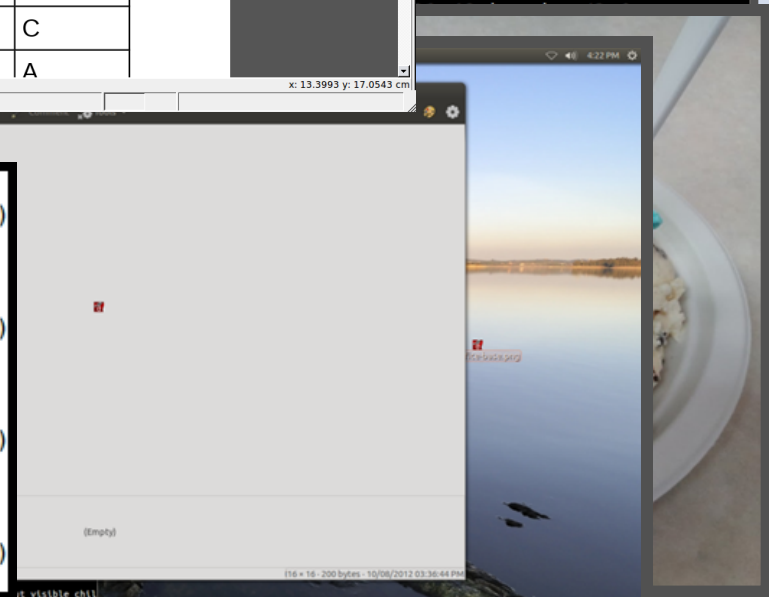
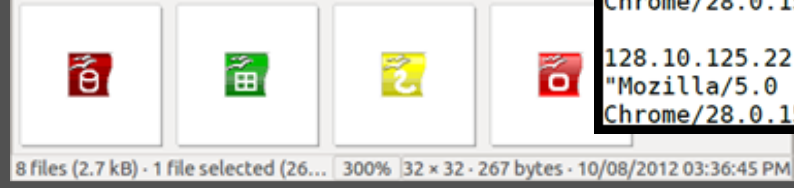
Exam	Final
C	A
B	A
C	B
C	D
A	B
C	C
A	A

```
: 0.26, 0.16, 0.10  
1 zombie  
0 hi, 0.0 si, 0.0 st  
255068 buffers  
14369524 cached
```

TIME+	COMMAND
32.62	compiz
40.37	Xorg
94.60	firefox
10.80	kworker/2:1
91.60	init
90.02	kthreadd
94.18	ksoftirqd/0
90.02	migration/0
90.97	watchdog/0
90.04	migration/1
96.06	kworker/1:0
93.14	ksoftirqd/1
90.84	watchdog/1
90.02	migration/2



```
128.10.125.22 - - [23/Jul/2013:16:36:10] "GET / HTTP/1.1" 304 0 "-"  
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/28.0.1500.71 Safari/537.36".  
  
128.10.125.21 - - [23/Jul/2013:16:42:25] "GET / HTTP/1.1" 304 0 "-"  
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/28.0.1500.71 Safari/537.36".  
  
128.10.125.22 - - [23/Jul/2013:16:44:07] "GET / HTTP/1.1" 304 0 "-"  
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/28.0.1500.71 Safari/537.36".
```



SQL log

788

```
GtkScrollbar @xid51c0 is not mapped  
it visible child  
(gthumb:8904): Gtk-WARNING **: GtkEmptyList @xidc970 is mapped but visible child  
GtkScrollbar @xid5990 is not mapped  
Breakpoint 3, gth_browser_update_title (Browser=@x1c4a100) at gth-browser.c:496  
496 const char *name = NULL;  
(gdb) █
```

# 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