XFOCUS TEAM
From internet,for internet

Embeddable AntiVirus engine with high granularity

---

our understanding and dream

seak

Seak@antiy.net

X'CON 2004

Syllabus

- Challenges to AV Principles
- High granularity processing
- Embeddable AV Engine

Challenges to AV Principles

AV Principles

- AV is not simply a technological battlefront. The overall AV system takes in many logical and legal factors. There are also project planning factors which have some basic principles in common.
- These common principles can be objectively summarized from the AV practice, and then applied to guide the design of an AV engine and tools.
- In 1995 we summarized the basic common principles in 44 items, informally named AV dialectics.

Some Important Items

- A computer virus is a kind of program in the final analysis
- The features of a computer virus are the only identifiers to classify the virus
- The crucial criterion in judging a program to be a virus should be its features or some characteristics of the content
- The only reason that feature code should be purged is if it is objectively or subjectively harmful
- Whether a certain program should be classified as a virus or not should be based on clear criteria

- **The clean up of a virus is the reverse of its infection**
- **User's rights to the AV software:**

  Right to decide: Users can customize the functionality of the AV software instead of using the default configuration

  Right to know: Users should know what the AV software has done in the system

  Right to backup: Users should be provided with means to backup infected files
- **Software should detect viruses inside packages and clean viruses without deleting the package if authorized**
- **Precaution principle: Virus monitoring should prevent the infected files from running and taking control of the system**

AV Dialectics

- With the development of both the application environment and virus techniques, many of our above stated points began to contradict each other
- The fundamental reason for these contradictions is the complication of information systems

Challenge 1: Puzzling Criterion

- Item: The crucial criterion of a computer virus should be the feature code or some characteristics of the content
- Exception: CMD backdoor left by Code Red
- Question: Traditional AV technologies deal with “Yes or No” problems, where the only criterion is the content of the program. But under some circumstances, the boundary between harmful and harmless becomes vague.

Challenge 2：Paradox of Range

- Item: Whether a certain program should be detected or not should be based on clear criteria
- Exception: psexec tool used in Worm.Dvldr .
- Question: The emergence of unwanted files is another puzzle in detection criterion. How far should AV software reach? What is the criterion? So far, many AV products include adware detection, is this reasonable or legal?

Challenge 3：Package Enigma

- Item: Detect viruses inside packages and clean viruses without deleting the package if authorized
- Exception：DIY worms (such as password worms), and worms using or saving in zip formats (such as some variation of netsky)
- Question: The basic assumption of traditional AV software is that a package file is normal file that may contain a virus. DIY worms are self-extracting packages. Some worms make many zipped backup copies on the disk which cannot be removed by AV software.

Challenge 4: Junk Files

- Item: The only reason that feature code should be purged is if it is objectively or subjectively harmful
- Exception: Crisis caused by unofficial evaluation
- Problem: If one company detects some trivial files, other companies will follow suit in order to win higher marks in competitive evaluation. Is this worthwhile behavior? How can it be balanced with efficient and high-throughput virus detection?

Challenge 5: Responsibility problem

- Item: The clean up of a virus is the reverse of its infection
- Case: Leftover backdoors leading to a worm returning
- Question: Is AV software responsible for recovering all the system modifications made by the virus? And how to deal with leaks? Is this work endless?

Challenge 6：The time of action

- Item: Virus monitoring should prevent infected files from running and taking control of the system
- Case: Arguments on file evaluation
- Question: Since it is difficult to detect unknown PE viruses, Trojans or backdoors, should the heuristic report based on behavior be acted on immediately?

Challenge 7：Active Protection

- Item: User's rights to the AV software
- Case: scanning worms changed the image of victims
- Question: At first, viruses aimed simply to infect users' systems. Now, more often than not, they try to make infected systems further infect other systems. In such a case, can a virus be removed without the user's permission? What means are acceptable? Is this a technological question or legal question?

Putting forward and solving the problems

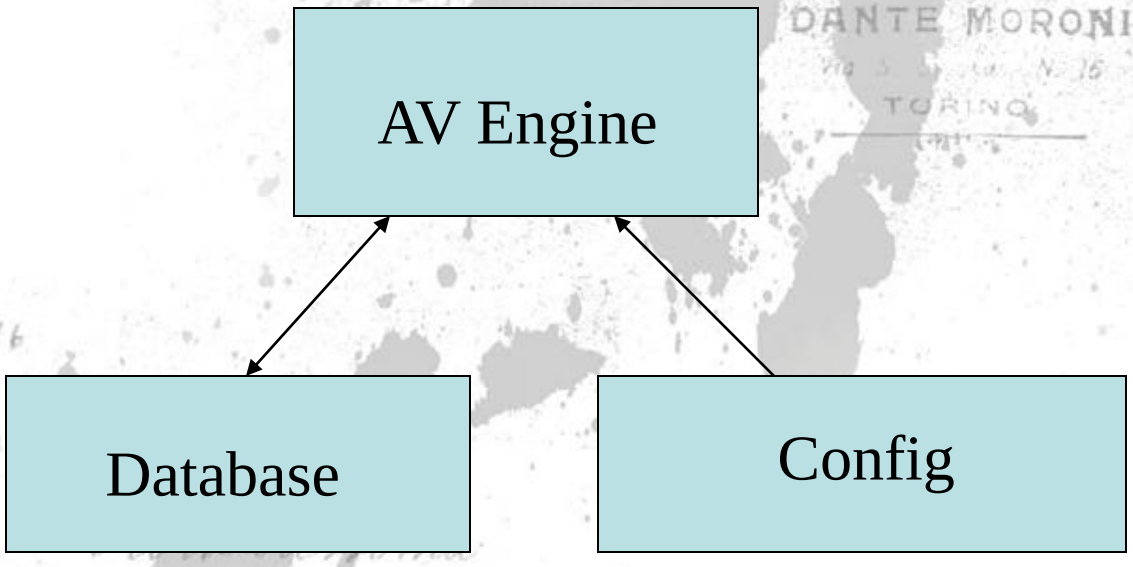
- None of these problems are too difficult to be solved technologically
- Some of them concern style and morals, however “Puzzling Criterion”, “Package Enigma”, and the “Responsibility Problem” are reactions to the traditional system and framework of the AV engine.
- We need more adaptive and reasonable engine frameworks instead of expediency in programming

XFOCUS TEAM
From internet,for internet

High Granularity Processing

X'CON 2004

The three elements of an AV Engine



The three elements of an AV Engine

The three elements of an AV engine are the engine, database, and configuration. The engine relies on the database to detect, and the definitions in the configuration to work.

Before, we put much emphasis on the engine. Now, we need to pay more attention to the configuration to see what gains it has to offer us.

We also need to reevaluate the database – the maintenance of which is traditionally mechanical – to see whether the potential for creativity still exists.

The Traditional Database

	Type 1	Type 2	Type 3	Type 4
Number	√	√	√	√
Mod num	√	√	√	√
Virus name	√	√	√	√
First word of Feature code			√	√
Offset1+Sign 1			√	√
Offset2+Sign 2			√	√
File type flag				√
Process arg	√		√	√
Processing module name			√	√

The Traditional Database

- In working with a database, 95% of viruses are detected via records of type 3 and type 4 (featuring code detection). Detecting the remaining 5% of special viruses is done with records of type 1 and type 2 (independent module detection).
- Over 80% of viruses are processed via argument, and the remaining 20% via processing module。

Basic characteristics of the traditional Database

- Object Control: what to detect
- Behavior Control: how to process
- Effectiveness Control: intensity of detection

Traditional Configuration Means

- Flow control (Program)
- Debug Switch (Developer)
- INI control (User)

Object Control

- Memory=Yes; check the memory
- Sectors=Yes; check the boot sector
- Files=Yes; check file system
- Packed=Yes; check packages
- Archives=Yes; check archives
- MailBases=Yes; check emails
- MailPlain=Yes; check encoded files
- FileMask=2; check the extended names
- UserMask= ?; user defined extension
- Exclude=No; Don't check customized extensions
- ExcludeMask= ; Don't check definition of extensions

Behavior Control

- InfectedAction=0; remove viruses
- InfectedCopy=No; back up viruses
- InfectedFolder=Infected; back up folders
- SuspiciousCopy=No; back up suspicious files
- SuspiciousFolder=Suspicious; back up folders
- Report=Yes; generate logs
- ReportFileName=Report.txt; name of log file

Effectiveness Control

- Warnings=Yes; Show warnings
- CodeAnalyzer=Yes; Open the code analyzer
- RedundantScan=Yes; Redundant scanning

That's enough?

- In the traditional AV environment, this kind of granularity is enough for control, however problems occur when it comes to more complicated environments.

Application Case 1

- Consider what different features the engine will have when working as AV software for a single computer VS working as one module in a mail server?
- I-Worm.Nimda.e is a infective worm. When processed locally, it should be regarded as a PE infected file, but for a mail server, it should simply be discarded.
- Win95.CIH is a infective virus. When detected, whether it is local or on mail server, it should be processed as an infected virus and the original file should be recovered.
- The essential difference is that Win95.CIH doesn't mail itself but rather is an executable program mailed by the user, while Nimda behaves contrarily.
- This situation requires different processes for different kinds of viruses in various environments, it is beyond the capacity of traditional engine control.

Application Case 2

- Network virus detection equipment contains several responding modules
- What policy should these responding modules work with?
- Some mail worms create addressees randomly, what will happen if sending creates a feedback loop?
- Some mail worms use bots to create addressees. What will happen if the worm starts sending duplicates?

- Email detection
- Duplicate email detection
- Feedback email detection
- Reset connection

Requirements from Engine User

	SMTP detection				POP3 detection			
	Faked Recipient	**True Recipient**	**True Sender**	**Faked Sender**	**Faked Recipient**	**True Recipient**	**True Sender**	**False Sender**
Feedback Detection			**effective**	**ineffective**			**effective**	**ineffective**
Duplicate Detection	**ineffective**	**effective**			**effective**	**ineffective**		

Application Case 3

- Integration with networking equipment is an effective response.
- See: OPSEC, TOPSEC

- Different processing for scanning worms and mail worms.
- It is simple to scan worm infection IP nodes. But if we do the same to email worms, they may send the same email over and over again causing DoS
- We should check whether there is a proxy server on the network

That's not enough!

- New demand goes beyond the capacity of the traditional engine
- How can we solve this problem?

XFOCUS TEAM
From internet,for internet

Embeddable AV engine with high granularity

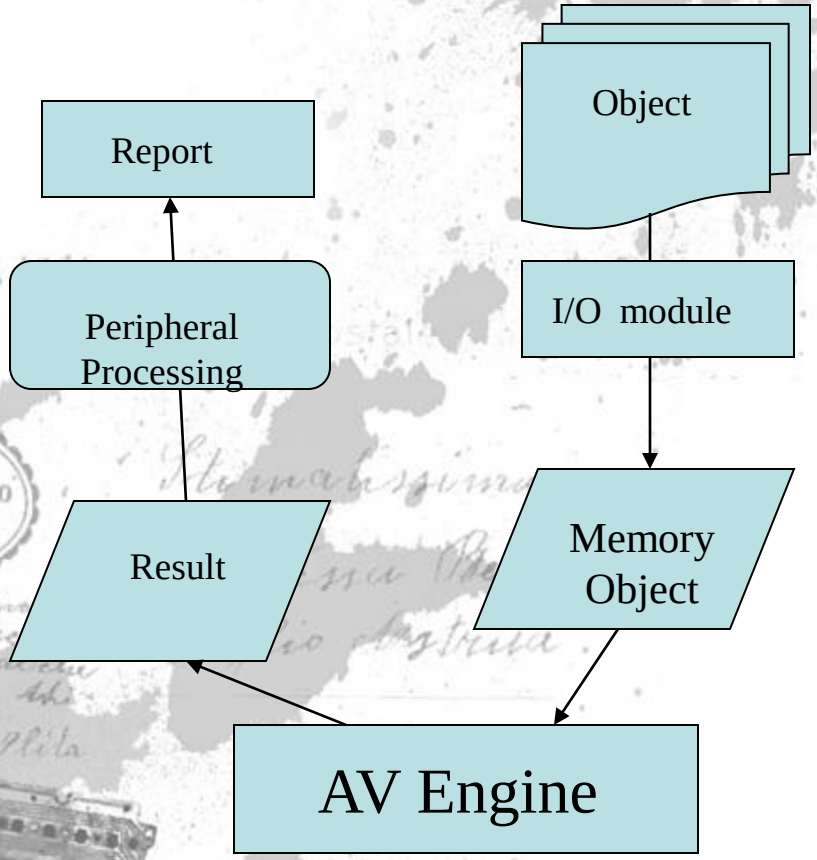
Putting Forward the Question

- The trending movement of network security products implies that virus filtering mechanisms will extend to equipment at different levels
- The above discussion shows the need for the AV engine to adapt to more complicated environments
- Embedded equipment or AV engines in other environments are designed for high granularity

Application of Embedded Engine

Application Form	Details
AV module in Firewall	Construct linear speed virus filtering module for package filtering firewall with a network engine. Construct file stream virus filter for app proxies, transparent proxies or a stream filtering firewall with a file engine.
AV module in router	Add virus filtering ability to routing equipment with high speed package level scanning
AV module in switch	Add virus filtering ability to switching equipment with high speed packet scanning.
Virus detecting plug-ins in IDS	Extend the network engine to provide the IDS with network virus detection ability
AV module in GAP device	Extend GAP equipment with virus filtering ability
Virus protection in mail system	Embed virus detection ability into mail servers
Independent AV software	User need only to program against an API to develop their own AV software

Basic Requirement 1: Memory Engine



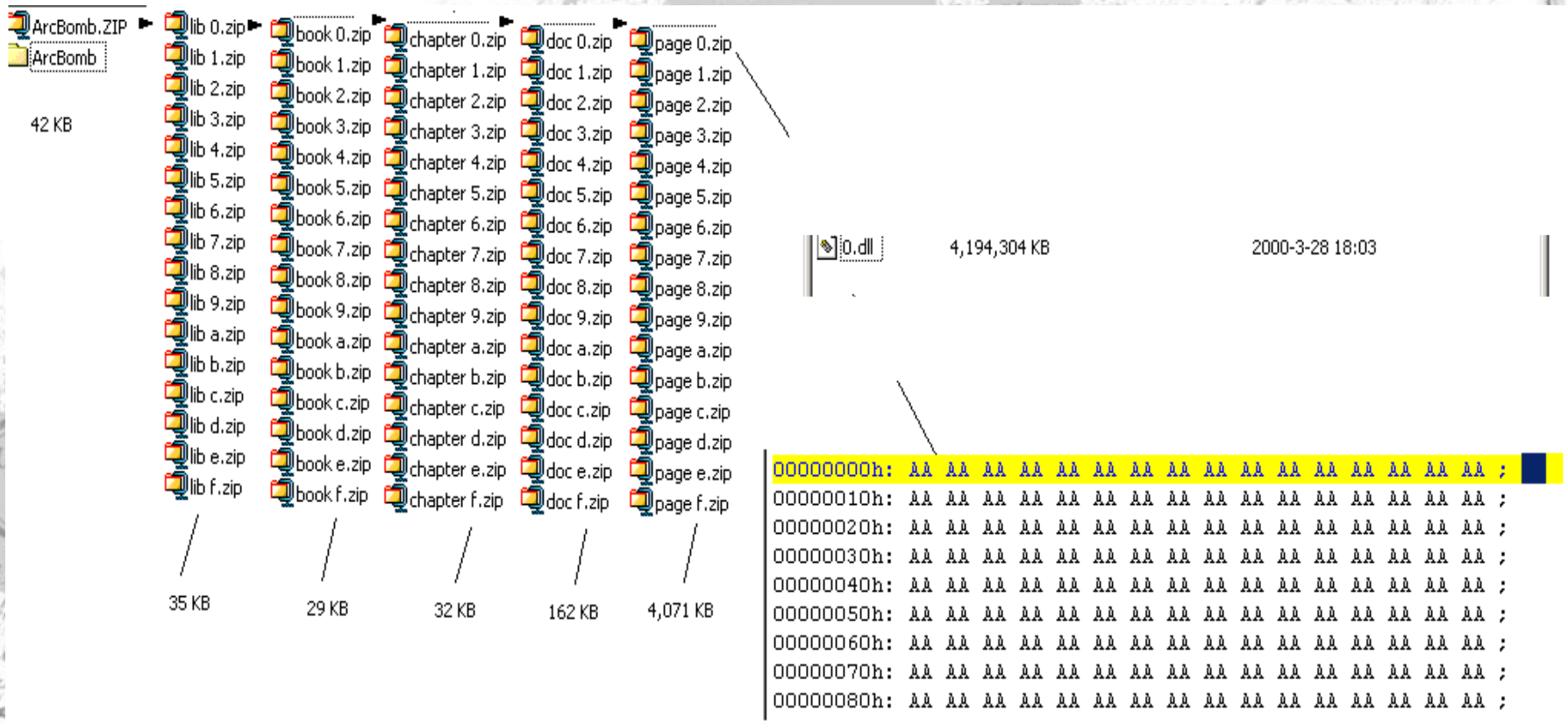
Memory Engine Interface Demo

```
/*scanning parameter structure */
typedef struct _AVLF_SDK_SCAN_PARA
{
char * pBuffer;                 /* pointer  to buffer */
unsigned long ulSize;           /*size of the buffer */
const char * pDescription;      /* description information */
int   bUnpack;                  /* whether to unpack*/
int  bKill;                     /* whether to  kill the virus */
int  bKilled;                   /* whether virus was killed successfully*/
} AVLF_SDK_SCAN_PARA,*PAVLF_SDK_SCAN_PARA;

/* set the receiver */
AVLEACHSDK_API int AVLF_SDK_SetReciver(IReportReciver *pReciver);

/*scanning：return 0 if no virus detected，return 1 if virus found，detailed information is received by
    the receiver class*/

AVLEACHSDK_API int AVLF_SDK_Scan(PAVLF_SDK_SCAN_PARA pParamter);
```

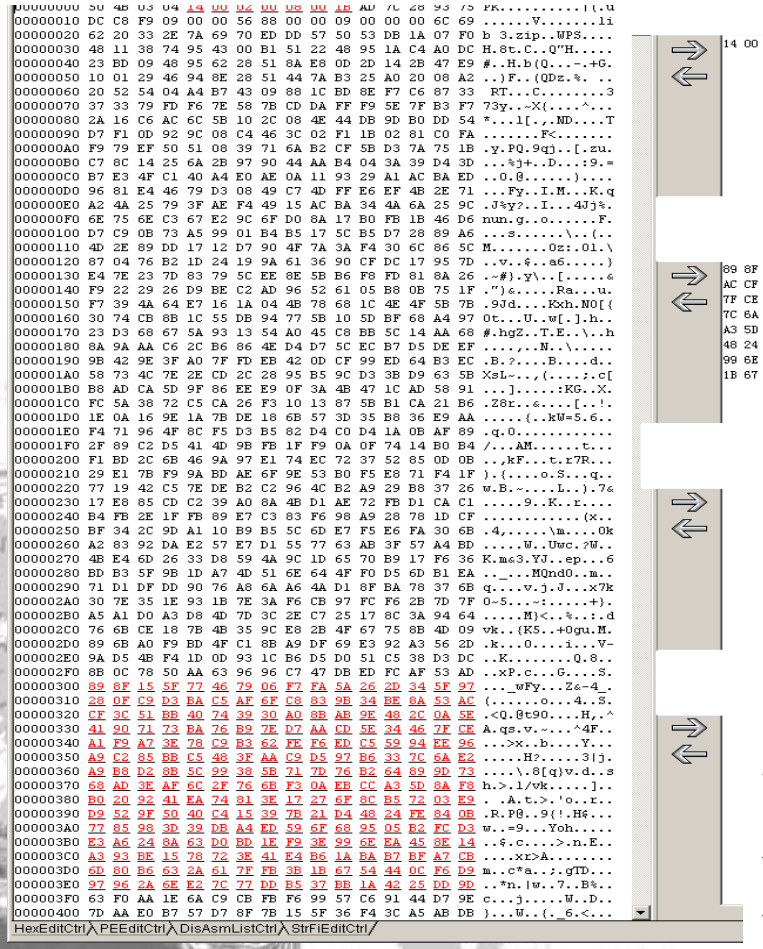
Basic Requirement 2—— Recursive Engine

- Modern AV engines have evolved from branched engines led by module-based format recognition to recursive engines
- In a recursive engine, scanned objects could have multiple flags, which can be detected by corresponding modules
- McAfee's bug in detecting SFX
- archbomb.zip

Archbomb.zip DoS AVware



How to detect archbomb.zip



sign1

Offset: 4h

Length: 7h

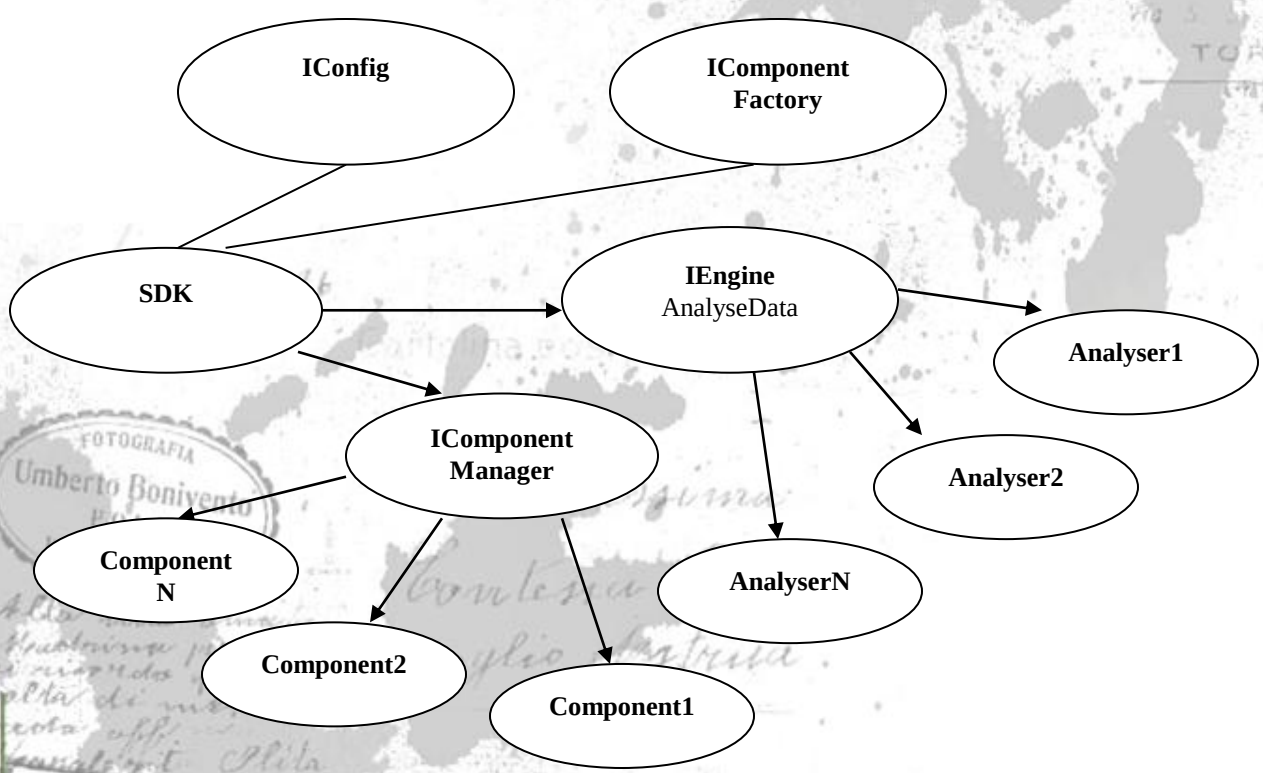
Sign 2

Offset:300h

Length:F0h

A Zip which is also a binary stream could be detected by binary engine instead of what would be done in the traditional branched engine – being passed to archive extracting module by the format recognition module.

Example—— Recursive Engine



1. Analyzers are parallel in structure, none are prerequisites.
2. Results from the analyzers can provide different priority ratings, with viruses listed as the highest and files needing further processing as the lowest
3. In principle, analyzers work serially, with higher priority results being forwarded.

Basic Requirement 3——portable engine

- Working environment could be the x86 architecture, or other architectures like PPC
- Modules written in x86 assembly language are a barrier to porting to other architectures.

Basic Requirement 4——Highly Controllable Engine

- What are the essential requirements for high granularity?
- Virus processing in different environments cannot only rely on detecting the infection feature but also the “specialty” of the virus.
- The granularity of control needs to reach the individual virus, the database needs to provide more information.
- Virus processing will be done with information from the database about the virus specialty.

Control Level

- Flow control (Program)
- Debug Switch (Developer)
- INI control (User)

- Flow control (Program)
- Virus attribute
- Debug Switch (Developer)
- Stencil (Condition)
- INI control (User)

Traditional Database V.S. High Granularity Database

```
struct vxdb
{
char name[255];
char fword[4];
char offset1[4];
char crc1[8];
char offset2[4];
char crc2[8];
…
};
```

```
struct tgvxdb
{
char name[255];
char fword[4];
char offset1[4];
char crc1[8];
char offset2[4];
char crc2[8];
…
int vxattribute ;
};
```

Basic Requirement 5——Precise Processing Engine

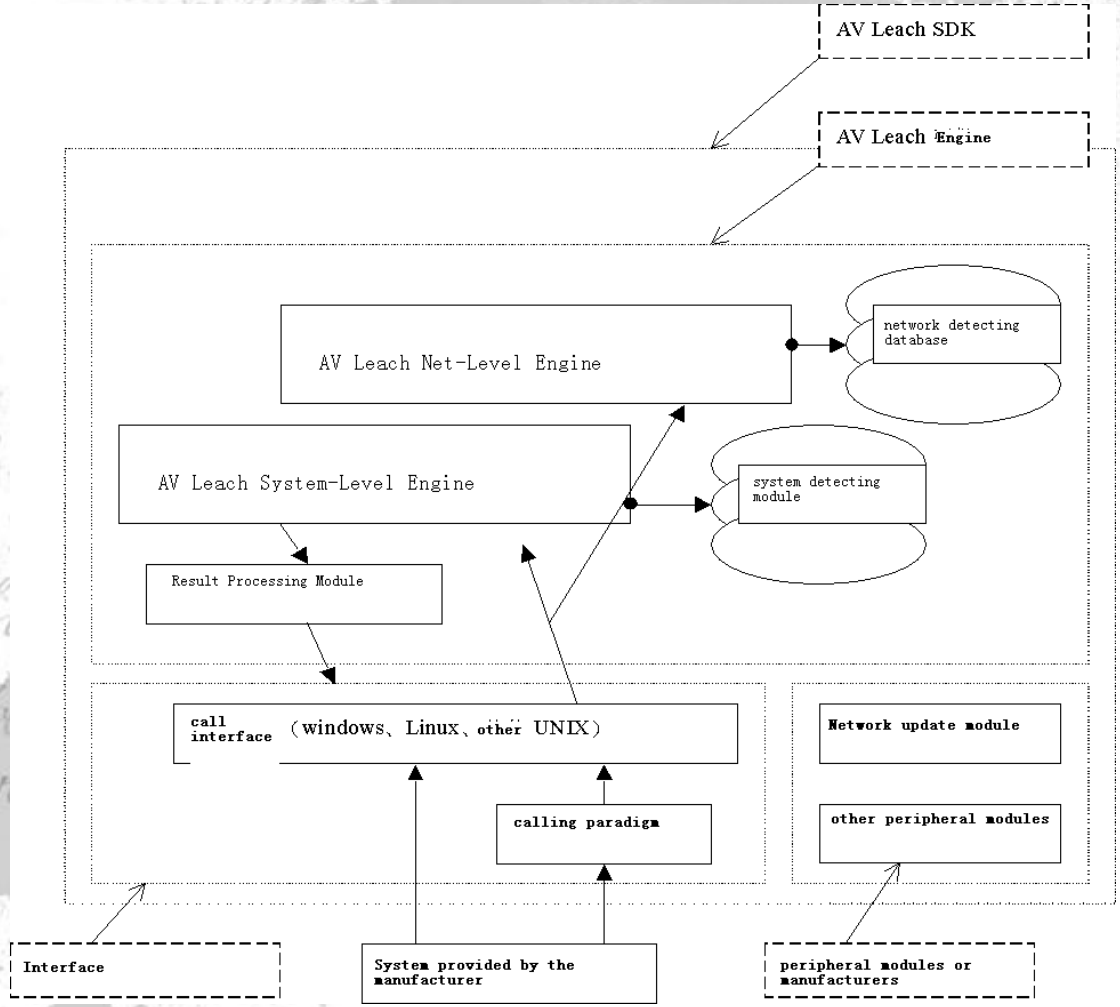
- Perfect reverse engineering is the end goal.
- The High Granularity Engine ends the era in which the AV company does not need to analyze the virus.

How to Process

- Clean com tail
- Clean com head
- Clean exe tail
- Clean ne tail
- Clean pe tail
- Remove file
- Copy data block
- Move data block
- Insert data block
- Modify data block
- Delete data block
- Fill in data block
- Truncate data tail
- Truncate data head

- On the left is the cleaning parameter set which is widely accepted by many companies.
- We need the same detailed processing script for non-infective viruses
- Is this work endless?

Our Model


AV Leach SDK
AV Leach Engine
AV Leach Net-Level Engine
network detecting database
AV Leach System-Level Engine
system detecting module
Result Processing Module
call interface (windows、Linux、other UNIX)
Network update module
calling paradigm
other peripheral modules
Interface
System provided by the manufacturer
peripheral modules or manufacturers

Our Understanding and Dream

- AV principles are not invariable. Instead, they are evolving dynamic principles. They require not only summarizing but also supplementing and replacing.
- We believe in our understanding and we persist in our dream.
- Thank you!