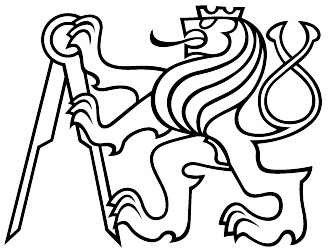


Fail2ban for Windows

Petr Vokáč
(vokac@fjfi.cvut.cz)

October 15, 2015

Introduction

- mimic unix/linux fail2ban python script behavior in windows
  - I have to admit that I'm not completely sure if fail2ban is really the best tool to achieve our goals (operation monitoring, DOS x UDP, ...)
  - anyway, configuration is flexible enough...
- implemented as Windows service(s)
  - main configuration read from application xml file, IPv4 & IPv6
  - modular design, multithreaded, message queues, hard memory limit
  - can run with normal user privileges
    - modified DACL on firewall + granted access to eventlog
  - can be executed as standalone application (debugging)
    - command line options: F2B*.exe -h
    - can be used to install/uninstall and start/stop windows service
- sources – https://github.com/vokac/F2B
  - C# (.Net 4.5 environment – VS2015) and C++/CLI to access WinAPI
  - should work with windows vista and newer (tested on win 7 and 10)

Overview

Overview

linux fail2ban is simple...

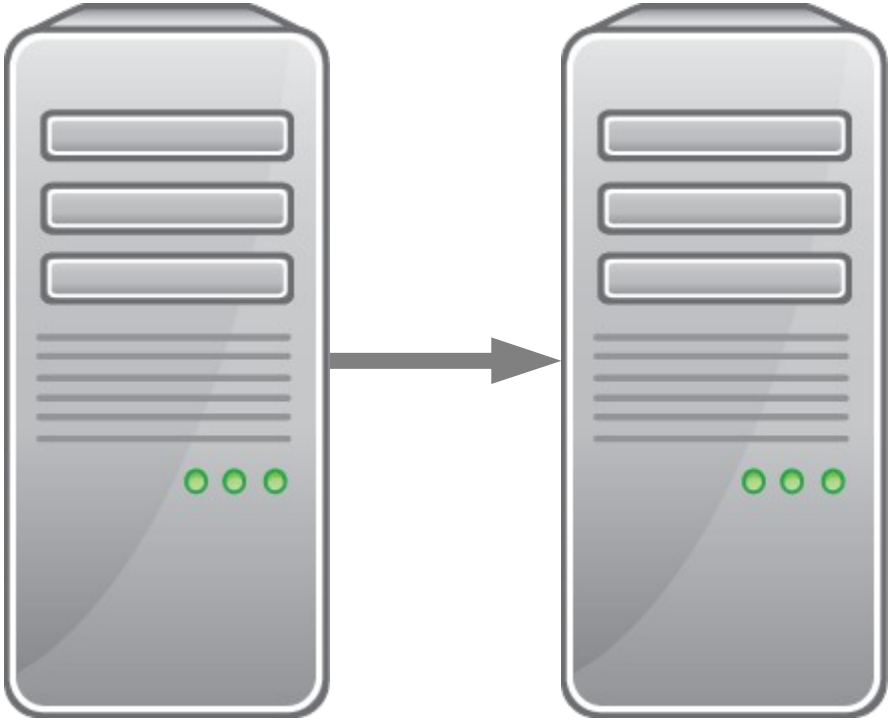
Overview

... let's make it more fancy on windows

Overview

F2B log analyzer

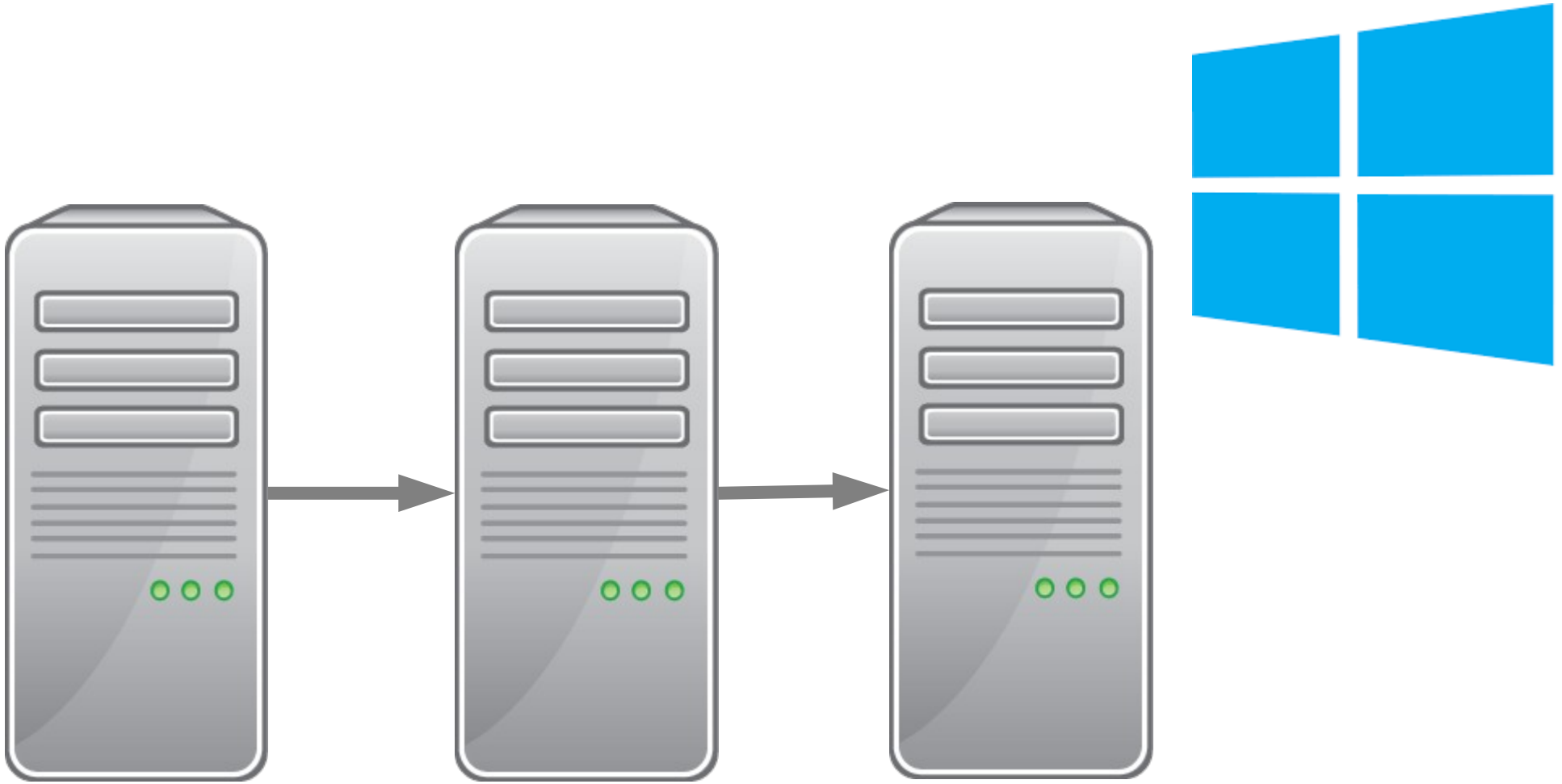
Overview



F2B log analyzer

F2B msg queue

Overview

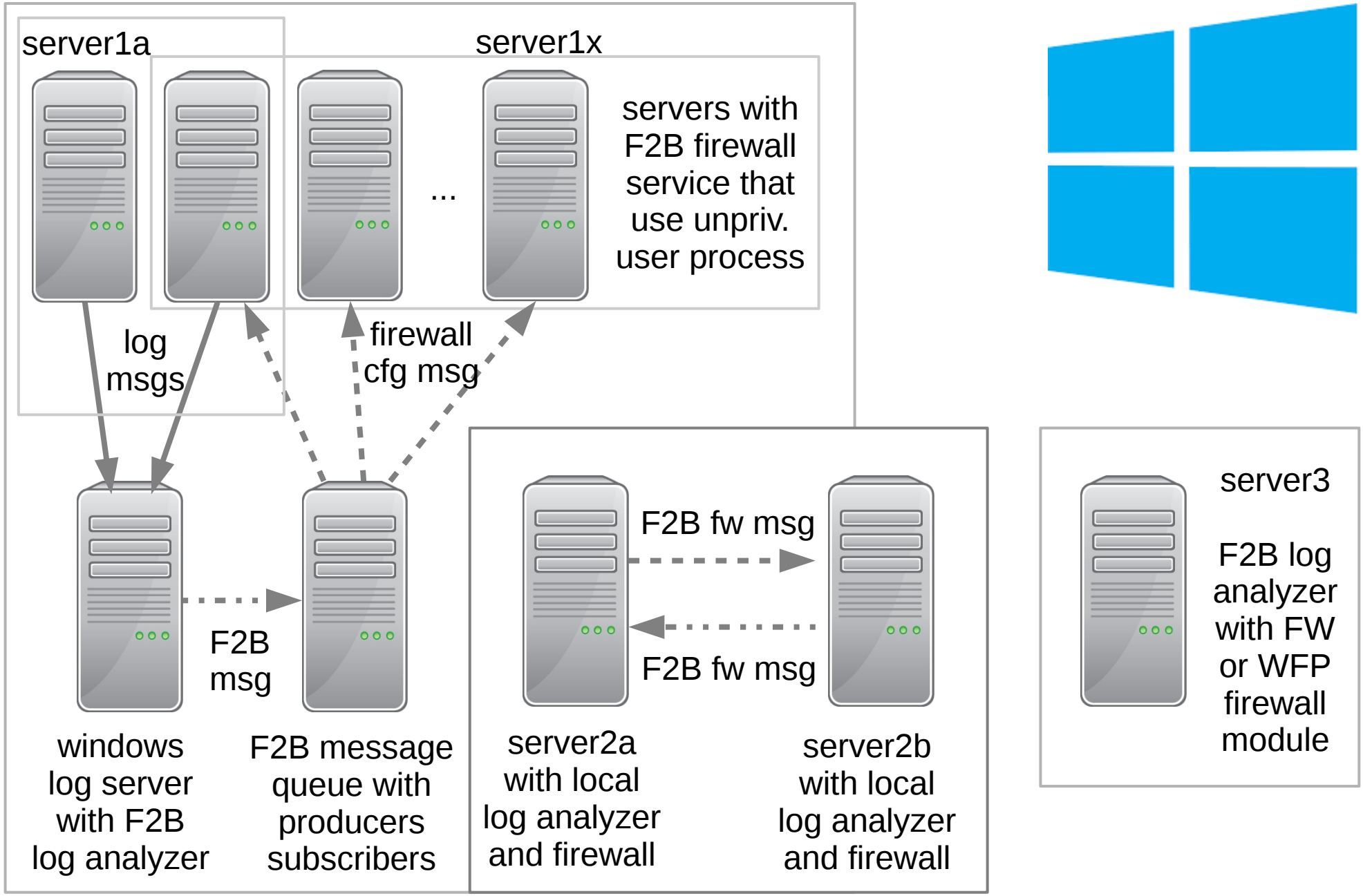


F2B log analyzer

F2B msg queue

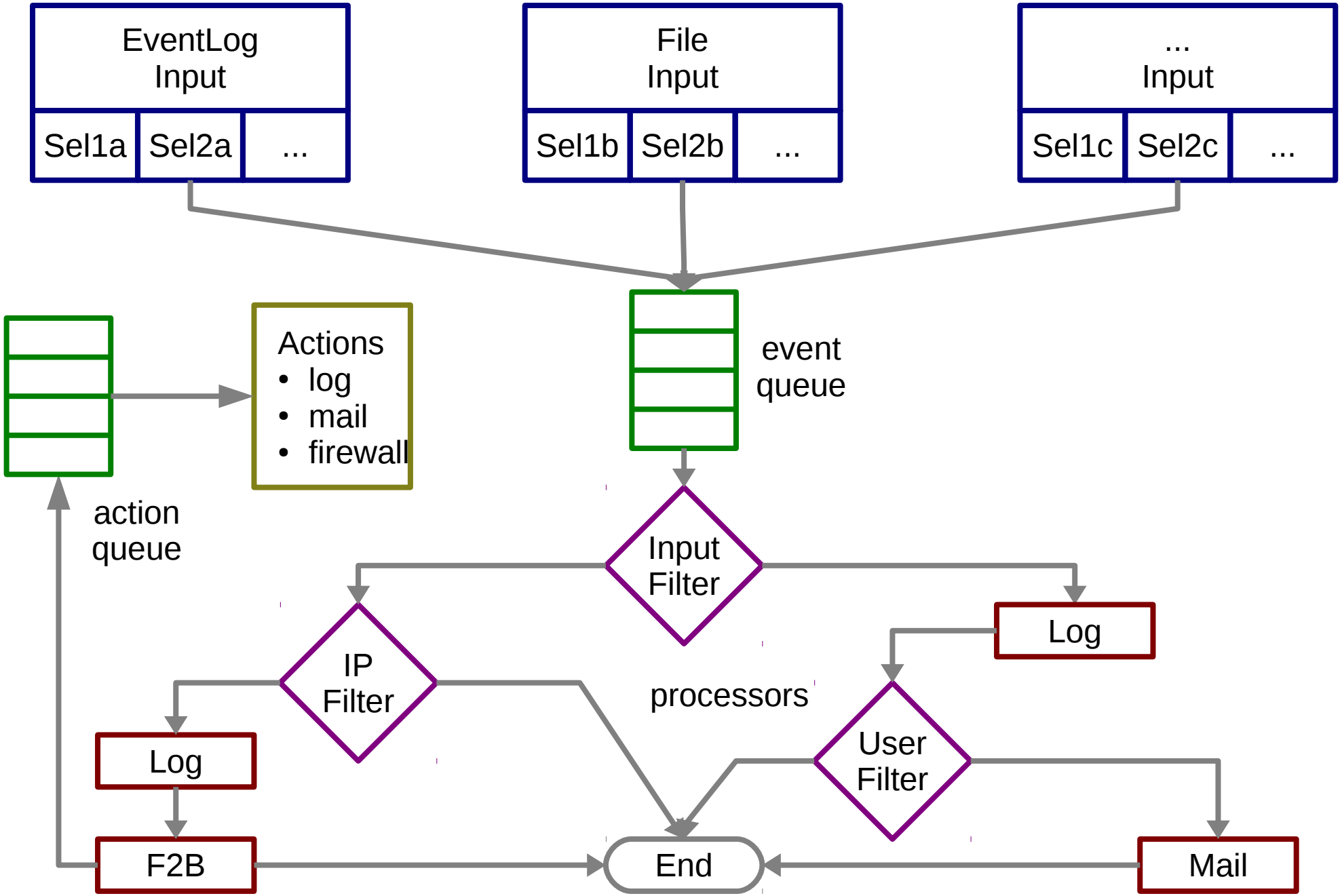
F2B firewall configurator

Overview


server1a
server1x
...
servers with F2B firewall service that use unpriv. user process
log msgs
firewall cfg msg
F2B msg
F2B fw msg
F2B fw msg
windows log server with F2B log analyzer
F2B message queue with producers subscribers
server2a with local log analyzer and firewall
server2b with local log analyzer and firewall
server3
F2B log analyzer with FW or WFP firewall module

F2BLogAnalyzer

F2BLogAnalyzer overview



F2BLogAnalyzer Inputs

- multiple inputs (multiple instances of each input type)
- implemented two input types
  - windows eventlog (“Event Log Readers” or “wevtutil sl /ca:SDDL”)
    - subscribe to local/remote event log
    - callback function for (pre)selected events
    - configuration

    <input name=”unique_input_name” type=”***eventlog***”/>

    <input name=”remote” type=”***eventlog***” server=”hostname”
    domain=”cvut.cz” username=”logadmin” password=”secret”/>
  - file (similar functionality with fail2ban python daemon)
    - read lines appended to simple log file, support file rotation
    - all required data must be logged on one line
    - each file is monitored by one thread
    - configuration

    <input name=”file_input” type=”***logfile***” logpath=”c:\log\file\name.log/>

EventLog – XML representation

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  - <System>
      <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-A5B
      <EventID>4771</EventID>
      <Version>0</Version>
      <Level>0</Level>
      <Task>14339</Task>
      <Opcode>0</Opcode>
      <Keywords>0x8010000000000000</Keywords>
      <TimeCreated SystemTime="2015-06-11T01:09:07.60484
      <EventRecordID>3390326</EventRecordID>
      <Correlation />
      <Execution ProcessID="464" ThreadID="4800" />
      <Channel>Security</Channel>
      <Computer>dc1.vtest.fjfi.cvut.cz</Computer>
      <Security />
    </System>
  - <EventData>
      <Data Name="TargetUserName">vokacpet</Data>
      <Data Name="TargetSid">S-1-5-21-1692801136-2096734761-4156021374-1108</Data>
      <Data Name="ServiceName">krbtgt/VTEST.FJFI.CVUT.CZ</Data>
      <Data Name="TicketOptions">0x50800000</Data>
      <Data Name="Status">0x18</Data>
      <Data Name="PreAuthType">2</Data>
      <Data Name="IpAddress">2001:718:2:1901::852</Data>
      <Data Name="IpPort">47804</Data>
      <Data Name="CertIssuerName" />
      <Data Name="CertSerialNumber" />
      <Data Name="CertThumbprint" />
    </EventData>
  </Event>
```

Audit failure for bad password
TGT request for user vokacpet
from kerberos client address
2001:718:2:1901::852

F2BLogAnalyzer Selectors

- extract useful data from logged event
- different for each input type or input name
  - Optional parameter “processor” (processor for events passing selector)
  - Optional parameter “login” kind (success, failure, unknown)
  - XML query for eventlog + XPath & regexp (with named params)

```
<selector name=”unique_name” input_type=”eventlog”>
  <query><![CDATA[<Select Path=”Security”>*[System/EventID=4771]</Select>]]>
  <address xpath=”Event/EventData/Data[@Name='IpAddress']”>
    <![CDATA[<?<address>.+)]]>
  </address>
  <username xpath=”Event/EventData/Data[@Name='TargetUserName']”/>
  <port ... />
...
</selector>
<selector name=”unique_name” input_name=”file_input”>...</selector>
```

    - https://msdn.microsoft.com/en-us/library/bb399427%28v=VS.90%29.aspx
    - Build filter: start `eventvwr` → Action → Create Custom View → “define your filter” → XML (copy <Query>...</Query> as F2B query)
  - regexp for parsing log files (similar to fail2ban)

F2BLogAnalyzer Queue

- classic producer/consumer pattern
  - multiple producers (selected inputs + actions) and consumers (processors)
  - queue size
    - don't exhaust all memory by setting limit in configuration file

      <queue><maxsize>10000</maxsize></queue>
  - consumers (number of threads)

      <queue><consumers>10</consumers></queue>
- queue data (some fields can be empty/null)
  - event id (uniq number incremented for each event), event record id
  - input name, selector name
  - timestamp, log hostname
  - client address, client port, username, domain
  - data (full EventLog object, full log line)

F2BLogAnalyzer Processors

- log event processors, filters and actions
- starts with processor name from selector or with first processor
- simple interface
  - Start() – Execute(log entry) – Stop()
  - Execute by default protected by lock (thread-safe), can be disabled if you are sure that your method is implemented thread-safe
  - Additional data in log entry dictionary (SetProcData, GetProcData)
- driven by configuration file (navigation using “goto” configuration)

```
<processor name=”unique_name” type=”ClassName”>
  <description>arbitrary text</description>
  <options>
    <option key=”name1” value=”value1”><option key=”name2” value=”value2”>
  </options>
  <goto next=”next_processor_name” error=”proc_name_after_error”/>
</processor>
```

Stop is default behavior with no name

By default use next processor from config file

F2BLogAnalyzer Processors (basic)

- Label
  - can be used as “goto” target for next/error/success/failure label

    <processor name=”my_label” type=”Label”/>
- Stop
  - default target for empty error option in goto element
  - unknown label used in goto element also terminate event processing

    <processor name=”terminate” type=”Stop”/>
- Parallel
  - run list of processors parallel
  - resend event to the queue with different first processor name
  - final/stop processor (don't call next processor ... goto next is empty)

    <options>
      <option key=”processors” value=”proc_name1,proc_name2,...”/>
    </options>

F2BLogAnalyzer Processors (filters)

- Filters with two additional “goto” targets (success / failure) – Bool
- Input filter (input / selector)
  - first processor name can be directly specified as selector attribute
  - regexp to choose more inputs and/or selectors at once (e.g. “(i1|i2|...)”)

    <options><option key=”type” value=”type_regexp”/><option key=”input” value=”input_regexp”/><option key=”selector” value=”selector_regexp”/></options>
- IP range (filter log events by client address)
  - Range (create ${Range.Mail} variable if “Mail” option exists)

    <ranges><range network=”147.32.0.0/16”/><range network=”2001:718:2::/48”/></ranges>
    <options><option key=”mail” value=”f2b@example.com”/></options>
  - RangeFile (auto-reload, create ${RangeFile.Mail} variable)
    - file format (email optional): IP_range”separator”mail1,mail2,...

    <options><option key=”filename” value=”c:\F2B\IP.range”/>
    <option key=”separator” value=”; ”/></options>

F2BLogAnalyzer Processors (filters)

- Account
  - don't ban service IP address for existing/disabled/locked accounts
  - use global account configuration – files, ldap/AD + caching

```
<accounts><account name=”account_src”>...</account></accounts>
<options>
  <option key=”account” value=”account_src”/>
  <!-- exists,locked,disabled,locked|disabled -->
  <option key=”status” value=”exists”/>
</options>
```

- Case
  - build next processor name from template
  - use error target in case produced name is not valid processor name

```
<options>
  <option key=”template” value=”${Event.Input}_${Event.Selector}”/>
</options>
<goto error=”default_label_for_undefined_names”/>
```

F2BLogAnalyzer Processors (filters)

- Login
  - filter events by ${Event.Status}
    - comes from selector attribute “login”
    - distinguish “login success” x “login failure” x “unknown”
  - more important functionality
    - defines ${${Login.Last}.Success} and ${${Login.Last}.Failure}
    - keeps login status and client address history
      - only for addresses with at least on successful login
      - can be used with “Case” processor
        - template: ${${Login.Last}.Success:=no_successful_login}

```
<options>
  <option key=”maxsize” value=”100000”/>
  <option key=”findtime” value=”86400”/>
  <option key=”count” value=”24”/>
  <option key=”ipv4_prefix” value=”32”/>
  <option key=”ipv6_prefix” value=”64”/>
  <option key=”state” value=”c:\F2B\login.state”/>
</options>
```

F2BLogAnalyzer Processors

- Logger
  - store event (processor) data in text file according template

```
<options>
  <option key=”file” value=”c:\log\file\name.csv”/>
  <option key=”size” value=”1073741824”/><!-- rotate log files -->
  <option key=”history” value=”4”/><!-- keep last 4 log files -->
  <option key=”synchronized” value=”true”/><!-- sync log lines -->
  <option key=”template” value=”${Event.Id};${Fail2ban.Address}\r\n”/>
</options>
```

- LoggerSQL
  - store event (processor) data in SQL database using ODBC (32/64bit)

```
<options>
  <option key=”odbc” value=”ODBC connection string”/>
  <option key=”table” value=”f2b”/>
  <option key=”columns” value=”col1,col2,col3”/>
  <option key=”column.col1” value=”${Event.Id}”/>
  ...
</options>
```

F2BLogAnalyzer Fail2ban

- Processor count number of events during given period
- Action modules called after reaching defined treshold
- Configuration options
  - support for IPv4/IPv6
    - ipv4_prefix (default /32)
    - ipv6_prefix (default /64)
  - findtime
  - history (“all” – used by unix fail2ban, “one”, “fixed”)
  - multiple tresholds
    - function + maxretry + repeat, bantime, action
      - action → “special” processor
      - only “one action” for each treshold
        - actually reference to chain of processor terminated by “Stop” processor

    <processor name=”a1” type=”Label”/>...<processor name=”s1” type=”Stop”/>
  - state file – load/save client address F2B history on start/stop

F2BLogAnalyzer Fail2ban

- Processor count number c
- Action modules called afte
- Configuration options
  - support for IPv4/IPv6
    - ipv4_prefix (default /32)
    - ipv6_prefix (default /64)
  - findtime
  - history (“all” – used by un
  - multiple tresholds
    - function + maxretry + re
      - action → “special” pro
      - only “one action” for e
        - actually reference

    <processor name=”a1” ty
  - state file – load/save clien

```
<processor name="fail2ban" type="Fail2ban">
  <description>Test fail2ban processor</description>
  <options>
    <option key="findtime" value="600"/>
    <option key="ipv4_prefix" value="32"/>
    <option key="ipv6_prefix" value="64"/>
    <option key="history" value="all"/>
    <!--
    <option key="history" value="all"/>
    <option key="history" value="one"/>
    <option key="history" value="fixed"/>
    <option key="history.fixed.count" value="10"/>
    <option key="history" value="rrd"/>
    <option key="history.rrd.count" value="5"/>
    <option key="history.rrd.repeat" value="2"/>
    -->
    <option key="tresholds" value="test,soft,hard"/>
    <option key="treshold.test.function" value="simple"/>
    <option key="treshold.test.maxretry" value="0"/>
    <option key="treshold.test.repeat" value="0"/>
    <option key="treshold.test.bantime" value="300"/>
    <option key="treshold.test.action" value="action_test"/>
    <option key="treshold.soft.function" value="simple"/>
    <option key="treshold.soft.maxretry" value="7"/>
    <option key="treshold.soft.repeat" value="0"/>
    <option key="treshold.soft.bantime" value="-1"/>
    <option key="treshold.soft.action" value="action_soft"/>
    <option key="treshold.hard.function" value="simple"/>
    <option key="treshold.hard.maxretry" value="10"/>
    <option key="treshold.hard.repeat" value="0"/>
    <option key="treshold.hard.bantime" value="600"/>
    <option key="treshold.hard.action" value="action_hard"/>
  </options>
</processor>
```

F2BLogAnalyzer Processors (actions)

- Using same processor interface and also same internal queue
  - same processors can be reused (e.g. logging)
  - action events should use higher priority or different queue (planned)
- Mail
  - use global SMTP configuration (SMTP AUTH supported)

```
<smtp>
  <host>smtp.example.com</host>
  <port>25</port>
  <ssl>false</ssl>
  <!-- <ssl>true</ssl><username>user</username><password>secret</password> -->
</smtp>
```

  - content is template based
    - sender, recipient, subject and body

```
<processor name="action_mail_test" type="Mail">
  <options>
    <option key="sender" value="helpdesk@example.com"/>
    <option key="recipient" value="f2b-admin1@example.com,${${RangeFile.Last}.Mail:=}”/>
    <option key="subject" value="[F2B] Fail2Ban[${Fail2ban.Last}] reached ${${Fail2ban.Last}.Treshold} ..."/>
    <option key="body" value="module: ${Fail2ban.Last}...”/>
  </options>
  <goto on_error_next="true"/>
</processor>
```

F2BLogAnalyzer Processors (actions)

- Fail2banMSMQ
  - use MSMQ to talk with F2BQueue or directly F2BFirewall daemon
  - requires installed MSMQ (vs. `F2BLogAnalyzer.nomsmq.exe`)

```
<processor name="action_fail2ban_msmq" type="Fail2banMSMQ">
  <options>
    <option key="queue_name" value=".\private$\F2BProducer”/>
    <option key="max_ignore" value="60”/> <!-- ignore same ban msgs for 60 seconds after we sent one -->
    <option key="bantime" value="600”/> <!-- default bantime if internal variable Fail2ban.bantime not set -->
  </options>
  <goto on_error_next="true"/>
</processor>
```

- Fail2BanCmd
  - execute arbitrary command with required arguments
    - F2BFirewall command line can change WFP
      - really poor performance (just an example)

```
<processor name="action_fail2ban_cmd" type="Fail2banCmd">
  <options>
    <option key="path" value="c:\F2B\F2BFirewall.exe"/>
    <option key="args" value="add-filter /address ${${Fail2ban.Last}.Address}/${${Fail2ban.Last}.Prefix}
                                          /expiration ${${Fail2ban.Last}.Expiration}”/>
    <option key=”max_ignore” value=”60”/>
  </options>
</processor>
```

F2BLogAnalyzer Processors (actions)

- Fail2banWFP
  - direct local firewall filter rules configuration
  - F2BWFP.dll
    - C++/CLI interface to WFP API
    - depends on Visual C++ Redistributable for Visual studio 2015
    - great performance (two order of magnitude faster than FirewallAPI)
      - different filtering layer (IP packet / application)
- Fail2banFW
  - another firewall module
  - use much less optimal FirewallAPI.dll
  - poor performance with more than few thousands F2B filter rules
  - `F2BLogAnalyzer.standalone.exe`
    - no MSMQ support → no dependency on MSMQ installation
    - no WFP support → doesn't depend on C++ Redistributable installation

F2BLogAnalyzer variables and expr.

- Can be used in module parameters parsed as template
- Variables
  - syntax: ${variable:=defaul_value}
  - list of most useful variables (see F2BLA App.config.full for details):
    - Event variables: Id, RecordId, Timestamp, TimeCreated, Hostname, Input, Selector, Address, Port, Username, Domain, Status
    - Fail2ban vars.: All, Last, Treshold, Address, Prefix, Bantime, Expiration
    - Range/RangeFile vars.: All, Last, Range, Mail
    - Login vars.: All, Last, Success, Failure
- Expression (support for very simple expression evaluation)
  - syntax: $(expression:=error_value)
  - examples
    - $(1+2)*3 / 4
    - $(13 & 10)
    - $(${Login.Last}.Success} > 5)

F2BLogAnalyzer Executables

- `F2BLogAnalyzer.exe`
  - includes all modules
- `F2BLogAnalyzer.nomsmq.exe`
  - no modules with dependency on MSMQ (e.g. Fail2banMSMQ)
  - works without MSMQ installation
- `F2BLogAnalyzer.nowfp.exe`
  - no modules with dependency on F2BWFP.dll (e.g. Fail2banWFP)
  - works without C++ Redistributable installation
- `F2BLogAnalyzer.standalone.exe`
  - no modules with dependency on MSMQ or F2BWFP.dll
  - works with default windows installation
    - no additional requirements!
      - except .Net 4.5 requirement on older windows versions
    - non-optimal performance (see details about Fail2banFW module)

F2BLogAnalyzer Executables

- User privileges to access windows event log
  - local administrator has sufficient privileges
  - F2BLogAnalyzer.exe can be executed as unprivileged user
    - user is member of “Event Log Readers” group
    - or user was added in event log SDDL

```
wmic useraccount where name='username' get sid
wevtutil gl LOG_NAME
wevtutil sl LOG_NAME /ca:”original SDDL”(A;;0x3;;;”SID”)
```

- Command line options

```
F2BLogAnalyzer.exe -h
F2BLogAnalyzer.exe install -c c:\F2B\F2BLogAnalyzer.config \
          [-u user] [-l INFO] [-g c:\F2B\F2BLogAnalyzer.log] \
          [--log-size 1048576 --log-history 4] \
          [-x 1073741824]
F2BLogAnalyzer.exe start
F2BLogAnalyzer.exe stop
F2BLogAnalyzer.exe uninstall
F2BLogAnalyzer.exe run -c c:\F2B\F2BLogAnalyzer.config
```

F2BQueue

F2BQueue

- MSMQ used for communication by F2B services
  - can be used over network (and can be secured by Krb or Certs)
  - privilege separation, can be started with user privileges
  - not installed by default on desktop windows
  - not started by default on servers
  - only low-level Queue functions for point-to-point messaging
    - no producer/subscriber pattern
- Input F2B configuration data from Fail2banMSMQ module
- Data forwarded to all F2B Firewall subscribers
  - on first F2B Firewall receives all (non-expired) configurations
  - subscription must be regularly refreshed (done automatically)
- Save/restore queue information
  - read data file on startup
  - write data on shutdown and periodically when service runs

F2BQueue

- Configuration only by command line options

```
F2BQueue.exe -h
F2BQueue.exe install -H . -p F2BProducer -r F2BSubscription \
          -s c:\F2B\queue.dat -i 300 -n 150 \
          [-u user] [-l INFO] [-g c:\F2B\F2BQueue.log] \
          [--log-size 1048576 --log-history 4] \
          [-x 1073741824]
F2BQueue.exe start
F2BQueue.exe stop
F2BQueue.exe uninstall
F2BQueue.exe run -H . -p F2BProducer -r F2BSubscription \
          -s c:\qdata -i 300 -n 150
```

F2BFirewall

F2BFirewall

- firewall component (x pure firewall F2BLogAnalyzer module)
- WFP (Windows Filtering Platform) API (Vista+)
  - can't use FirewallAPI.dll (used by windows firewall GUI)
    - incredibly slow (loading 100k rules takes more than 30 minutes)
    - probably caused by using “application layer firewall”
  - WPF == “ip utils, iptables, ipsec, libnetfilter*, libpcap, ss”
    - `netsh wfp show state` (no utility for general modification of WFP?!)
  - WFP doesn't have C# API (not huge number of “users”)
  - wrapped in managed C++/CLI DLL (x86, x64)
    - provider, layer, sublayer, filter – code to configure security descriptors
    - rules can be active within session, _till reboot_, forever
  - using different (“lower”) filtering layer than FirewallAPI.dll
    - FWPM_LAYER_INBOUND_IPPACKET_V4
    - FWPM_LAYER_INBOUND_IPPACKET_V6
  - fast compared to FirewallAPI.dll (100k rules in 30s), fast packet proces

F2BFirewall

- Service management

```
F2BFirewall.exe -h
F2BFirewall.exe install -H . -r F2BSubscription -n 150 -i 240 \
          [-u username] [-l INFO] [-g c:\F2B\F2BFirewall.log] \
          [-x 1073741824]
F2BFirewall.exe start
F2BFirewall.exe stop
F2BFirewall.exe uninstall -u username
F2BFirewall.exe run -H . -p F2BProducer -n 150
F2BFirewall.exe run -H . -r F2BSubscription -n 150 -i 240
```

- F2B WFP management

```
F2BFirewall.exe list-filters
F2BFirewall.exe add-filter -a 192.0.2.0/24 [-e “expiration time”] \
          [-w weight] [-permit] [-persistent]
F2BFirewall.exe remove-filter -f “filterId”
F2BFirewall.exe remove-filters
F2BFirewall.exe remove-expired-filters
F2BFirewall.exe remove-unknown-filters
F2BFirewall.exe add-wfp
F2BFirewall.exe remove-wfp
F2BFirewall.exe list-privileges
F2BFirewall.exe add-privileges -u username
F2BFirewall.exe remove-privileges -u username
```

Using F2B

Standalone Fail2ban installation

- Install .Net 4.5 (not included in Windows Vista, 7, 2008, 2008 R2)
- Install Visual C++ Redistributable for Visual Studio 2015 (x86)
- Initialize WFP (add new Provider and SubLayer)

```
c:\F2B\F2BFirewall.exe add-wfp
```

- Install Fail2ban service (use “run” instead of “install” for debugging)

```
c:\F2B\F2BLogAnalyzer.exe install \
          -c c:\F2B\F2BLogAnalyzer.exe.config \
          -g c:\F2B\F2BLogAnalyzer.log -l ERROR \
          --log-size 1048576 –log-history 4
```

- Configure Fail2banWFP as one of the Fail2ban processor action
- Test can be done using (run several times to reach F2B treshold)

```
c:\F2B\LogEvents.exe range 192.0.2.204 192.0.2.205
```

- Show current firewall configuration done by F2B

```
c:\F2B\F2BFirewall list-filters
```

- Whitelist IP (range) – this IP can't be banned by F2B (permit rule)

```
c:\F2B\F2BFirewall --address 192.0.2.0/28 --weight 2^64-1 --permit
```

Standalone Fail2ban performance

- Test performed on 100k test events
  - Intel Quad Q9550 @ 2.83GHz, 4GB RAM (± 2009 desktop)

```
LogEvents.exe range 10.0.0.0 10.1.134.160
```

Results 1 (10) thread	Memory usage [MB]	LogEvents run-time [s]	F2BLA run-time [s]
no processor		50	50
logger+range	9 (17)	60 (61)	60 (61)
...+fail2ban	38 (58)	65 (~~84~~)	65 (~~84~~)
...+fail2banWFP	200 (210)	70 (~~80~~)	81 (~~104~~)

- Almost 2/3 of test time (50s) spend by windows logging
- CPU usage close to 100%
- Analyzing 100k log events with debug build of the F2B
  - 20s spend in F2BLA processors
  - 10s spend in WFP calls that added firewall filter rules
  - significant (private) memory usage – I expected ~ ½ → 30MB/100k

Distributed Fail2ban installation

- Install .Net 4.5 (not included in Windows Vista, 7, 2008, 2008 R2)
- Install Visual C++ Redistributable for Visual Studio 2015 (x86)
  - only required by F2BFirewall.exe (use F2BWFP.dll)
- MSMQ configuration
  - install Microsoft Message Queue (Turn Windows features on or off)
  - configure firewall on machine with F2BQueue.exe
    - open 1801/TCP for hosts with F2BLogAnalyzer and F2BFirewall
  - securing MSMQ communication(?) – not implemented
    - use secured transport layer
      - user account (Kerberos) can be used for machines joined in domain
      - client/server (self-signed trusted) certs from machine certificate store
        - allow access to private key only to user that runs F2B service
    - use message body encryption/signature (PSK or certificates)
- (Remote) logging or event log subscription must be secured
  - probably done by default on Windows (prevent log injection)

Distributed Fail2ban installation

- Log analyzer machines (service F2BLA)

```
c:\F2B\F2BLogAnalyzer.exe install \
        -c c:\F2B\F2BLogAnalyzer.exe.config
```

  - Fail2banMSMQ queue_name must point to production queue on message queue machine (F2BQueue.exe “-p” parameter)
- Message queue machine with name `queuehost` (service F2BQ)

```
c:\F2B\F2BQueue.exe install -H . -p F2BProducer \
        -r F2BSubscription -s c:\F2B\queue.dat -i 300 -n 150
```

- Firewall machines (service F2BFW)

```
c:\F2B\F2BFirewall.exe install \
        -H queuehost -r F2BSubscription -n 150 -i 240
```

- Run service as non-privileged user
  - add “-u DOMAIN\username” to the install command line
  - add log analyzer user to “Event Log Readers” group or wevtutil
- Manage service: `sc query F2Bxx & sc qc F2Bxx 4096 & sc start F2Bxx`