Greppin' Logs: Leveling Up Log Analysis

Jon Stewart and Noah Rubin

Prepared by Jon Stewart and Noah Rubin

Empower Results®

Agenda

Empower Results®

Introductions and Guiding Principles

Empower Results®

Whoami

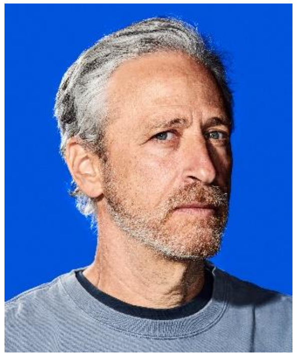
Noah Rubin

Director, DFIR at Stroz Friedberg LLC, an Aon Company.

5 years of experience as an incident responder.

Before Stroz Friedberg, worked as a software engineer and data scientist for data journalism and healthcare startups, as well as a technology company in China.

Passionate about automating and scaling forensic analysis capabilities and open source software.


(Not actually Jon Stewart)

Jon Stewart

VP, Solutions Development at Stroz Friedberg LLC, an Aon Company

Leads and manages a software development team building internal DFIR tools.

Before Stroz Friedberg, Jon co-founded Lightbox Technologies, Inc which created the open source library and (now) command line tool Lightgrep, the “fastest computer forensics search tool.”

Guiding Principles

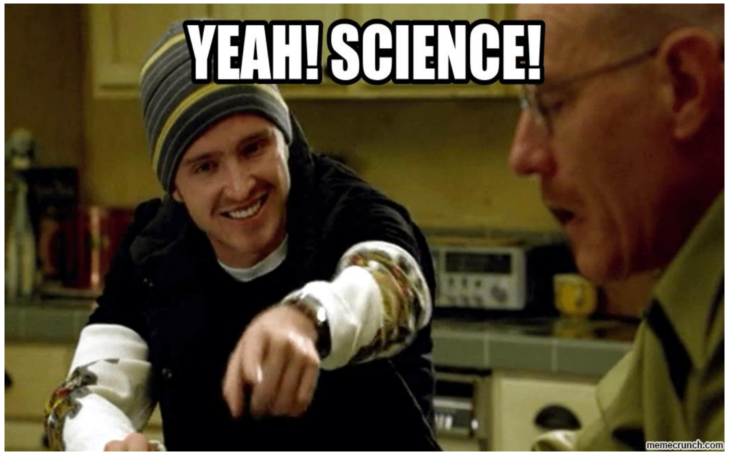
1) Prefer command line/programmatic tools over GUIs.
2) Use simple tools that accomplish a single task well.
3) Compose simple tools and create processing pipelines.
4) Do things fast (enough).
5) Forensics is Data Engineering and Data Science, and should be approached as such.

Empower Results®

Forensics is Data Engineering and Data Science

The Forensic Process

- Data Engineering: shape data into a usable state.
- Data Science: interpret data to extract actionable insights.
- The process:
  1) Receive dataset from client or internal tooling
  2) Define a set of questions to answer given the data available
  3) Determine the necessary engineering and analysis steps to answer those questions
  4) Shape and transform original data into intermediate datasets
  5) Execute the analysis plan and document findings
  6) Repeat
- Key takeaways:
  - The quicker the data engineering process, the sooner you can provide insights during investigations.
  - Try to define your questions before performing data engineering.
  - Never mutate the original dataset.
  - Create intermediate datasets to avoid unnecessary processing and reduce time between engineering and analysis.



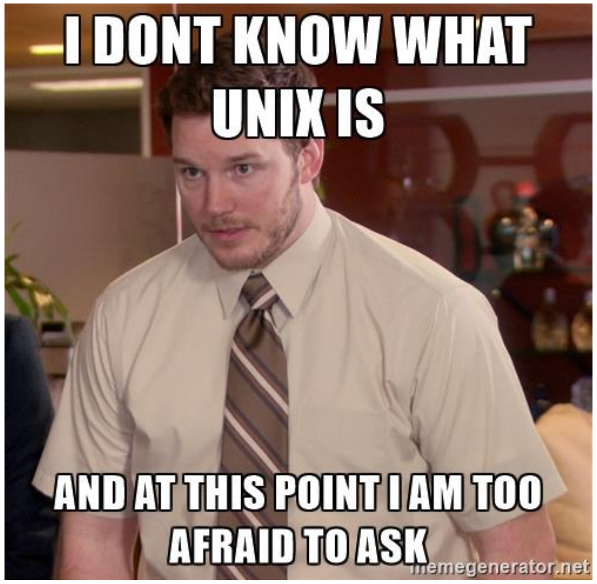
Core Unix Command Line Tools

Prepared by Jon Stewart and Noah Rubin

Empower Results®

Unix: A Brief History

- Unix, a play on the project that inspired it named *Multics*, was first created in 1969-1970 at AT&T's Bell Labs.
- Ken Thompson, Dennis Ritchie, Brian Kernighan, and others at Bell Labs wanted a simple time-sharing system: "What we wanted to preserve was not just a good environment in which to do programming, but a system around which a fellowship could form."
- Many of the features present in Unix in the 1970s are still used heavily today:
  - Man pages
  - IO redirection
  - Device files (part of the PDP-7 Unix file system)
  - **Pipes**
- Many command line utilities today known as the "GNU coreutils" originated from early Unix versions:
  - cat (replacement for the "pr" program present in Multics)
  - sort (ported from Multics)
  - uniq
  - awk (first appeared in Version 7 Unix)
  - sed (first appeared in Version 7 Unix)



Core Unix Command Line Tools

- Key command line tools that should be part of any analysis toolbox:
  - cat
    - Print file to standard output.
  - head
    - Preview content from the top of a file.
  - tail
    - Preview content from the bottom of a file.
  - less
    - View content from a file or stream in a buffer.
  - grep
    - Search for patterns in files or streams.
  - cut
    - Extract sections of text from lines in a file or stream.
  - awk
    - Programming language and runtime for text processing.
  - sed
    - Transform lines of a file or stream.
  - sort
    - Sort lines of a file or stream.
  - uniq
    - Dedupe and aggregate lines of a file or stream.

Processing Structured Data

Prepared by Jon Stewart and Noah Rubin

Empower Results®

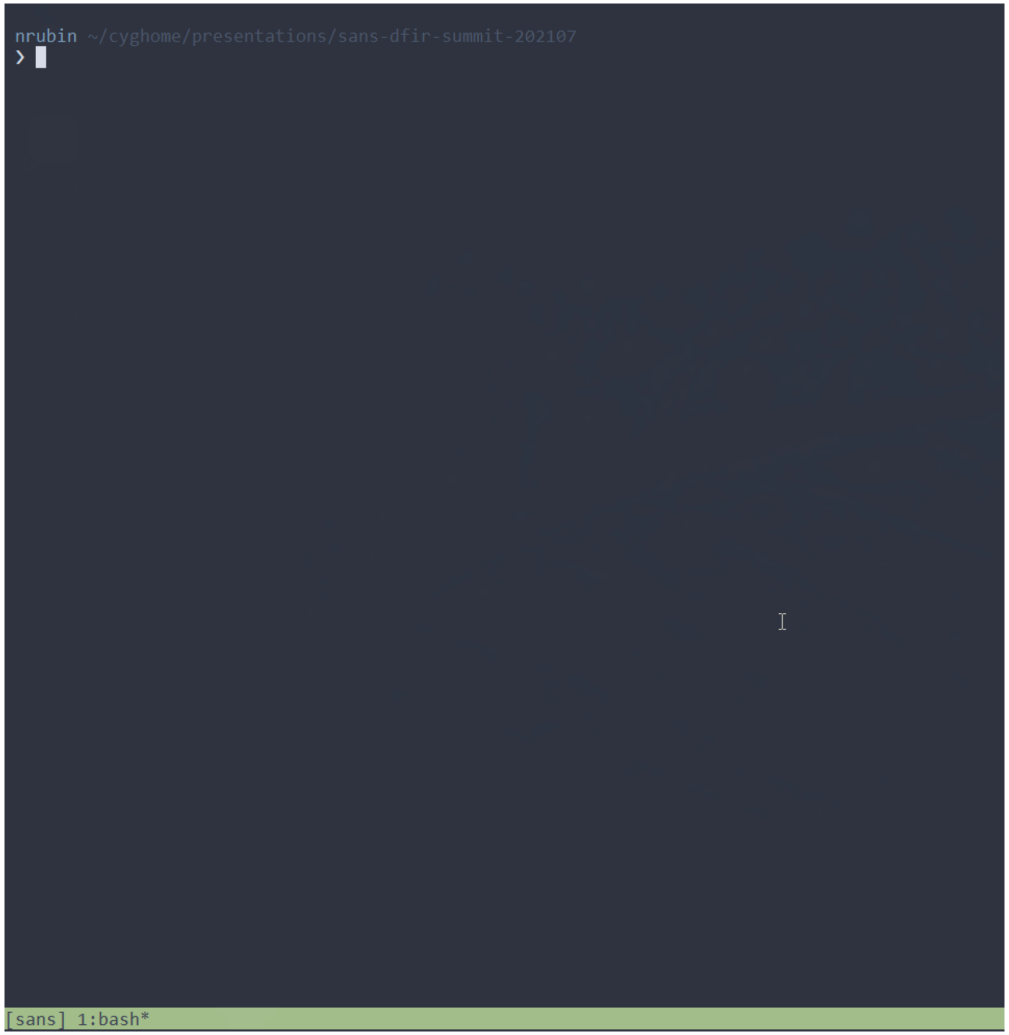
Processing Structured Data: CSV

- csvkit is a purpose-built set of command line tools for processing and analyzing CSV, TSV, and other row-oriented tabular data.
- Every csvkit tool by default normalizes input data by:
  - Removing optional quote characters
  - Changing the delimiter to comma (",")
  - Changing the record delimiter to line feed ("\n")
  - Changing the quote character to double quotation ('"')
  - Changing the encoding to UTF-8
- *csvclean*: report rows with incorrect number of columns.
- *csvformat*: change the delimiter, quote character, line terminator, or escape character.
- *csvlook*: print tabular data in a markdown-friendly format.
- *csvstat*: print descriptive statistics for each columns.
- *csvcut*: like *cut* but handles tabular data intricacies better.
- *csvjoin*: join tabular data using similar to logic to SQL joins.

Processing Structured Data: CSV

Example: Using csvkit to analyze an employee list and associated salary data.

- Check the columns of each dataset
- Preview the first 5 rows to get a sense for what the data look like
- Check each dataset for inconsistencies/errors and fix them
  - Always leave the source data as-is and create a copy with cleaned data
- Get descriptive statistics for each column in the datasets
- Join them together to get a master list of employees and their salaries
- Cut the combined dataset to show an exact match with the original source data



```
nrubin ~/cyghome/presentations/sans-dfir-summit-202107
❯ █

[sans] 1:bash*
```

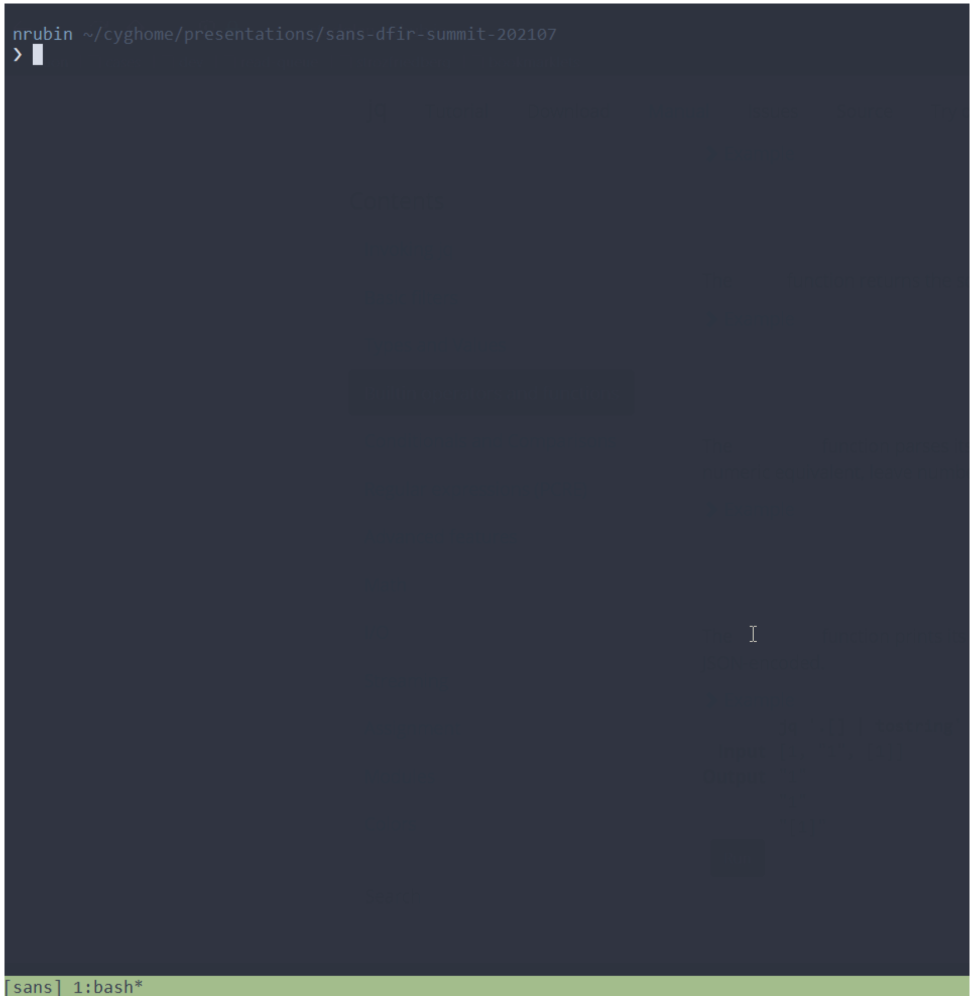
Prepared by Jon Stewart and Noah Rubin

Processing Structured Data: JSON

- *jq* is the command line swiss army knife for dealing with JSON data. Its feature set covers functionality provided by sed, awk, and (sometimes) grep for text processing.
  - Process arrays of JSON objects.
  - Extract specific keys and values from potentially nested JSON objects.
  - Perform arithmetic operations over values.
  - Filter objects based on equality checks, regular expressions, and other complex criteria.
  - Assign values to variables and build complex processing pipelines.
  - Supports a streaming mode to process gigabytes of JSON data quickly.
  - Transform data into CSV with a single operator.

Processing Structured Data: JSON

- Example: Extracting tabular data from compressed AWS CloudTrail log records.
  - Unzip the CloudTrail log file into JSON
  - Check the top-level key names, types, and lengths
  - Check the key names for each “Records” object
  - Generate a CSV from each “Records” object with specific fields of interest
  - Generate descriptive statistics for each column



```
nrubin ~/cyghome/presentations/sans-dfir-summit-202107
❯ █

[sans] 1:bash*
```

Performance Upgrades

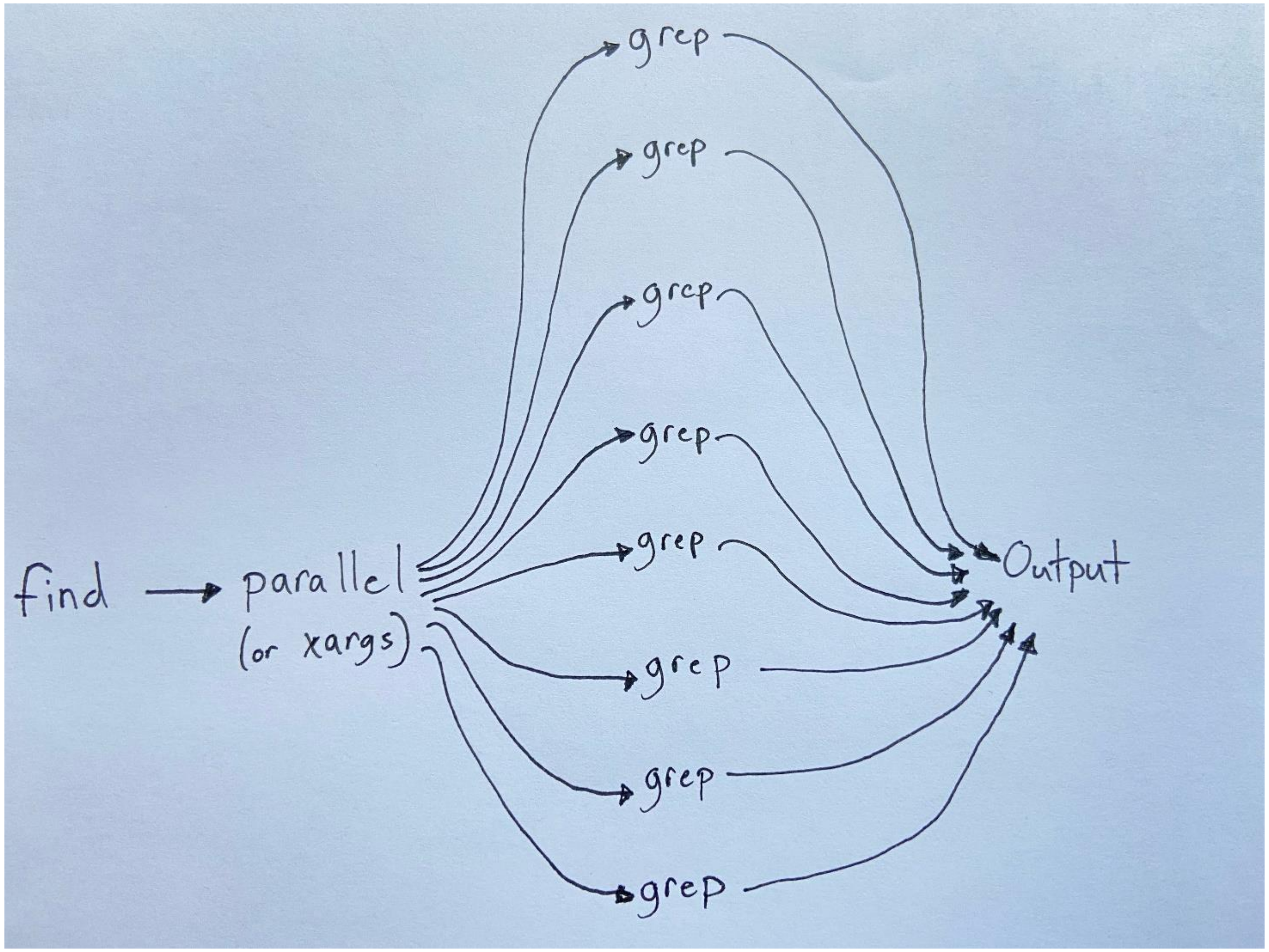
Performance Upgrades

- *find*: query the filesystem for specific files and directories.
  - https://linux.die.net/man/1/find
- *parallel*: run commands in parallel over files or streams.
  - https://www.gnu.org/software/parallel/parallel.html
  - *xargs* can be used similarly, but we recommend *parallel*
- *xsv*: command line tabular data toolkit written in Rust.
  - https://docs.rs/crate/xsv/0.13.0
- simdjson: library for parsing JSON using SIMD instructions written in C++.
  - https://simdjson.org/
- orjson: Python library for parsing JSON, much faster than standard `json` module.
  - https://pypi.org/project/orjson/

Honorable Mentions

- *Visidata (vd)*: command line interactive analysis tool for tabular data. Like Excel, but with the power of Python and in the terminal.
  - https://www.visidata.org/
- *textql*: command line tool for querying tabular data with SQL.
  - *https://github.com/dinedal/textql*

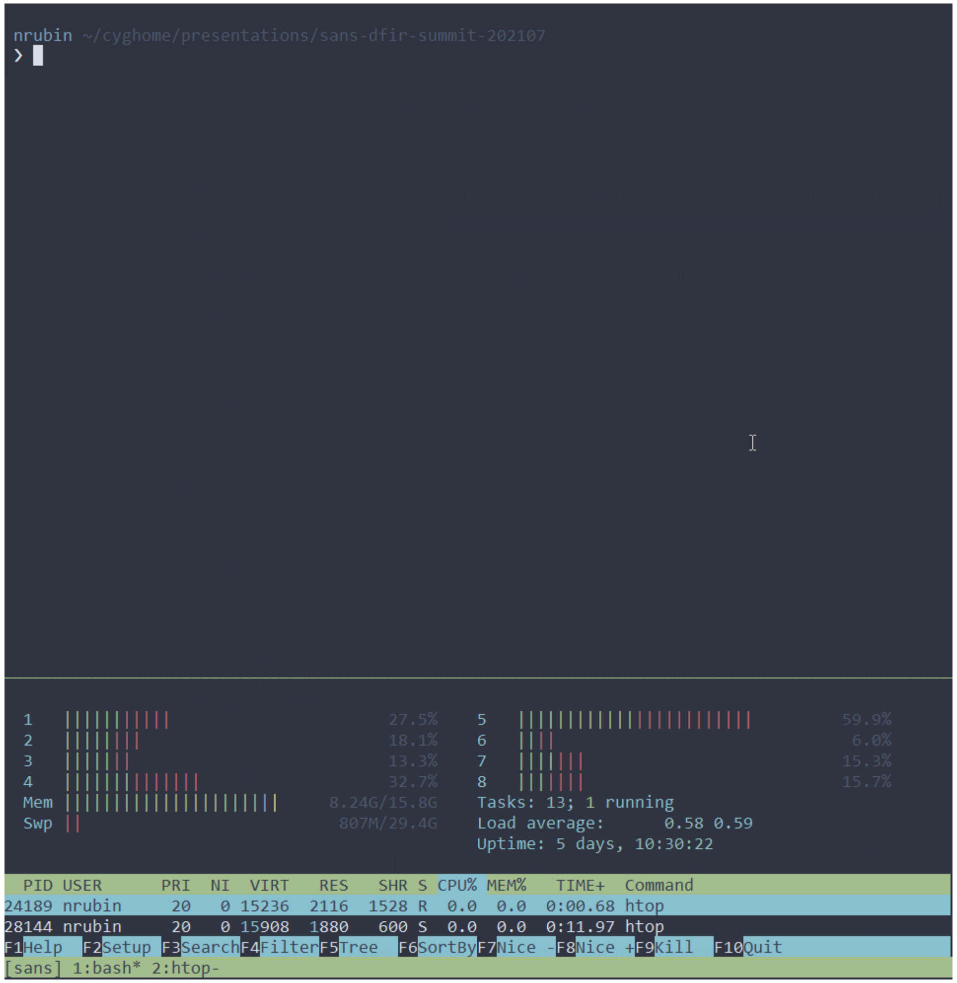
Use your CPU cores! Be smart with I/O!



- Intel Core2: 2006 Single-core work has been obsolete for 15 years.
- M.2 NVMe drives are cheap and *fast*.
- Put output on a different drive to avoid mixed I/O.

Performance Upgrades

- Example: Use *find*, *parallel*, *gunzip*, *jq*, *tr*, and *sed to unzip over 1,000 CloudTrail log files and parse the individual JSON records to CSV.*



```
nrubin ~/cyghome/presentations/sans-dfir-summit-202107
❯ █
```

```
  1   |||||||||||                       27.5%    5   ||||||||||||||||||||||||             59.9%
  2   ||||||||                          18.1%    6   ||||                                  6.0%
  3   |||||||                           13.3%    7   |||||||                              15.3%
  4   ||||||||||||||                    32.7%    8   |||||||                              15.7%
  Mem |||||||||||||||||||||||      8.24G/15.8G   Tasks: 13; 1 running
  Swp ||                            807M/29.4G   Load average:      0.58 0.59
                                                 Uptime: 5 days, 10:30:22

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
24189 nrubin     20   0 15236  2116  1528 R  0.0  0.0  0:00.68 htop
28144 nrubin     20   0 15908  1880   600 S  0.0  0.0  0:11.97 htop
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
[sans] 1:bash* 2:htop-
```

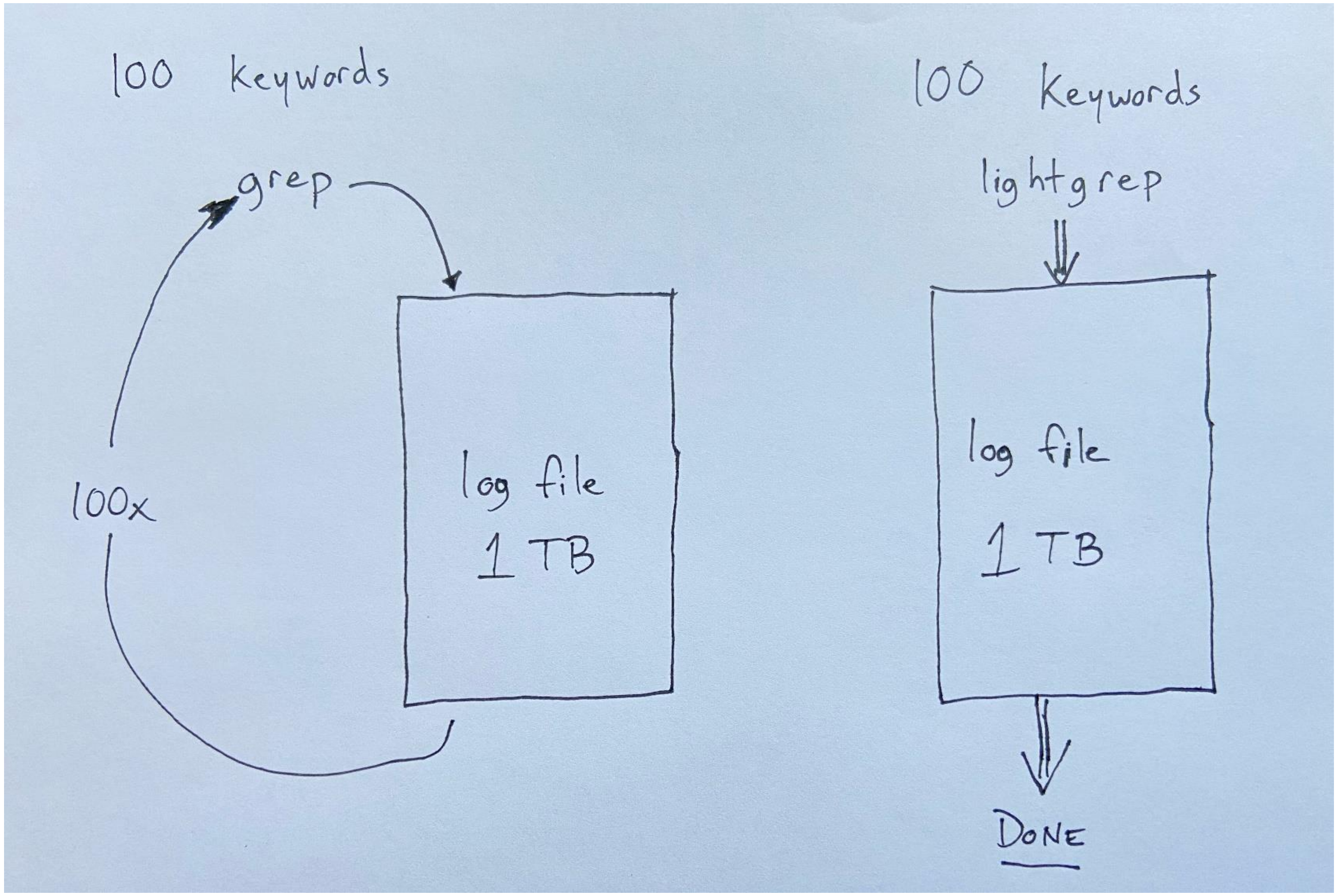
Lightgrep

Lightgrep Overview

- Multipattern regular expression search engine for forensics
- “Lightgrep for EnCase” released 2012
- Library open sourced and integrated into bulk_extractor in 2013
- Standard Perl-compatible syntax
- Robust Unicode support, 100+ encodings
- Powers a lot of forensics processing at Stroz Friedberg

- **Today:** 🎉 Version 1.5 🤩
  - lightgrep command line tool
    - Search binary or text
    - Windows build available
  - Open source license change: GPL v3 → Apache 2
  - https://github.com/strozfriedberg/lightgrep

Why is multipattern search better for logs?



Prepared by Jon Stewart and Noah Rubin

Lightgrep Command Line in Action

- TODO

Scaling with AWS

Prepared by Jon Stewart and Noah Rubin

Empower Results®

Scalable Log Processing with AWS

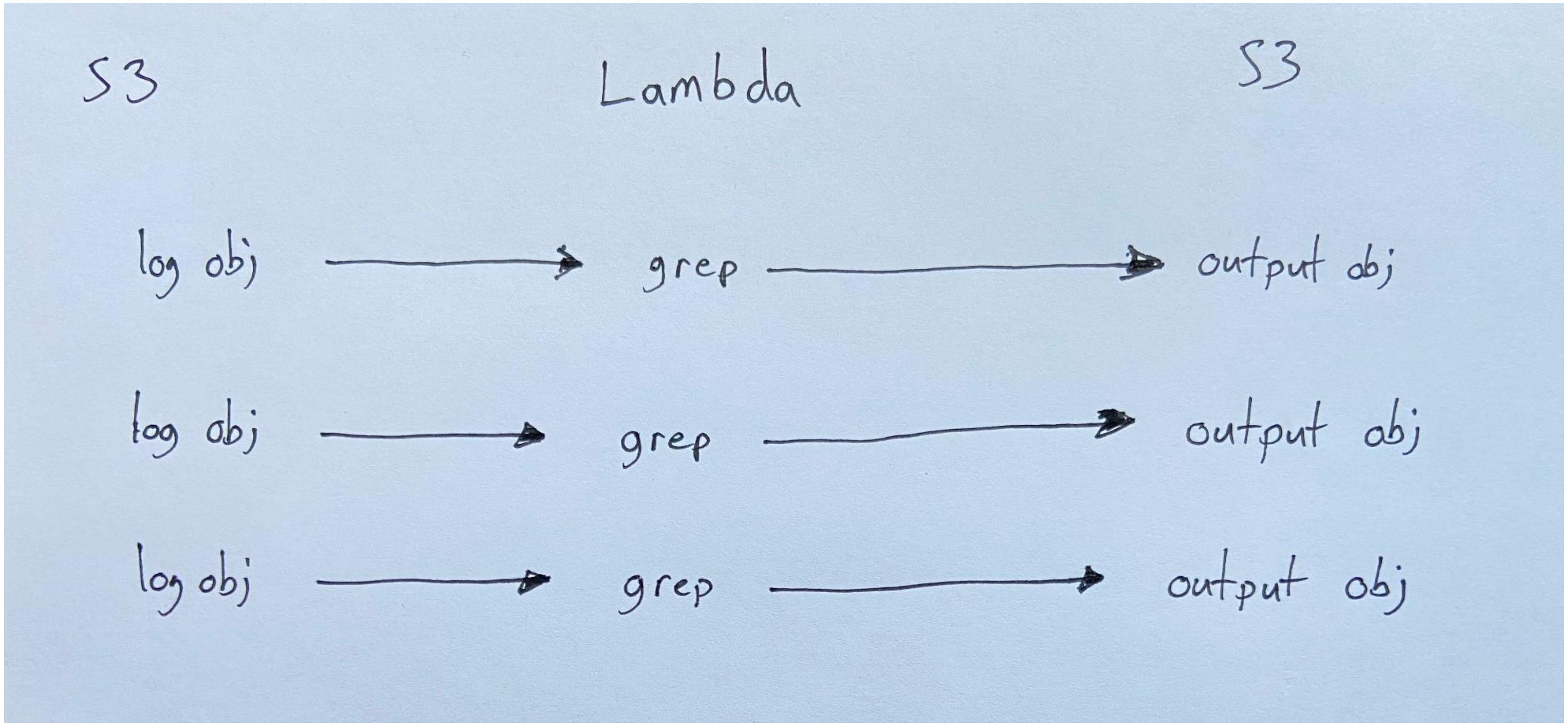
- S3
  - Simple Storage Service
  - Files are “objects”
  - No filesystem, GET/PUT over HTTP
  - 2.3₵ / GB / month, ~$23/TB
  - Safe, minimal chance of data loss

- Lambda
  - Serverless compute
  - Just a function!
  - Different events can trigger function
    - e.g., S3 upload
  - Cheap, and no cost when idle

- CDK
  - Cloud Development Kit
  - AWS DevOps scripting library
  - Configures AWS resources & services
  - Put security rules all in one place

Empower Results®

Storage + CPU == Near-Infinite Processing



Bottleneck-free!

Deploying with AWS CDK

- Example: deploying a full-scale log processing pipeline with S3 and Lambda using AWS CDK.
  - Make sure NodeJS, NPM, and the AWS CDK command line interface are installed
  - Navigate to the “cdk-app” directory within the greppin-logs repo
  - Review the CDK app configuration and Lambda function code
  - Deploy the CDK app using “cdk deploy”
  - Check the stack in AWS CloudFormation

Empower Results®

Resources

- GitHub repository created for this talk: https://github.com/strozfriedberg/greppin-logs
  - Template CDK app
  - Shell scripts with the commands run during examples in this presentation
  - Dockerfile with most of the tools discussed during the presentation
  - Some example datasets
- Lightgrep: https://github.com/strozfriedberg/lightgrep
- GNU coreutils documentation: https://www.gnu.org/software/coreutils/
- csvkit: https://csvkit.readthedocs.io/en/latest/
- jq: https://stedolan.github.io/jq/
- GNU parallel: https://www.gnu.org/software/parallel/
- xsv: https://docs.rs/crate/xsv/0.13.0
- simdjson: https://simdjson.org/
- orjson: https://pypi.org/project/orjson/
- Visidata: https://www.visidata.org/
- textql: https://github.com/dinedal/textql
- AWS CloudTrail: https://aws.amazon.com/cloudtrail/
- AWS CDK: https://aws.amazon.com/cdk/

Appendix: Core Command Line Tools

Prepared by Jon Stewart and Noah Rubin

Empower Results®

Core Command Line Tools: cat

- Description:
  - Command line utility for printing files to standard out and concatenating files. Typically used at the beginning of a pipeline to pipe content to another command.
- Examples:
  - Print file to standard output:

  cat FILENAME

  - Concatenate multiple files into a new file:

  cat FILENAME01 FILENAME02 > NEWFILENAME

Core Command Line Tools: head

- Description:
  - Preview content from the beginning of a file or stream. By default, head treats its input as line-oriented text content and will display the first 10 lines on standard output.
- Examples:
  - Print the first 10 lines of a text file:

  head FILENAME

  - Print the first 100 lines of a text file:

  head -n 100 FILENAME

  - Print the first 30 bytes of a binary file:

  head -c 30 FILENAME

  - Print all but the last 30 bytes of a binary file:

  head -c -30 FILENAME

  - Print the first 5 lines where the line delimiter is the NUL byte instead of newline ("\n"):

  head -n 5 -z FILENAME

Core Command Line Tools: tail

- Description:
  - Preview content from the end (or beginning) of a file or stream. By default, tail treats its input as line-oriented text content and will display the last 10 lines on standard output.
- Examples:
  - Print the last 10 lines of a text file:

  tail FILENAME

  - Print the last 100 lines of a text file:

  tail -n 100 FILENAME

  - Print all but the first line of a text file:

  tail -n +2 FILENAME

  - Print the last 30 bytes of a binary file:

  tail -c 30 FILENAME

  - Print all but the first 30 bytes of a binary file:

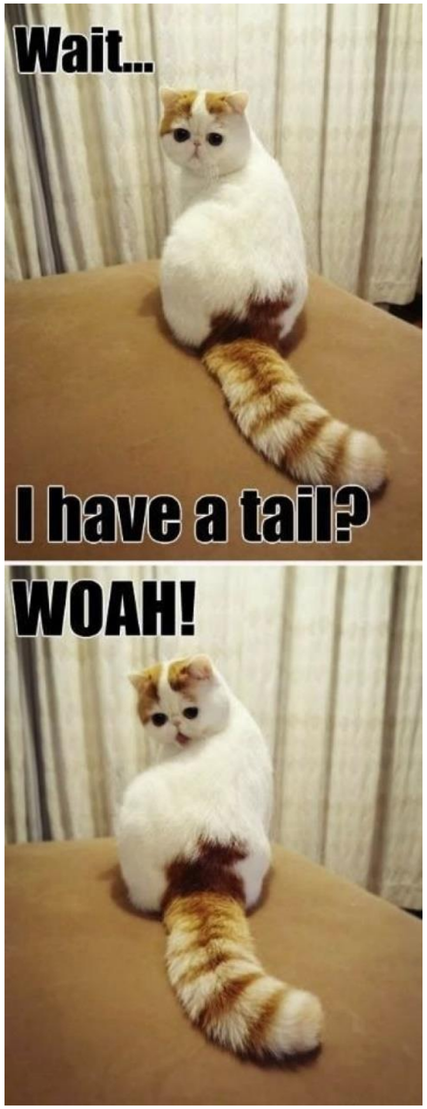
  tail -c -30 FILENAME

  - Print the last 5 lines where the line delimiter is the NUL byte instead of newline ("\n"):

  tail -n 5 -z FILENAME

  - Continuously watch a file for new content:

  tail -f FILENAME



Empower Results®

Core Command Line Tools: less

- Description:
  - Terminal pager program for viewing contents of a file one "screen" at a time. Like *more* but supports forward and backward navigation. Can load file content before the whole file is read.
- Examples:
  - View file contents:

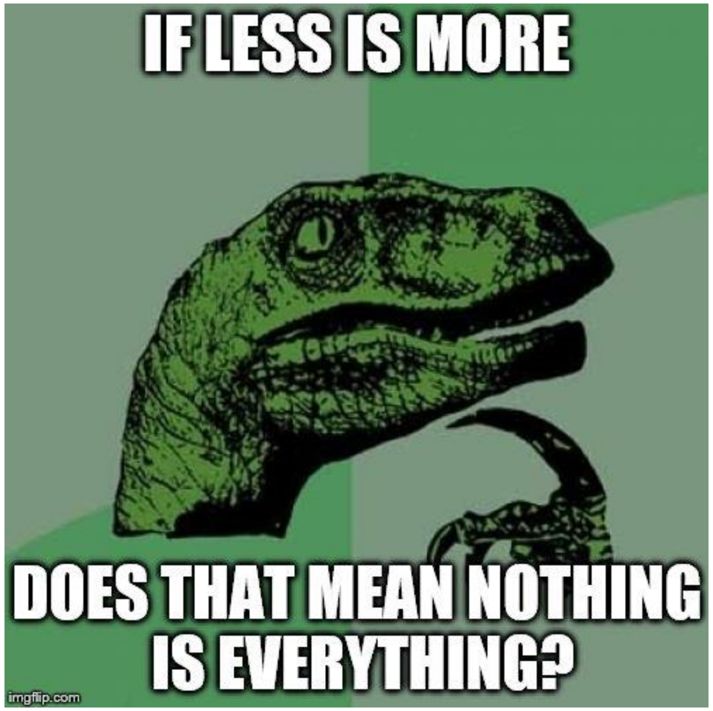
  less FILENAME
  - View standard output stream from another command:

  head FILENAME | less
  - View standard output stream and quit when reach EOF:

  tail FILENAME | less -E
  - View standard output stream and write contents to new file while being viewed:

  cat FILENAME | less –oNEWFILENAME
  - View standard output stream and jump to the first occurrence of a pattern:

  cat FILENAME | less -pPATTERN



Core Command Line Tools: grep

- Description:
  - Line-oriented pattern searching tool. Typically supports basic, extended, and Perl (PCRE) regular expressions. Has in the tens of command line flags that control matching and output behavior.
- Examples:
  - Search for the phrase “hello, world!” in a file:

grep –F ‘hello, world!’ FILENAME

  - Extract all IP addresses from a file:

grep -oE ‘[0-9]{1,3}\.[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}’ FILENAME

  - Search a list of keywords contained in a newline-separated file in another file:

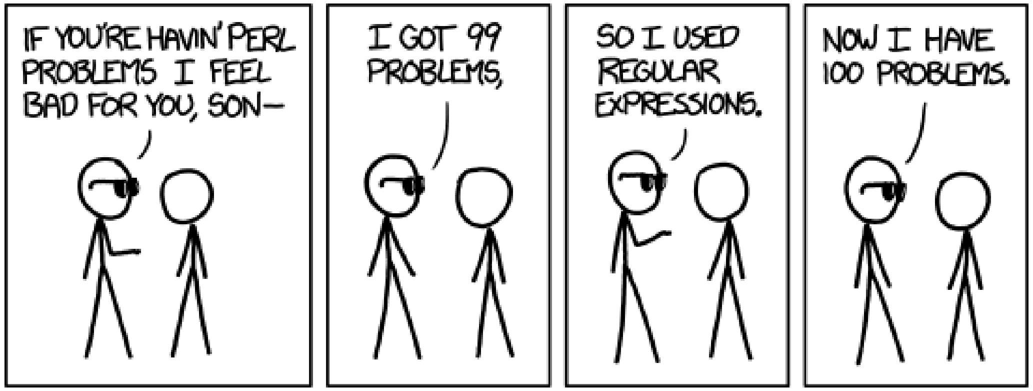
grep –f KEYWORDFILE FILENAME

  - Include matching filenames and line numbers in the output:

grep -Hn ‘pattern’ FILENAME

  - Include all content within 5 lines before or after a matching line:

grep -A5 -B5 ‘pattern’ FILENAME


IF YOU'RE HAVIN' PERL PROBLEMS I FEEL BAD FOR YOU, SON—
I GOT 99 PROBLEMS,
SO I USED REGULAR EXPRESSIONS.
NOW I HAVE 100 PROBLEMS.

AON
Empower Results®

Core Command Line Tools: cut

- Description:
  - Line-oriented tool for extracting parts of each line from files or streams. Does not support complex field delimiters or field reordering.
- Examples:
  - Extract the first and third columns from a TSV file:

  cut -f1,3 FILENAME.tsv

  - Extract the second and fourth columns from a CSV file (without intra-field commas):

  cut -d',' -f2,4 FILENAME.csv

Core Command Line Tools: awk

- Description:
  - Line-oriented programming language and command line tool designed for text processing and data extraction. Supports variables, user-defined functions, arithmetic, aggregation, and other useful features.
- Examples:
  - Print the first and fourth columns in a space-separated file:

  awk '{ print $5 }' FILENAME

  - Print the fourth column in a space-separated file in lines containing the word "bar":

  awk '/{{bar}}/ { print $4 }' FILENAME

  - Sum the last column in a CSV and print the total:

  awk -F',' '{ sum+=$NF } END {print sum}'

  - Print all lines from a CSV where the fifth column equals a specific value:

  awk -F',' '($5 == value)'

  - Convert MySQL dump file to SQLite-compatible dump:

  See https://github.com/dumblob/mysql2sqlite

Core Command Line Tools: sed

- Description:
  - Line-oriented command line tool for text processing and data extraction, similar in nature to *awk*. The most common use case is substitution, often using regular expressions and sometimes in-place.
- Examples:
  - Redact IP addresses from a (log) file in-place:

sed -i‘’ -r 's/[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}/X.X.X.X/g’ FILENAME

  - Remove the first (header) and second rows from a CSV file:

sed ‘1,2d’ FILENAME.csv

  - Remove empty lines or lines with only spaces from a file:

sed “/^[[:space:]]$/d” FILENAME*

  - Combine every two lines in a file into a single line separated by a space:

sed 'N; s/\n / /; P; D’ FILENAME

Core Command Line Tools: sort

- Description:
  - Line-oriented command line tool for sorting content (using the merge sort algorithm).
- Examples:
  - Sort a file using natural ordering and print the first 5 lines:

  sort FILENAME | head -n 5

  - Sort a CSV file by the third column using numeric ordering:

  sort -t ‘,’ -k 3 -n FILENAME

  - Check if a file is already sorted using natural ordering:

  sort -c FILENAME

  - Sort a space-separated file by the first column using four parallel threads:

  sort -k 1 --parallel=4 FILENAME

  - Randomly sort a file (like *shuf*):

  sort -R FILENAME

Core Command Line Tools: uniq

- Description:
  - Line-oriented command line tool for deduplicating and aggregating text data. Typically used together with *sort* because it only dedupes already-sorted data.
- Examples:
  - Print unique values with counts of the fifth column in a CSV sorted descending:

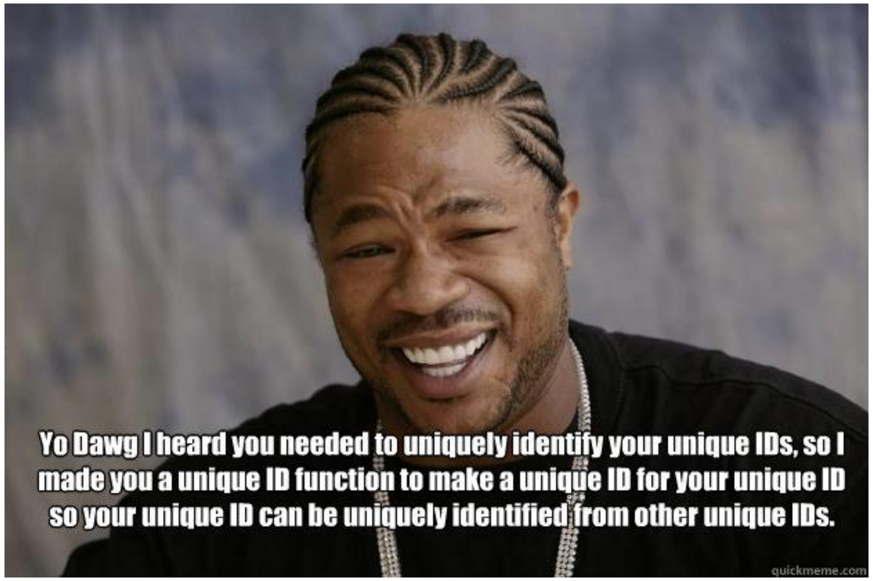
  awk -F‘,’ ‘{ print $5 }’ FILENAME.csv | sort | uniq –c | sort -nr

  - Print duplicate lines from a file (one for each duplicated line):

  sort FILENAME | uniq -d

  - Print all unique lines from a file:

  sort FILENAME | uniq -u


Yo Dawg I heard you needed to uniquely identify your unique IDs, so I made you a unique ID function to make a unique ID for your unique ID so your unique ID can be uniquely identified from other unique IDs.
quickmeme.com

About Cyber Solutions

Aon's Cyber Solutions offers holistic cyber risk management solutions, unsurpassed investigative skills, and proprietary technologies to help clients uncover and quantify cyber risks, protect critical assets, and recover from cyber incidents.

About Aon

Aon plc (NYSE:AON) is a leading global professional services firm providing a broad range of risk, retirement and health solutions. Our 50,000 colleagues in 120 countries empower results for clients by using proprietary data and analytics to deliver insights that reduce volatility and improve performance.

aon.com/cyber-solutions

Empower Results®