Snort 3.0

Why do we need it?

What will you do with it?

William Young
william.young@sourcefire.com
Years of Intrusion Monitoring
Static defenses usually work
Staff is well trained but...
Threats are constantly changing
So is the network
“Nine out of Ten breaches involved one of the following

- A system unknown to the organization
- A system storing data that the organization did not know existed on that system
- A system that had unknown network connections or accessibility
- A system that had unknown accounts or privileges

Context changes everything
IPS: Threat Impact is key

- Detectable Threats
- Meaningful Threats
Basic Impact Analysis Works
Need Dynamic Risk Awareness
Need Dynamic Risk Awareness
Need Dynamic Risk Awareness
Need Dynamic Risk Awareness
Need Dynamic Risk Awareness
Network behavior is telling
Real world: Systems have identity
Real world: Systems have identity
“The breakdown is in the process. What these organizations seem to lack is a fully proceduralized regimen for collecting, analyzing, and reporting on anomalous log activity.”
Integrated Security Model
Increase operational efficiency
Enable business value
Still limitations to the model...
What’s Driving Development?

- High-speed inline operation
- Multi-core CPU environments
- Better hooks for hardware acceleration
- Next generation adaptive engine
- Continuous operation capability
- Extending capabilities of Snort more easily
Design

- Efficient
- Snort 3.0 is architectured to be accelerated
- Snort 3.0 is multithreaded
  - Engines can run continuously, reloads unneeded
  - Engines can be parallelized for multi-core CPUs
- Clean (NEW!) code base
- Reduce lines of code count
- Eliminate old code, unused features
Snort Lessons

- Reinvent tuning
- Address evasion
- Improve Snort’s language
- Rethink prioritization
- Take advantage of modern hardware
Snort Pipeline

Config Loader

Dynamically Created Detection Framework

Snort.conf

Data Acquisition

Decode

Preprocess

Detect

Output

Snort 1.5 through 2.8
The Snort 3.0 project has split into two major components

SnortSP - Snort Security Platform

“Platform” for running analysis and control “applications”

Engines

“Applications” that run on SnortSP
**Data Source API**

typedef struct _data_source {
    s_mutex_t               mutex;
    volatile int_t               config:
    volatile int_t               run:
    int_t          *daq_config:
    daq_module_t     *daq:
    flow_manager_t   *flow_mgr:
    defrag_manager_t *defrag_mgr:
    decode_instance_t  *decode_instance:
    dsr_callback_idle_func idlefunc:
    ref_engine_t                *user_context:
    packet_t                *free_packet_list:
    traffic_t               *free_traffic_list:
} i_data_source_t;

int dsrc_init(void);
void dsrc_cleanup(void);

* Show list of data source instances

int dsrc_show_sources();

* Create/delete a data source

int dsrc_new(data_source_config_t *ds_config);
int dsrc_delete(const char *name);
int dsrc_config_daq(data_source_t *src);

* Run or stop a configured instance of a data source

int dsrc_start(data_source_t *src);
int dsrc_stop(data_source_t *name);
int dsrc_run(data_source_t *src);
int dsrc_finish_traffic(data_source_t *data_src, struct _traffic_t, ANALYZER_ACTION action);
int dsrc_finish_flow(flow_t *flow);

Registration methods for user-provided code

int dsrc_register_user_context(data_source_t *src,
    ref_engine_t *context);
int dsrc_register_idle_function(data_source_t *src,
    dsrc_callback_idle_func idlefunc);

* Data source lookup function.  Must call dsrc_release to release the reference when no longer needed.
data_source_t *dsrc_get_dsrsbyname(const char *name);

* Data source release function.  Must be called after dsrc_get_dsrsbyname.
data_source_t *dsrc_release(data_source_t *src):

* Show a data source's configuration

int dsrc_show_config(data_source_t *src);
int dsrc_show_config_byname(const char *name);

* Show stats

int dsrc_show_stats(data_source_t *src);
int dsrc_set_run_state(data_source_t *src, int *run);
int dsrc_get_run_state(data_source_t *src, int *run);
int dsrc_set_inuse_state(data_source_t *src, int *inuse);
int dsrc_get_inuse_state(data_source_t *src, int *inuse);

* Show a data source's configuration

int dsrc_show_config(data_source_t *src);
int dsrc_show_config_byname(const char *name);

* Run or stop a configured instance of a data source

int dsrc_start(data_source_t *src);
int dsrc_stop(data_source_t *name);
int dsrc_run(data_source_t *src);
int dsrc_finish_traffic(data_source_t *data_src, struct _traffic_t, ANALYZER_ACTION action);
int dsrc_finish_flow(flow_t *flow);

Registration methods for user-provided code

int dsrc_register_user_context(data_source_t *src,
    ref_engine_t *context);
int dsrc_register_idle_function(data_source_t *src,
    dsr_callback_idle_func idlefunc);
Abstraction layer for the Data Source subsystem

SSP doesn’t care if the Data Source is implemented as software or hardware

Insert hardware accelerator HERE!
Flow Management

- SSP supports two-level flow acceleration
- Engines can signal dispatcher to ignore flows
  - Dispatcher stops forwarding traffic to that engine thread for duration of flow
- Dispatcher can signal flow manager to “fastpath” a flow if all engines “sign off” on it
  - Fastpath flows stay within the data source
- Flow Slots - state data is stored outside engine threads
- Run-time config can be changed without losing state!
Attribute Manager

- Network map can be kept resident in the engine
- Addressable/updatable in real-time via the snortd command shell
- Enables self-tuning analysis engines
- Tell the network what it’s defending and it’ll figure out how to defend it!
Adaptive (Target Based) Detection
Snort 3.0 Language

- Snort is not a language project!
- Snort’s rules and configuration languages are what is know as a “Domain Specific Language” (DSL)
- Embed a language designed for implementing DSL’s!
- Snort 3.0 is using Lua
Snort 3.0 Language FAQs...

- Will I have to throw out my existing rules?
- No! Snort 2.8.x detection framework is ported!
- Why Lua?
- Designed for the problem space
- Used in Nmap, Wireshark, World of Warcraft, Adobe Photoshop Lightroom, BBEdit, etc
- I heard Snort 3.0 has a command shell?
- Yes! Snort 3.0 is designed to run without stopping
Progress

- Major components of SSP are complete
- Engines
- PEP - Implemented, in testing
- RNA - Ported, in testing
- Snort 2.x - Ported, in testing
- Snort 3.0 - Prototype under development
Timelines

- Open Source 1st Beta released 2Q08
- Snort 2.8.2 engine only
- Open Source initial relase
  - 4Q08
- Snort 3.x engine will debut in 2009
What's Possible?

* New detection engines
  * Target App Filters?
  * Protocol specific?

* New context sources
  * DLP?
  * AV?
  * RSS?
  * ???

* Targeted acceleration
Next Generation Security: Context Enabled & Network Aware
Modular Design - Dynamic Analysis
“Most breaches resulted from a combination of events, rather than a single action:”
Fundamental Change:
Real-time analytics at the point of security detection/enforcement.

Today

Tomorrow

Detection Policy

Event Detection

Context

Event Analysis

Detection Policy

Event Detection

Context

Event Analysis

Fundamental Change: Real-time analytics at the point of security detection/enforcement.