SandboX86: Specification-based Monitoring for Windows

Louis Kruger and Hao Wang
(lpkruger,hbwang@cs.wisc.edu)

WiSA
University of Wisconsin – Madison

The Problem

- It's easy to download programs from the Internet today without knowing what they really do.
- Many programs do more than their advertised features:
  - Observe/collect user activities
  - Transmit private information to home server
  - Download/install other programs (Trojans)
  - Change system/application settings
  - And much more…

KaZaa—An Example

- KaZaa is one of the most popular software today
  - It's free; downloadable in minutes
  - Allow people to share/exchange files
  - Millions of users
    - 247 Million downloads as of July 6th, 2003

(source: CNet.com)

KaZaa (cont'd)

- BUT there is a catch…
  - When installed, you get more than just KaZaa; you also get:
    - Cydoor—a tracking/advertising software
      - displays popup ads to user
      - tracks web surfing habits
    - Gator—ad-driven backdoor software
    - Altnet—a hidden P2P system software
    - And many other potential Malware KaZaa wishes to include

Problems

- These software often run silently in the background, without users' knowledge!
- It is very hard to detect these non-destructive but intrusive activities
- The undesirable features are often closely integrated with desirable features
  - It is hard to remove the undesirable features
  - And… they intentionally make it even harder

Outline

- Motivation
- Problem definition
- Dynamic sandboxing
- Demo
- Future work

Problem Definition

- How to specify program's behavior
- How to detect and enforce program's behavior

Dynamic Sandboxing Using Specifications

- How to specify program's behavior
  - Specification generation
- How to detect and enforce program's behavior
  - Program monitoring
  - Policy enforcement

Milestones

- Prototype ready for Windows platform
  - EDL and ESL compilers
  - Windows Platform Interceptor and Monitor
  - Successfully tested on several applications

Outline

- Motivation
- Problem definition
- Dynamic sandboxing
- Demo
- Future work

Specification Generation

- Goal—allow security analyst to specify acceptable/unacceptable program behaviors
  - For example:
    - Program A cannot connect to host B
    - Program C can only access files in its home directory
    - Program D cannot have any network connection

Specification Generation

- What defines a program's behavior?
  - Using operating system services.
    - (everything else is just computation)
  - Why?
    - OS services are the gateways to the system and beyond.

Specification Generation

Platform Independent

- Two languages
  - Event Description Language (EDL)
    - Set of observable events between programs and the OS
    - Used to model these events
  - Event Specification Language (ESL)
    - Set of sequence of EDL events
    - Used to specify allowable sequences of events

Monitor and Enforce Program Behavior

- Hook System APIs at low level
- Capture and log events in transit
- Policy:
  - allow (pass through)
  - deny (simulate event failure)
  - silent deny (fool the application into thinking event succeeded)

Example EDL and ESL

EDL

```
typedef sockaddr_edl struct { string host, int
      address }
typedef socket_edl int

event connect(socket_edl sock, sockaddr_edl
      addr) = int
event gethostbyname(string host) =
      sockaddr_edl
```

ESL

```
edl "c:\malware\kazaa.edl"

map ipaddr
string blocked =
"cydoor|doubleclick|adserver|fastclick.com|
      brilliantdigital|…"
match gethostbyname(blocked) -> allow <<
  addHash(ipaddr, ret, name)
>>
match connect(sockaddr, socket, <<
      isHashed(addr) >> ) -> deny
match connect(sockaddr, socket, <<
      regex(blocked,
   gethostbyaddr(sockaddr_edl) >> ) -> deny
```

Example EDL and ESL

EDL

event connect(socket_edl sock, sockaddr_edl addr) = int

Characterizes a connection attempt to the host identified by "addr" using socket "sock"

```
typedef sockaddr_e
      address }
typedef socket_edl int

event connect(socket_edl sock, sockaddr_edl
      addr) = int
event gethostbyname(string host) =
      sockaddr_edl
```

```
string blocked =
"cydoor|doubleclick|adserver|fastclick.com|
      brilliantdigital|…"
match gethostbyname(blocked) -> allow <<
  addHash(ipaddr, ret, name)
>>
match connect(sockaddr, socket, <<
      isHashed(addr) >> ) -> deny
match connect(sockaddr, socket, <<
      regex(blocked,
   gethostbyaddr(sockaddr_edl) >> ) -> deny
```

Example EDL and ESL

string blocked =
"cydoor|doubleclick|adserver|fastclick.com|...

Specifies a list of hosts as regular expression to be blocked.

addr) = int

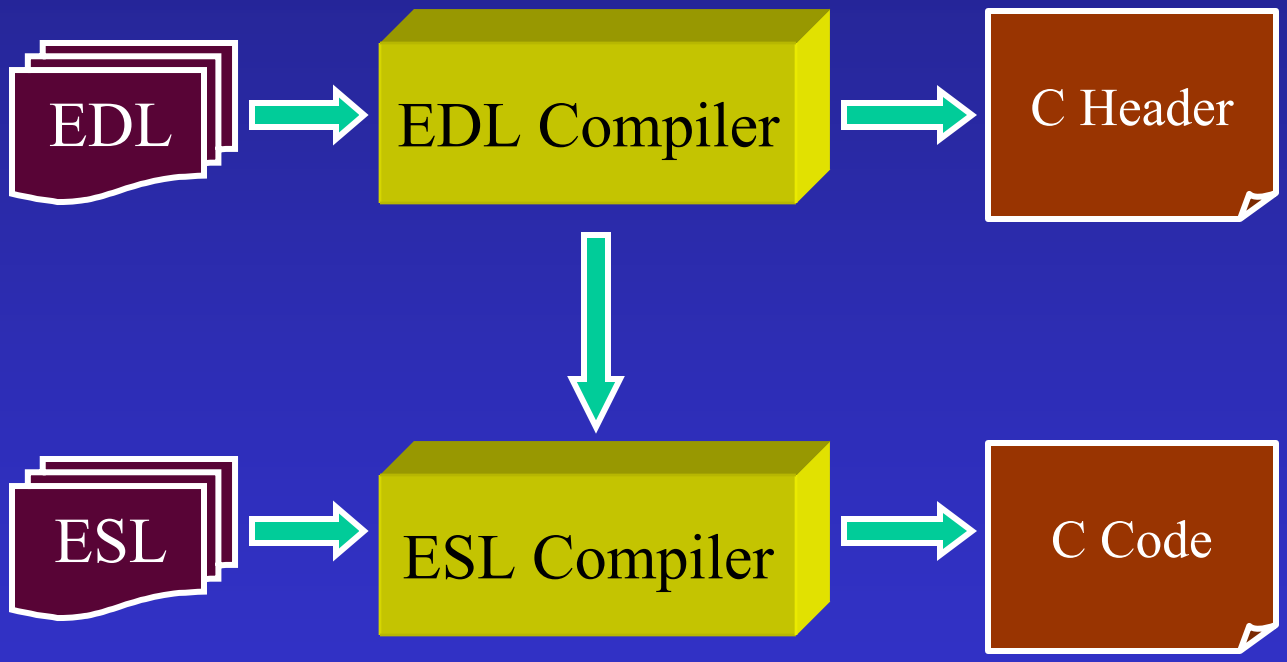
match connect(sockaddr, socket, <<
regex(blocked,
gethostbyaddr(sockaddr_edl) >>) -> deny

Disallow connection to IP address based on reverse lookup.

ESL

```
edl "c:\malware\kazaa.edl"

map ipaddr
string blocked =
"cydoor|doubleclick|adserver|fastclick.com|
  brilliantdigital|..."
match gethostbyname(blocked) -> allow <<
  addHash(ipaddr, ret, name)
>>
match connect(sockaddr, socket, <<
      isHashed(addr) >> ) -> deny
match connect(sockaddr, socket, <<
      regex(blocked,
  gethostbyaddr(sockaddr_edl) >> ) -> deny
```

EDL & ESL



EDL & ESL

Raw event vocabulary

event connect(socket_edl sock, sockaddr_edl addr) = int

Specified by security analyst (for now)

match connect(sockaddr, socket, << regex(blocked, gethostbyaddr(sockaddr_edl) >>) -> deny

EDL & ESL

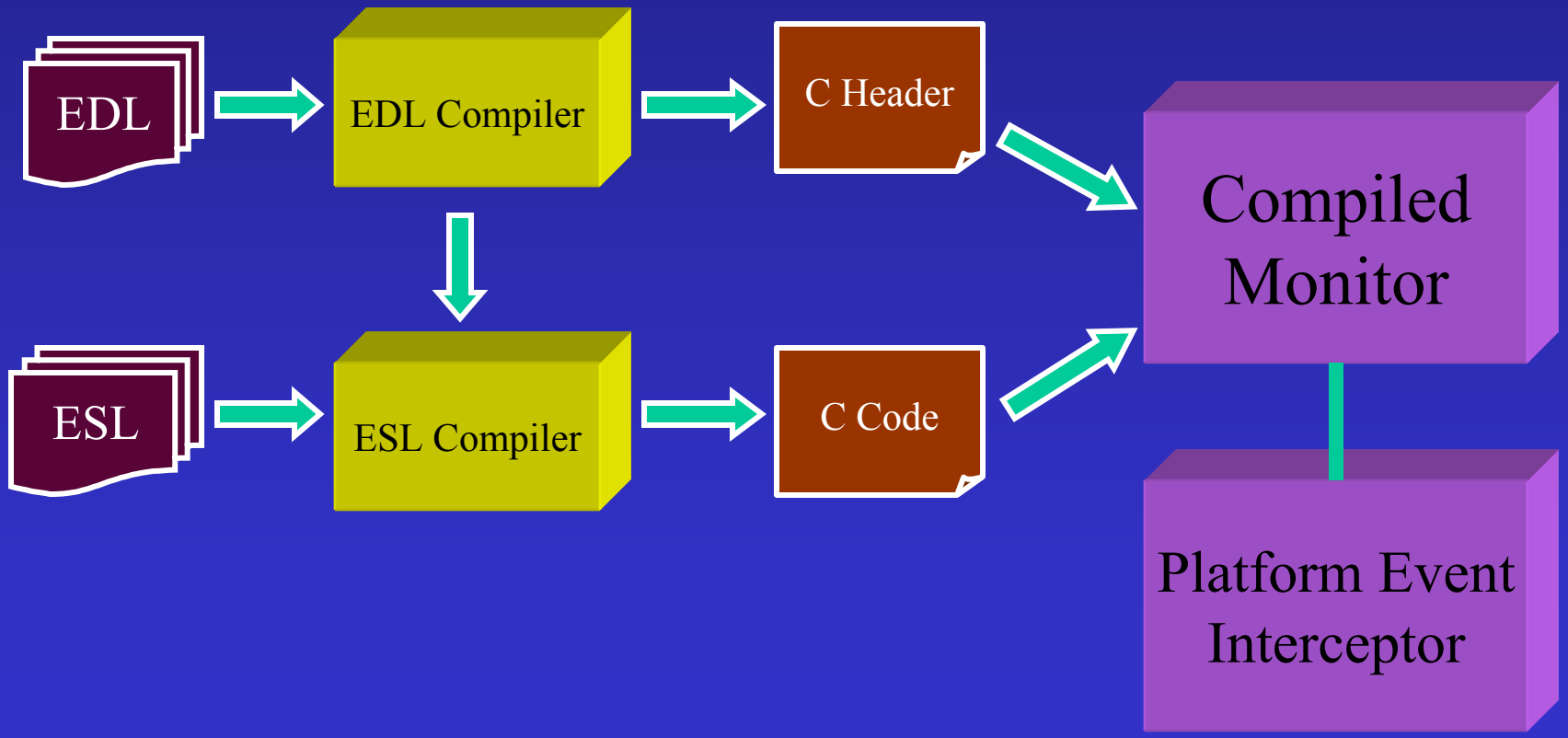
```
int STDCALL eventcall_connect(int sock, struct
edl_sockaddr addr);

void STDCALL eventret_connect(int retval);
```

C Header

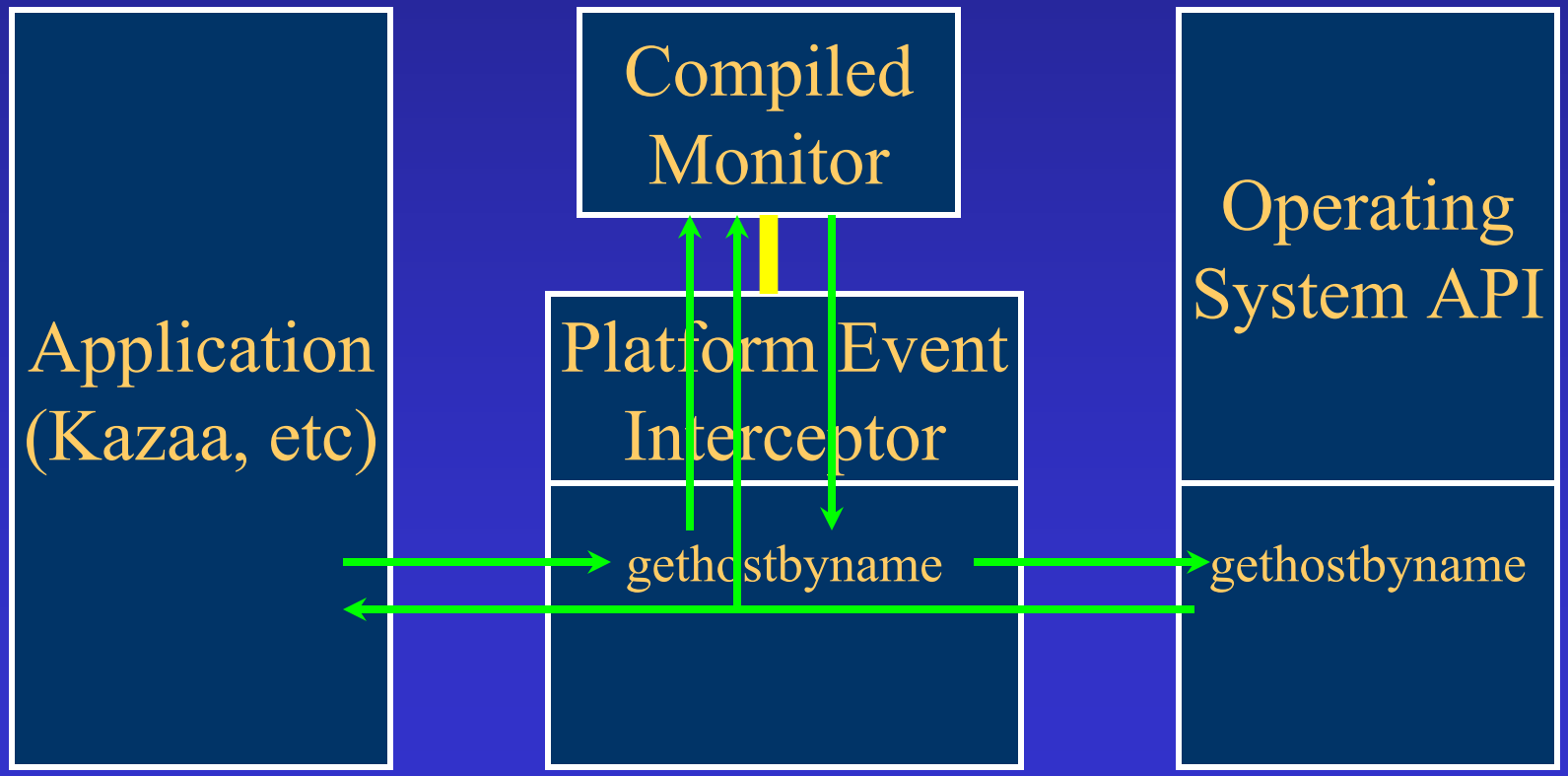
```
int STDCALL eventcall_connect(int sock, struct
edl_sockaddr addr) {
  if (regex(blocked, gethostbyaddr(addr.val.u1)) {
    return EDL_DENY;
  }
  return EDL_ACCEPT;
}
```

C Code

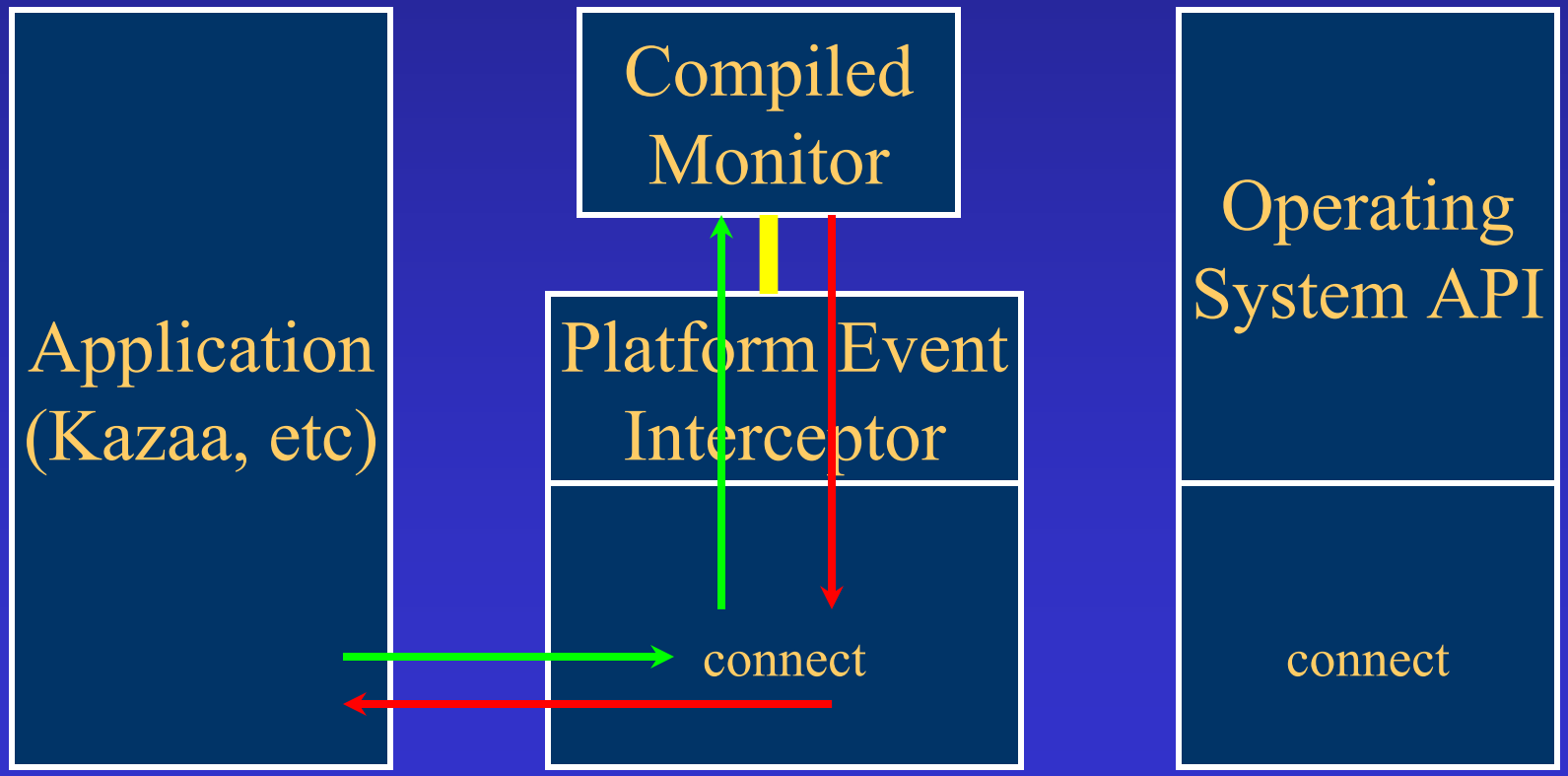
EDL & ESL


EDL
EDL Compiler
C Header
ESL
ESL Compiler
C Code
Compiled Monitor
Platform Event Interceptor

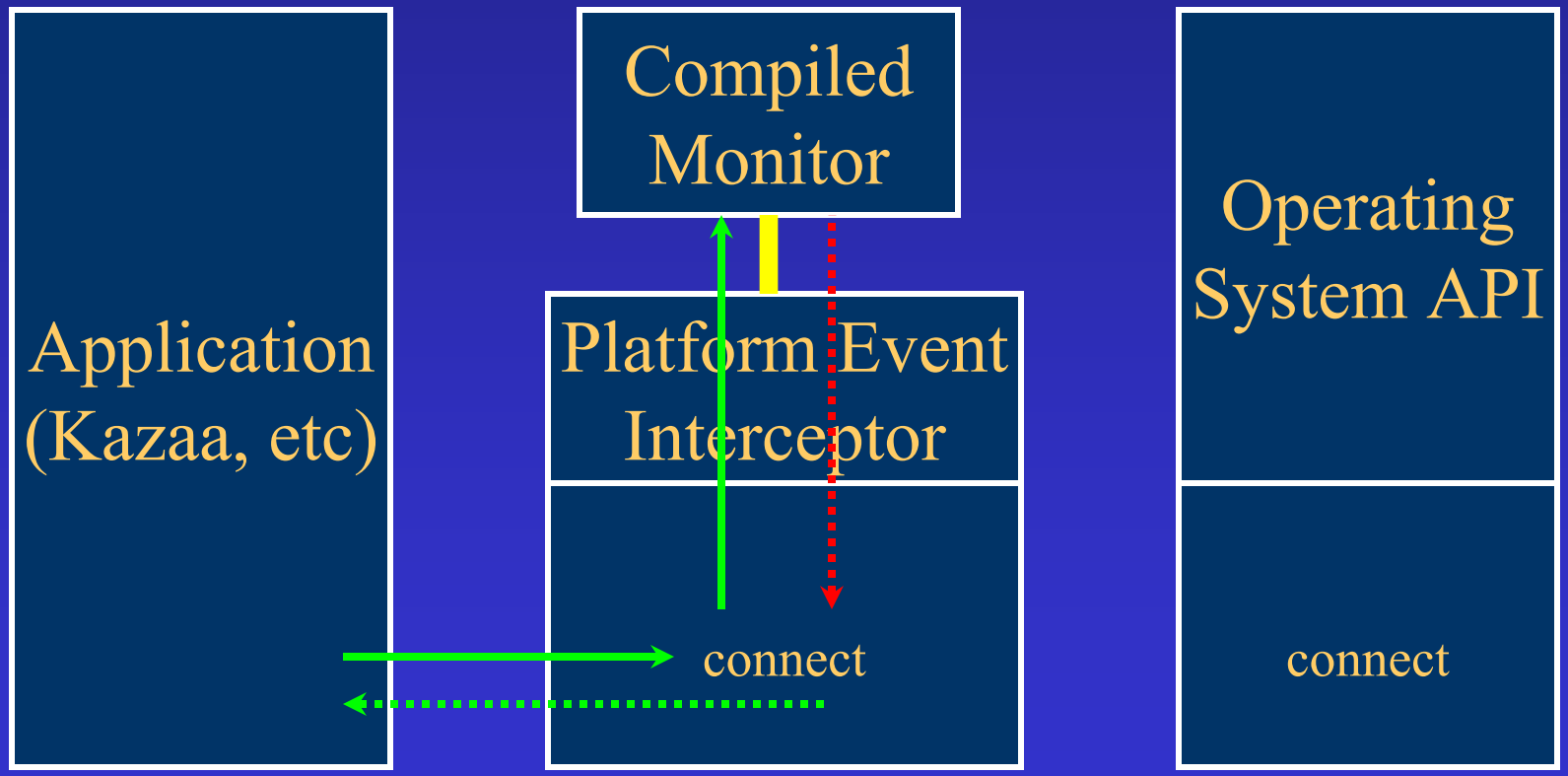
Policy: Accept



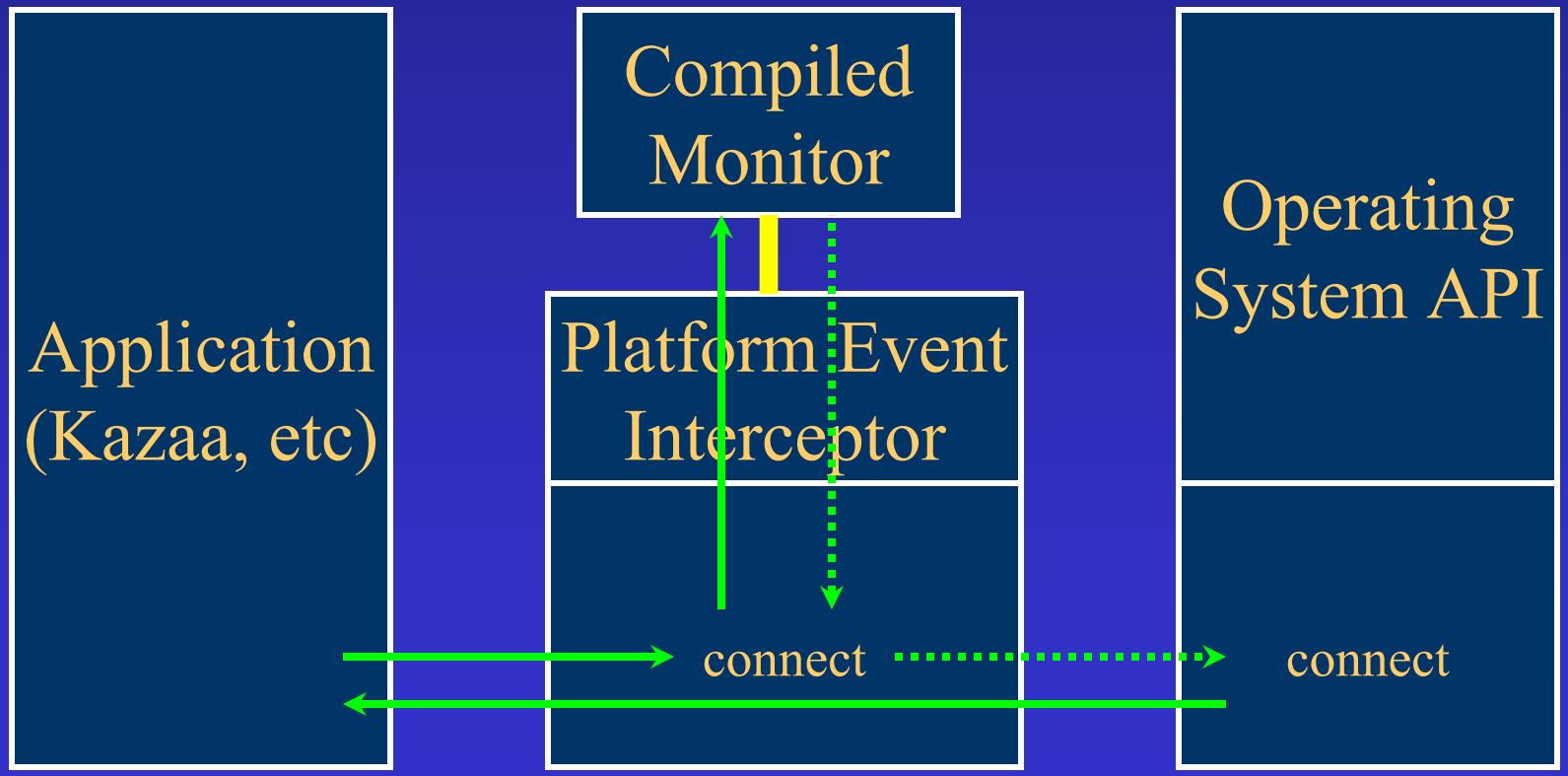
Policy: Deny


Application
(Kazaa, etc)
Compiled
Monitor
Platform Event
Interceptor
connect
Operating
System API
connect

Policy: Silent Deny 1



Policy: Silent Deny 2



Outline

- Motivation
- Problem definition
- Dynamic sandboxing
- Demo
- Future work

Implementation Status

- EDL, ESL compiler: ~1,700 lines of Java code
- Windows Platform Interceptor: ~1,400 lines of C code
- ESL and EDL specifications for KaZaa:
  - 5 specifications for the compilers
  - Generate ~120 lines of C code

Future Work (1)

- Fully implement Kernel Monitor
- Better silent deny
- Refine and extend EDL and ESL
- Port to other platforms
  - EDL and ESL are platform independent

Future Work (2)

- Automatic trace analysis
  - Assist users in detecting malicious event sequences
  - Automatically generate ESL policies
  - Use dynamic slicing techniques

References

- An ever growing Malware List
  - http://www.cexx.org
  - http://www.spychecker.com/home.html
- Other Anti-Malware tools
  - Ad-Aware, Spybot
    - Uses anti-virus signature approach
    - Can not adapt to new Malware