Steal this Movie

Automatically Bypassing DRM Protection in Streaming Media Services

Ruoyu 'Fish' Wang[12], Yan Shoshitaishvili[1],

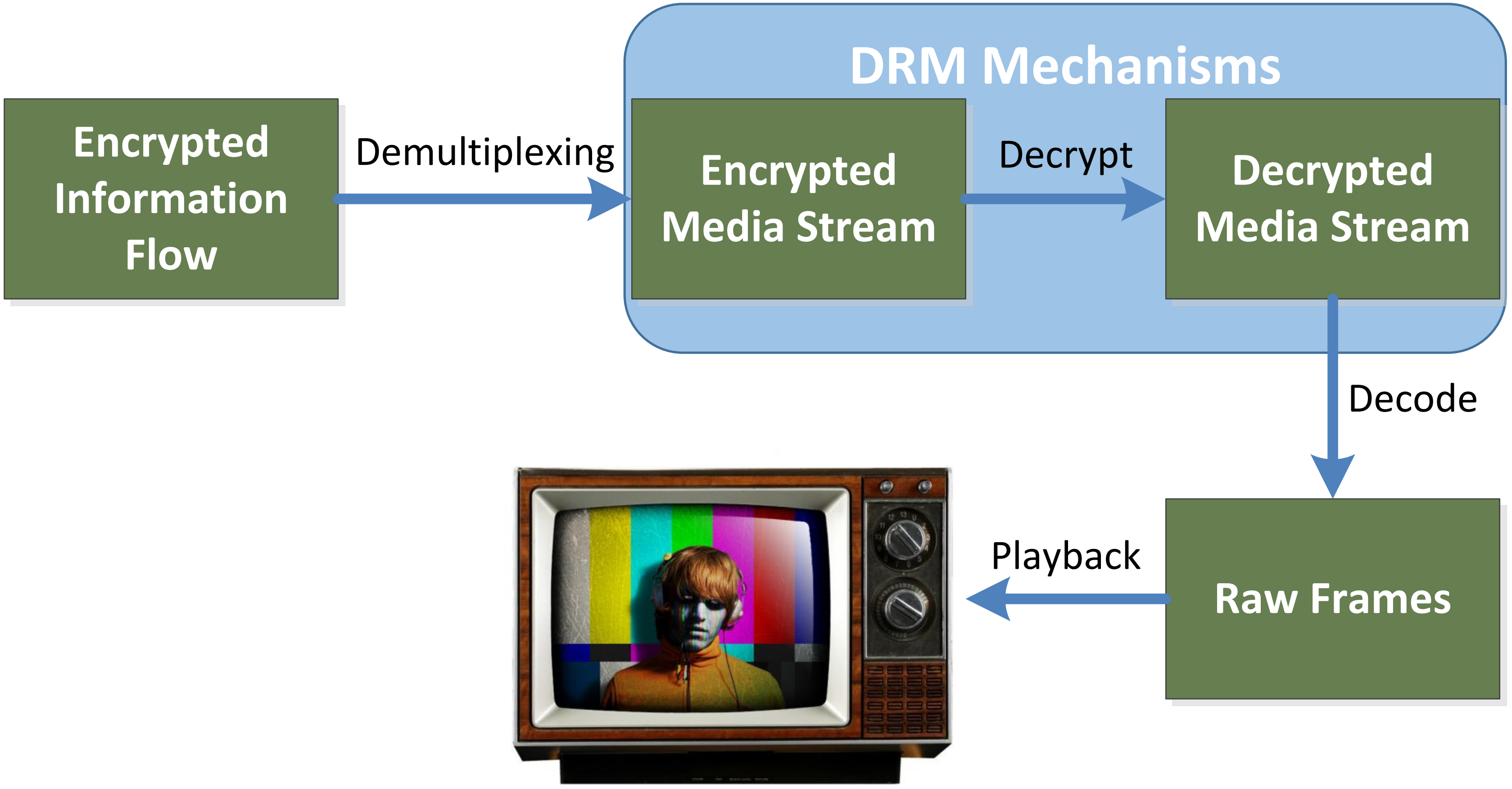
Christopher Kruegel[1], Giovanni Vigna[1]

[1] UC Santa Barbara

[2] Tsinghua Unversity

Digital Rights Management

How DRM works



DRM Mechanisms

Encrypted Information Flow

Demultiplexing

Encrypted Media Stream

Decrypt

Decrypted Media Stream

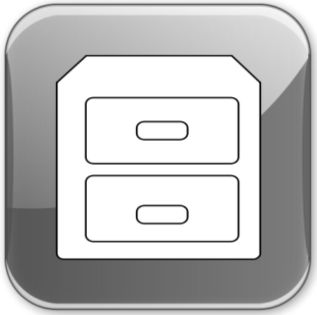
Decode

Raw Frames

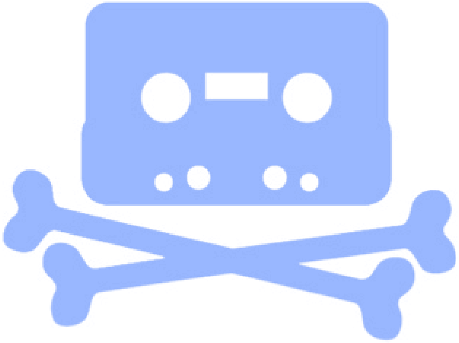
Playback

DRM bypasses

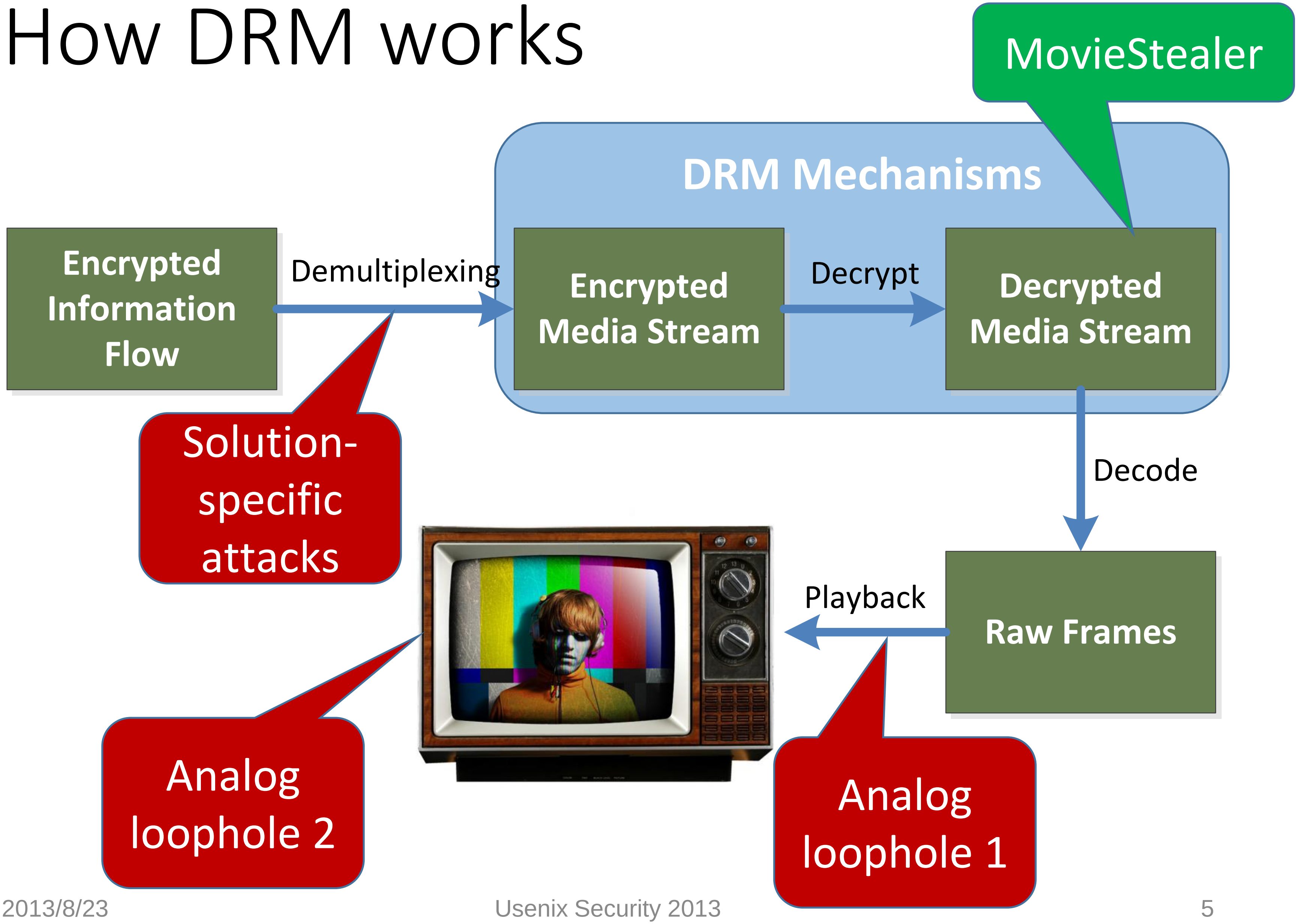
- What for?



- Solution-specific
  - DeCSS - DVD Jon
  - Despotify
  - HDCP master key leaking
- Analog loophole



How DRM works



Automatic attacking

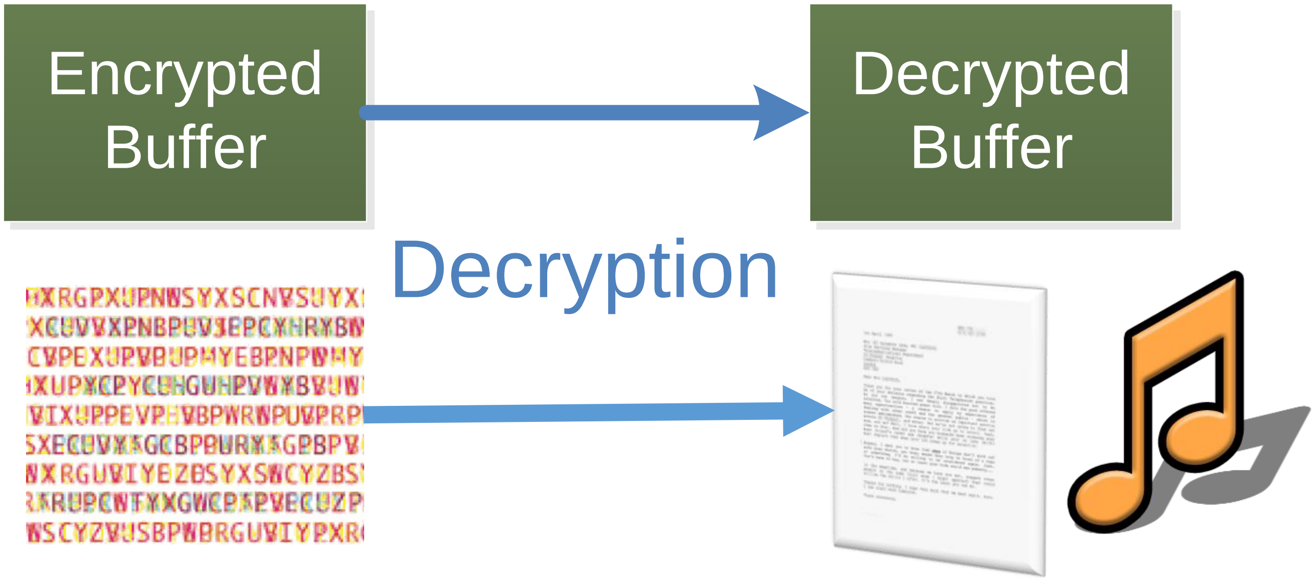
Challenges

- Complexity

- Performance

- Generality

Intuitions



Encrypted Buffer

Decrypted Buffer

Decryption

Final goal:
Identify the decrypted stream and dump it!

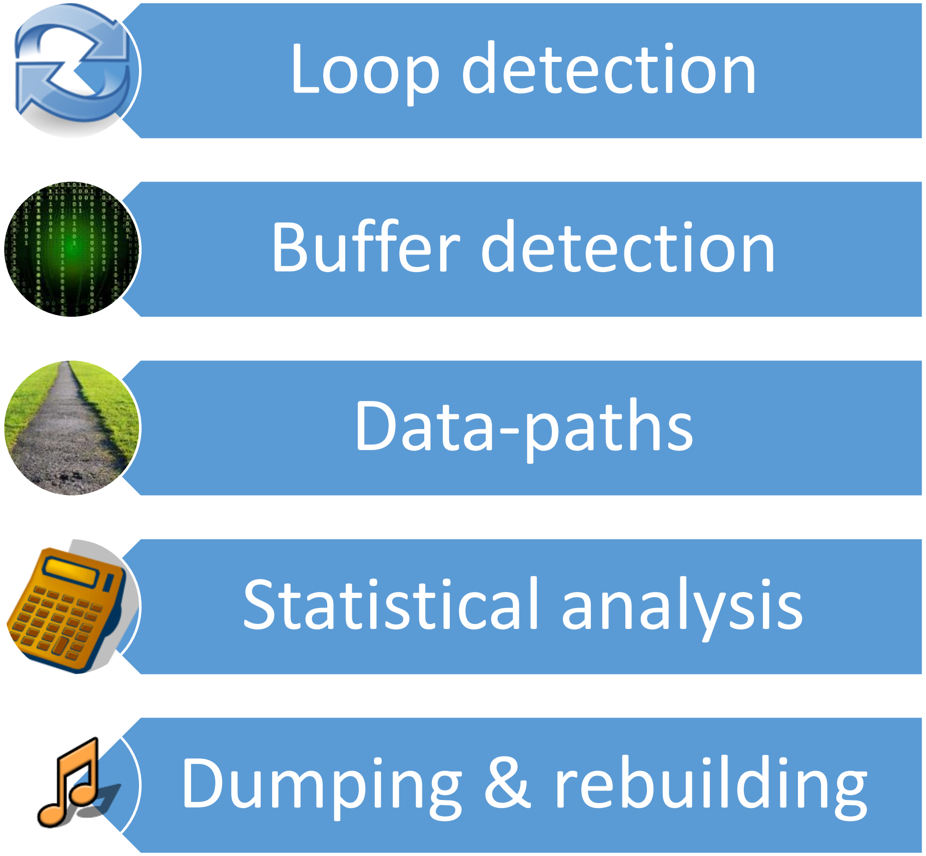
Overview

- MovieStealer design & optimizations
- Experimental results
- Countermeasures
- Ethics and legality

MovieStealer Design

Approach overview

Goal: find the decrypted stream!



Loop detection

Buffer detection

Data-paths

Statistical analysis

Dumping & rebuilding

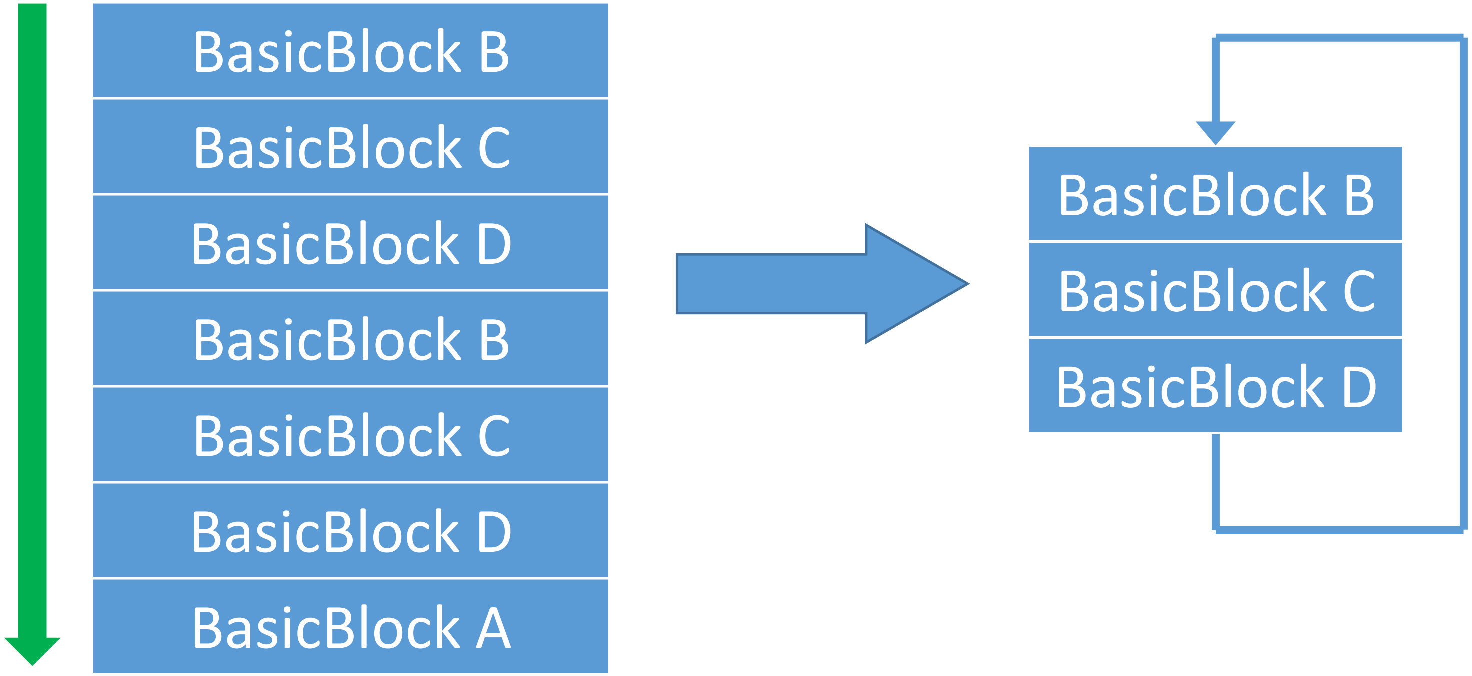
Approach overview

Goal: find the decrypted stream!

- Loop detection
- Buffer detection
- Data-paths
- Statistical analysis
- Dumping & rebuilding

Loop detection (1)

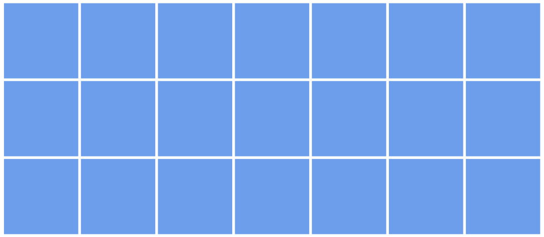
- Based on LoopProf*



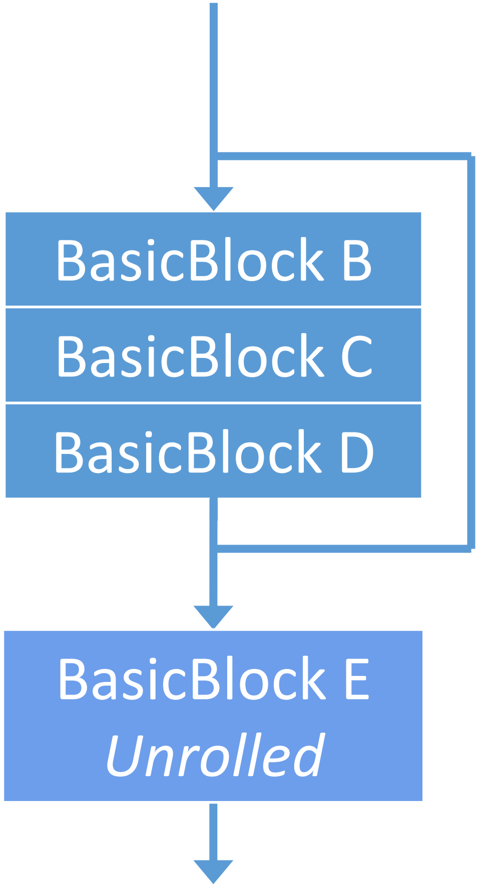
**LoopProf: Dynamic Techniques for Loop Detection and Profiling*, T Moseley, et al.

Loop detection (2)

- Handling unrolled loops



BBLs accessing the same buffer with similar patterns



BasicBlock B

BasicBlock C

BasicBlock D

BasicBlock E

Unrolled

Loop and call-path

```
void crypto_loop(const char *key, void *in, void *out,
                  int len);

void encrypt() {
    crypto_loop("key", dec, enc, len);
}

void decrypt() {
    crypto_loop("key", enc, dec, len);
}
```

Approach review

Goal: find the decrypted stream!

- Loop detection
- Buffer detection
- Data-paths
- Statistical analysis
- Dumping & rebuilding

Buffer detection

- Reason about buffers based on access patterns
  - Consecutive bytes
  - Inconsecutive blocks

0x1000	Original buffer
0x1004	Original buffer
0x1008	Original buffer
0x100c	Original buffer
0x1010	Original buffer
0x1014	Original buffer
0x1018	

0x1000	Composite buffer
0x1004	
0x1008	Composite buffer
0x100c	
0x1010	Original buffer
0x1014	Original buffer
0x1018	

0x1000	Composite buffer
0x1004	
0x1008	
0x100c	
0x1010	
0x1014	Composite buffer
0x1018	

Approach review

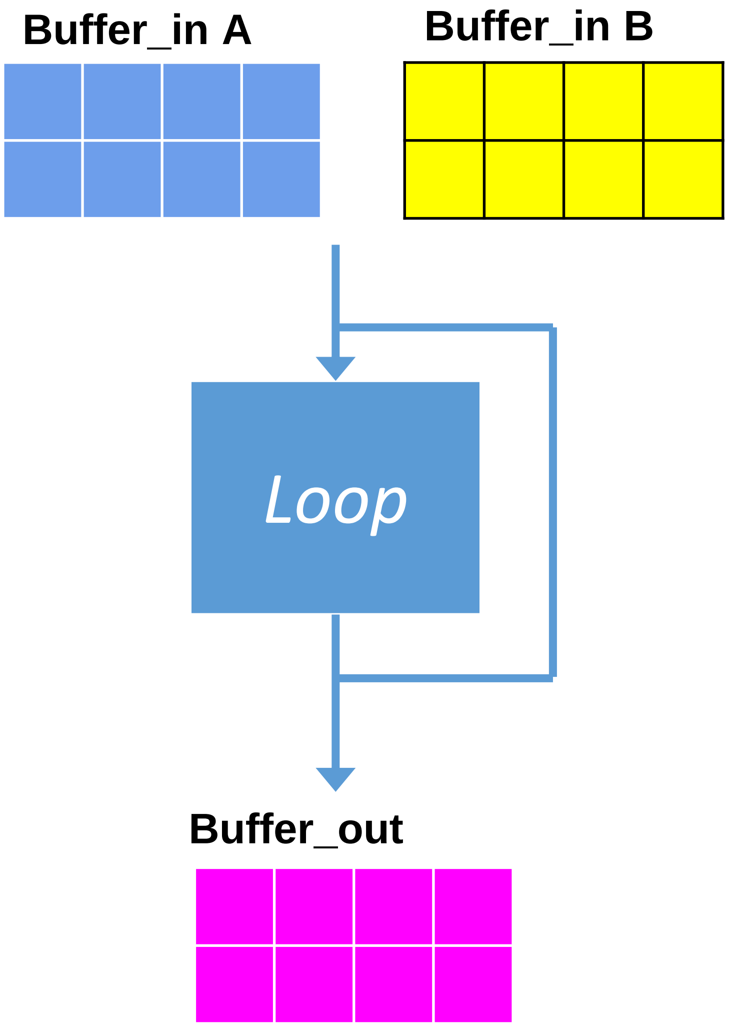
Goal: find the decrypted stream!

- Loop detection
- Buffer detection
- Data-paths
- Statistical analysis
- Dumping & rebuilding

Data-paths

Identify *paths* through a loop which modify the input data to output data

A sensible data-path, find it!



Approach review

Goal: find the decrypted stream!

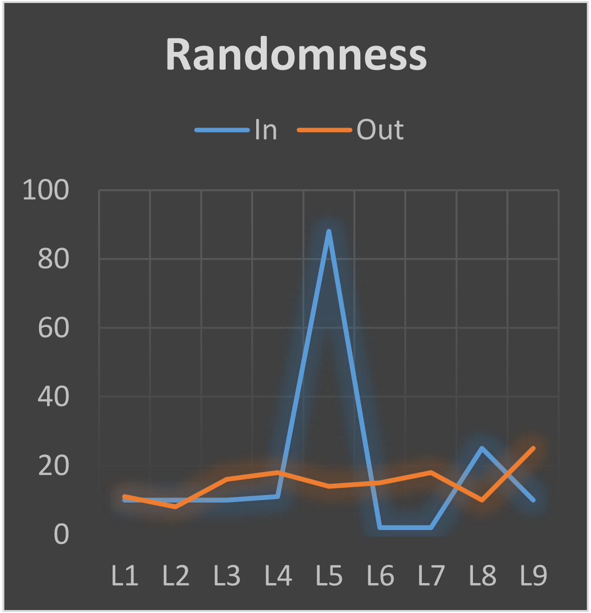
Loop detection

Buffer detection

Data-paths

Statistical analysis

Dumping & rebuilding



Statistical analysis (1)

- Cipher-text indistinguishability
  - Basic requirement for secure cryptosystems

- Entropy should be pretty high, as data is from Internet

Statistical analysis (2)

Stage	Input		Output	
	Entropy	Randomness	Entropy	Randomness
Download	▲	▲	▲	▲
Decrypt	▲	▲	▲	⬇
Decode	▲	⬇	⬇	⬇

Approach review

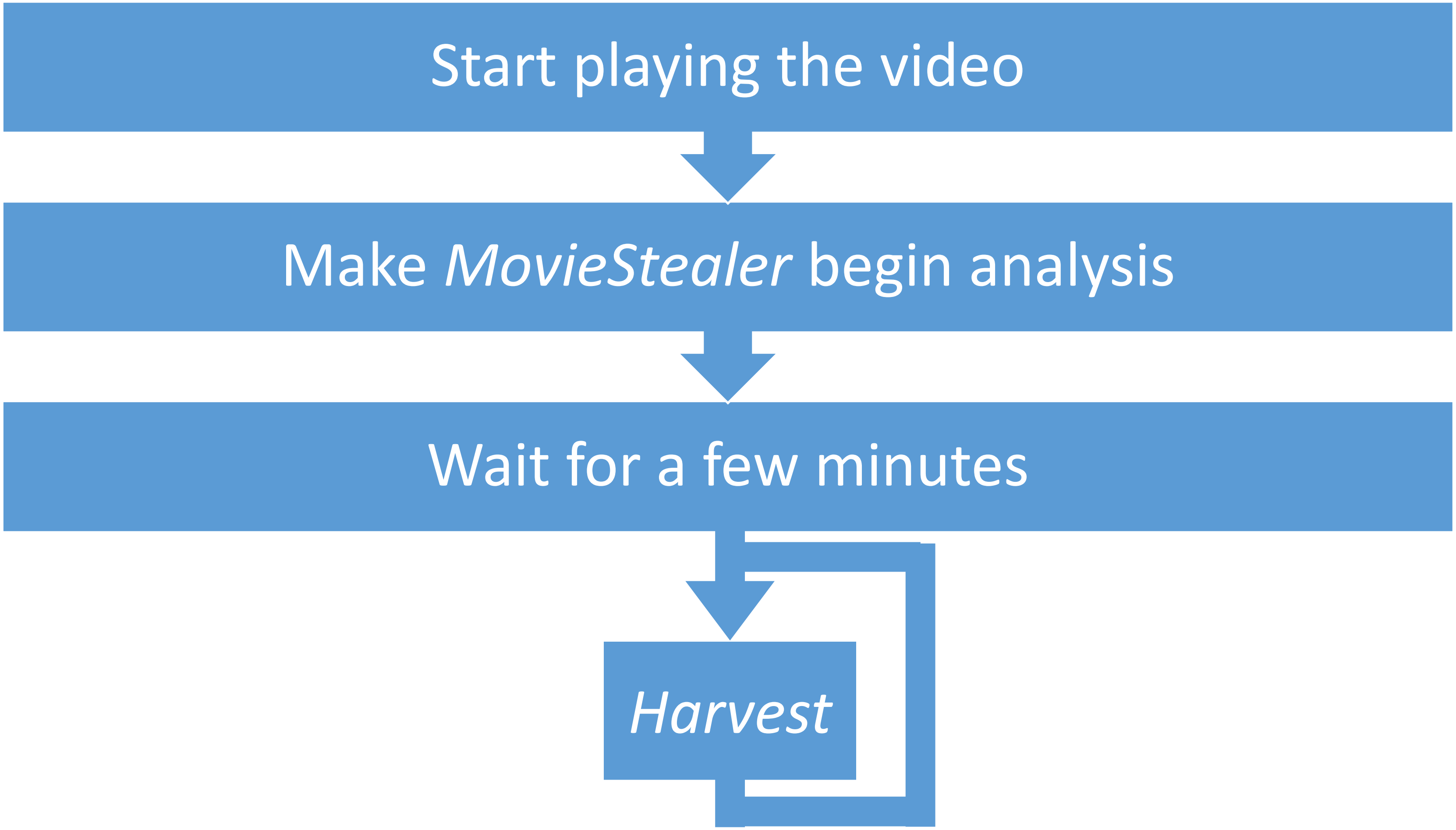
Goal: find the decrypted stream!

- Loop detection
- Buffer detection
- Data-paths
- Statistical analysis
- Dumping & rebuilding

Dumping and reconstruction

* Intentionally omitted

Workflow



Start playing the video

Make *MovieStealer* begin analysis

Wait for a few minutes

Harvest

MovieStealer Optimizations

Problem of the basic approach

- Too much overhead!

- Won’t sniff enough data
- Media players don’t function normally
- Some media players check the performance
- Might get caught by checking systems of DRM

Optimizations

Goal: minimize overheads!

- Improved loop selection
- Efficient loop analysis
- On-Demand instrumentation
- Execution frequency
- Instruction analysis
- Bandwidth filtering
- Copying optimizations
- **Callstack key**

```
on_enter
callstack_key ^= func_addr

on_exit
callstack_key ^= func_addr
```

Callstack key

```
void crypto_loop(const char *key, void *in, void *out,
                  int len);
void encrypt() {
    crypto_loop("key", dec, enc, len);
}
void decrypt() {
    crypto_loop("key", enc, dec, len);
}
```

```
on_enter
callstack_key ^= func_addr
on_exit
callstack_key ^= func_addr
```

Experimental Results

Implementation

- Dynamic binary instrumentation (DBI)
  - Intel PIN framework
- Under Windows 7 32-bit
- Testing
  - A common workstation

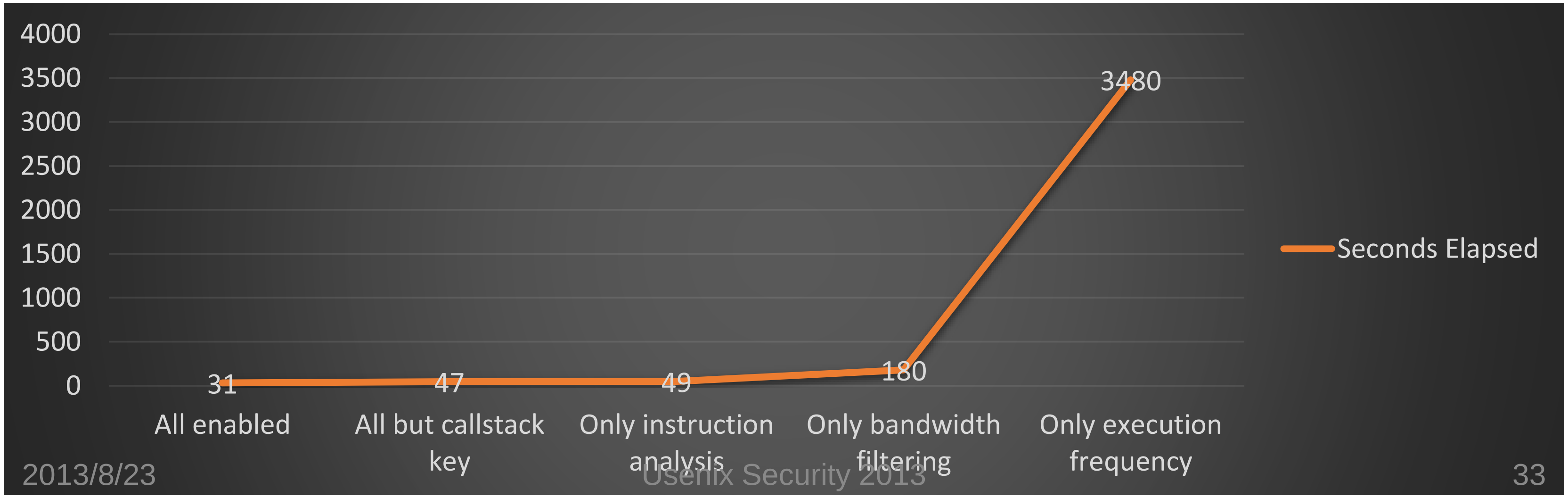
Evaluation

- *GnuPG* for testing optimizations

- Three DRM platforms
  - *Microsoft PlayReady (Netflix)*
  - *Adobe RTMPE (Hulu and Amazon Video)*
  - *Spotify*'s music protection

Results - GPG

Optimizations	Loops Instrumented	Seconds Elapsed
Only execution frequency	40	3480 (112x)
Only bandwidth filtering	35	180 (5.8x)
Only instruction analysis	10	49 (1.58x)
All but callstack key	6	47 (1.51x)
All enabled	7	31



Results - DRM

Platform	Protection	Loops Instrumented	Loops Traced	Buffers Identified	Seconds Elapsed*
Netflix	*Dynamic code*	2274	58	80	110
Hulu	-	1529	46	14	281
Amazon Video	-	1258	35	6	146
Spotify	*Packing and self-checking*	2305	224	60	536

* seconds elapsed before MovieStealer breaks the DRM protection

Countermeasures

Countermeasures

- Attack the instrumentation
  - Anti-debugging
- Attack the loop detection
  - VM’ing those loops
- Attack the buffer detection
  - Non-consecutive buffer layouts
- Anti-piracy
  - Watermarking

Ethics and Legality

I care about Copy right, i didn't know why people here are happy to buy a CD,DVD or somethin that not original. i mean why didn't they care about the people who has make that think. and put the owner of right nothin but just "yeah you have the right, so? ". argghh.... i hope that people that don't respect the author can change their mind to buy the original. if piracy has grown bigger than now, i don't know how this world can be.

Windows 7
Build 7600
This copy of Windows is not genuine

Ethics

- Responsible disclosure
  - Contacted Microsoft, Spotify, Adobe, Amazon, Netflix, and Hulu
  - Microsoft, Spotify, and Adobe responded
    - Tested MovieStealer
    - Confirmed DRM bypass
    - Provided comments and encouraged publication
- Some details omitted (*e.g. reconstruction*)
- No tool/source release

Legality

- We believe this work to be legal under DMCA
  - Consulted with UC counsel and the EFF
  - Thank you all

Acknowledgement

- Thanks for support from Microsoft, Adobe, and Spotify
- Thanks Kevin Borgolte, Yanick Fratantonio, Christian Kreibich, and Thorsten Holz for presentation advice

Contact info

Send us an email

fish@cs.ucsb.edu, yans@cs.ucsb.edu, chris@cs.ucsb.edu, vigna@cs.ucsb.edu

Question time

Questions?

I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.
I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.
I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.
I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.
I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.
I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.
I WILL NOT ILLEGALLY DOWNLOAD THIS MOVIE.