

# Stack Frame Protection with LD\_PRELOAD



@auth: pancake  
@place: FIST  
@date: 20040507

# Outlook

- Buffer overflows and stack basics;
- Protection methods;
- Target on preload;
- LibSFP {aka my testing lib};
- Internal work;
- Few code examples;
- Links and EOF;

# Buffer overflows basics

- The first cause of insecurity;
- Every function is closed into an stack frame.
- The stack frame saves information about local variables and return pointer.
- Programmers must focus in secure code, not just external security.

# Protection methods

- Development stage.
  - Patches to Gcc that uses canary-based methods to ensure the SF integrity.
  - Use lint to clean insecure function calls.
- Runtime
  - Ptrace-based security. 3x slower, but the most secure.
  - Library-based security. Faster and protects almost basic bugs.

# Preload method

- Dynamically load of a library with `LD_PRELOAD` or `ld.so.conf` by `ld.so`;
- Replacement for the most buggy function symbols by secure ones (`strcpy`, `memcpy`, `strlen`, ...);
- Exists some libraries that do that:
  - Libsafe – secure libc functions.
  - Libformat - secure format strings.
- Main problem: non-portable.

# LibSFP

- I decide to write a libformat/libsafe replacement.
- Target on:
  - UNIX-OSes portability (GNU,\*BSD,...)
  - Architecture portability (endian, stack)
  - Open, active development. It's GPLd.
- Actually its development is stopped. But i'll be happy to receive contributions and follow the project.

# Internal work

- Basically it's a library that rewrites every symbol.
- Cross all stack frames layers until find the current one.
- Measures the current SF size and limits calls to this size.
- Library can be configured at runtime
  - Offset: Change overflow margins.
  - Action: alert, ignore, force CoreDump...

# Internal work

- There are 3 kind of variables:
  - **Local** – stored in the stack frame. (easy to protect).
  - **Global** – stored in Heap. (difficult to know the limits).
  - **Malloc** – stored in Heap space with chunk header information. (the assigned space limits could be read from chunk headers).
- Malloc techniques:
  - LibSFP stores a magic value into the chunk header to separate global variables from chunked ones.
  - Chunks are memory-aligned, it means that size isn't exact.

# Internal work

- There are 3 kind of variables:
  - **Local** – stored in the stack frame. (easy to protect).
  - **Global** – stored in Heap. (difficult to know the limits).
  - **Malloc** – stored in Heap space with chunk header information. (the assigned space limits could be read from chunk headers).
- Malloc techniques:
  - LibSFP stores a magic value into the chunk header to separate global variables from chunked ones.
  - Chunks are memory-aligned, it means that size isn't exact.

# Few examples

Now it's the moment for going to the terminal and show some examples...

# Links and EOF

- Libsafe
  - <http://www.research.avayalabs.com/project/libsafe/>
- Immunix Gcc StackGuard
  - <http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/>
- Libsfp isn't released yet, but if I receive interest I would probably upload into:
  - <http://www.nopcode.org/>
  - <http://pancake.host.sk/altres/src/>

EOF

[questions, tips, apologise..]