Automating Security Analysis

Andrew Ellis – GSEC, GCIA

December 27, 2016

Table of Contents

1. Automating Security Analysis

Automation is becoming a ubiquitous component of modern security analysis[a]. Acting as an extension of the analyst, automation increases the overall potential for review and therefore the benefits of automation cannot be ignored. The size of security-related data sets are constantly growing, while incident mitigation time-lines steady fall – leaving little room for conducting analysis at the "speed of humans." Automation and custom-developed tools are a critical part of addressing these time-line concerns as well as the other issues that arise from large-scale analysis. Producing and supporting these new tools, processes or capabilities introduces additional complexity and potential for error that must also be considered for automation to be considered successful. With benefits, concerns and mitigations in mind security teams can identify new and valuable places to focus research and development on automated capabilities.

The need for and benefits of automation are often the most clear reason to implement automation in an analysis workflow. While the end results of automation may produce an unlimited number of possibilities, the general benefits typically fall into the categories of force multiplication, quality and consistency, and knowledge sharing. For each benefit, multiple pitfalls also exist that must be considered when implementing new automation. Many of these problems affect multiple benefits and must be given additional attention to ensure new capabilities don't introduce more problems than they solve. Due to the complexity of these situations, no discussion of the benefit-concern relationships around automation could ever be complete; however, by examining the types of concerns and solutions around common motivations for automation, researchers can more effectively produce new tools.

A culture of research and development must exist for automation to be truly integrated into analysis. This begins by having processes that takes into account the agile and ever changing development around analyst-driven automation. A clear path for taking a concept to an implementation needs to be laid out and supported by both analysts and developers. Beyond this all members of the team, including non-developers, need to be aware of what qualifies as automation and how to describe the situations which merit further research. A tight feedback loop between the developers and analysts is required to ensure that effort is directed at the right problems or enhancements and not wasted on distractions.

Looking at benefits alongside concerns helps provide a clear view of what is needed to successfully integrate automation into an analysis workflows. These benefits do not come without an investment effort and analysis automation is not a silver bullet for security problems. However, when combined with other components of good security and analysis, automation can enhance the overall abilities of a team. Through an understanding of the benefits and concerns surrounding automation, security teams can provide higher quality and more valuable analysis.

2. Benefits of Automation

2.1 Force multiplication

2.1.1 Benefits

A core motivation for automation is its ability to force multiply the efforts of any particular analyst or team of analysts such that larger workloads can be handled with similar levels of analyst effort. Benefits of force multiplication fall into two similar, yet distinct, categories. The first and most direct way analysts benefit from automation is by increasing the ability for analysts to handle more overall data by reducing workload. This type of force multiplication is usually needed when trying to increase the amount of review analysts are capable of. Analysis also benefits from the ability of an analyst to review sets of data previously impossible to review, due to size. Analysts have an upper limit on the amount of data they can mentally keep track of at once[b], and tools that enable analysis against larger sets of data can also provide new analysis capabilities.

2.1.1.1 Increasing scale

As organizations become increasingly complex, the number of discrete data sources available to analysts for review has risen[cd]. Traditional security logging devices, such as intrusion detection sensors or web proxies, are now frequently paired with network flow collection or full packet capture platforms, host based detection and introspection, and a wide variety of intelligence sources. When combined with operational logging, such as Windows Event or authentication logs, analysts have access to a staggering amount of information about the activities within an environment. Despite very talented individuals reviewing this information, analysis takes time and large data sets may be infeasible to fully process in time to be useful.

Automation allows for analyst time to be spent more effectively by either doing mundane and easily repeatable work for the analyst, saving effort, or by actually making decisions about the data based on programmed logic, eliminating the need for human review. In practice, a large amount of an analyst's 'review time' is spent making queries of the data, waiting for results, and organizing and comparing the output. Many of these components are common across all types of analysis, such as the desire to correlate the current output with known indicator lists, and automation designed to present this type of information can have a large impact on analysis. An analyst may only spend 5 minutes collecting relevant IP addresses and reviewing intelligence surrounding them, but over the course of days and weeks this adds up to a significant amount of effort.

Machine time is cheap[e] and easy to scale relative to analyst time. By providing the analyst correlation information up front, analysis time can be saved and lost focus from context switching can be avoided[f]. Over time, specific enhancements may grow and be refined to very high levels of detail and accuracy. These trusted tools can then be used for particular types of triage, whitelisting or analysis and allow tasks to be completely offloaded from analysts to automation.

2.1.1.2 New capabilities

In addition to allowing more overall data to be reviewed, the ability to process a large number of items in a standard way allows for review that an analyst could never reliably achieve. The size of most data sets prohibits complete review by a human analyst making detecting patterns, filtering records and identifying specific values are often tasks best left to automation. The flexibility of custom automation has limitless possibilities for analysis and can be used to generate leads for analysts, narrowing the scope of review on large data sets. These new capabilities are an extension of the scalability of automation, allowing larger amounts of review to be conducted; however, it is important to distinguish between simply allowing more analysis to happen and tools which allow new types of analysis to happen.

Platforms which present APIs allow for analysts with development backgrounds to create personalized tools on the fly. This automation acts as an extension of the "analyst-developer" and allows for additional analysis capabilities even before a formal tool exists. Alongside the ability to scale out analysis in larger data sets, analyst-developers are able to conduct more thorough and in-depth analysis. Using software as 'glue', they can combine and organize information from different data sets, historical information or even to offload technical analysis for mass processing. This scalability and flexibility allows an analyst to effectively play with the data in many different ways depending on the situation.

Custom automation developed in this way and used to identify specific security threats can be significantly more effective than traditional automation, such as intrusion detection systems or indicator or compromise look-ups. Signature based detection is good at identifying known threats in large, high speed data sets, but may be unable to identify even slight variations. Customized automation can target known and unknown threats by using a variety of methods not limited to signatures.

General behaviors, like exfiltration or command-and-control beaconing, do not necessarily need specific samples to identify common patterns. Tools written to processes network or host data can detect these behaviors and provide consolidated lists for analysts to review. Malware designed to avoid detection frequently uses techniques that are difficult for static rules or indicators to identify and may require more advanced methods. Automation targeted at these specific tactics can be more accurate and flexible than rules or indicators, allowing for a wider variety of threats to be detected and reviewed.

2.1.2 Considerations

The multiplicative benefits of automation are well documented[g] and are one of the strongest reasons to focus resources on developing automation. These benefits often overshadow the complexities and problems which arise from automating analysis. Many of these issues come from situations where more automation actually increases workloads or decreases analysis accomplished[h]. Beyond this, even when successful, automation has the potential to distort the value of analysts and create more problems than it solves.

While the goal of automation may be to decrease the analyst work required for a potential task, when incorrectly implemented the task may actually require more work an analyst. Poorly tuned automation can create a flood of findings, requiring an analyst to review significantly more data than by the previous manual process. The larges lists of alerts may be considered "comprehensive," but when the majority of findings are false positives, analysts' time may be better spent elsewhere. Similarly, even with smaller result sets, confusing or unclear output may take analysts more effort to review than the raw data they are used to. To avoid misuse of analyst resources, metrics around the effectiveness and efficiency of analysis should be tracked to identify situations where automation is acting as a "force divider." Automation which reduces overall analysis capabilities needs to be removed from production and re-evaluated until it can be proven to help analysts.

Automation takes effort to develop and long term operational attention to maintain. Teams consisting purely of analysts may find resources lacking to produce the desired tools. Even when developer resources exist, diverting effort from analysis to development reduces the overall capacity for analysis. Time spent researching, creating, testing and maintaining new pieces of automation is time that could have been spent otherwise conducting analysis. The possible impact, both short and long term, of creating and using automation need to be be balanced against actual analytical needs. Automated tasks which require less development, but still reduce analysis time should be prioritized when development time is scarce.

Creating usable automation may also be restrained by platforms outside the control of the analysis team. Analysts and developers often require specific configurations, user accounts or other automation-related changes be made to underling data sources to allow for tool development. Teams in charge of these platforms are typically required to provide high levels of up time and may not be opened to untested modifications needed for designing truly integrated tools. Effort spent designing tools created to work with platforms outside the analysis team's control needs to be weighed against the potential technical and political issues around joint-team or multi-network collaboration. When challenges do exist, flexibility to adjust focus to other aspects of the analysis workflow may produce similar benefits in different ways, but without the added complexities.

As more analysis is moved to automation, a negative perception of analysts can also arise. Automated analysis may make analysts efforts seem less valuable and more scalable than is sometimes true. This may create a false empowerment of analysts with unrealistic abilities, or diminish their necessity as part of successful review. Furthermore, the "always on" model of some automation can guide non-operational teams into situations where monitoring and alerting is forced upon them. Managers and team leads need to be well informed on what automation can and cannot do, and what it takes in terms of time and effort to create solutions for new problems. Clear communication of capabilities and specifics about what will be done are required to ensure scope creep does not extend beyond a team's actual capacity.

Over time, the security concerns identified by analysis teams will likely be remediated, causing findings produced by automated reports to decrease. Eventually empty reports presented to customers may require justification for the development and analysis time invested to create them. Using exclusive language to define what was not found, previously empty reports can be presented as listings of threats not identified. Allowing analysts to offload basic checks for known threats, only requiring analyst interaction if findings are present, gives analysts more time for other types of analysis. With common threats eliminated, analysts can handle other alerts, conduct manual review of data, or research new automation without sacrificing baseline checks – all of which may produce previously unreported findings.

2.1.3 Summary

Automation can act as a force multiplier for analysis by:

- Reducing the work required by analysts to review data
- Increasing the overall amount of data which can be processed by a single analyst
- Allowing analysts to review data sets larger than they can mentally keep track of
- Providing greater flexibility to work with and correlate different sets of data
- Identifying non-signature based and unknown threats more accurately than static rules

When using automated solutions to increase the scale of analysis, the following should be considered:

- ✗ More automation is not always more efficient and may require additional review time by analysts
  - ✔ Metrics around effectiveness and efficiency of analysts should be tracked to identify automation which does not increase analysis capabilities
- ✗ Lack of dedicated developers or control over platforms may reduce the capacity for developing new automations
  - ✔ Be flexible and identify capabilities which are less complex or require less development, but still provide analytical benefits. Automation does not need to be difficult to be effective
- ✗ Analysts' abilities may become undervalued or overstated
  - ✔ Analysts need to clearly communicate capabilities and problems so that expectations can be managed appropriately
- ✗ Empty reports (due to effective identification and remediation of concerns) may need to be justified
  - ✔ Reports of what was not found provide validation that review was conducted and communicate what security concerns can be eliminated
  - ✔ Use time allocated for reviewing automated findings to conduct proactive review or identify new opportunities for automation

Ultimately, tools and automation have great potential to reduce workloads and allow teams of analysts to handle larger and more complicated sets of data. This assistance helps analysts quickly triage incoming data and focuses analysis efforts on more relevant events. Automation also has the potential to increase the amount of analysis, both in terms of volume and capabilities, than each analyst can achieve. This increase in the scale of analysis and output can lead to more findings and better analysis, but special care must be taken to ensure it is accurate and repeatable.

2.2 Quality and consistency

2.2.1 Benefits

The need to deliver regular reports to dedicated customers, often by different analysts at different times, shines a light on the need to let automation guide analysts. For example, a prior incident response that included an indicator and intelligence review by one analyst, sets the precedent that all analysts in future engagements will deliver similar results. Likewise, regularly delivered reports that lack consistent formatting make findings hard for recipients to correlate and follow-up on. Compounded with these operational problems is the potential for analyst-introduced mistakes. Common oversights, typographical errors, or misunderstandings often produce incorrect findings which withstand basic scrutiny, especially when review is done by a team member unfamiliar with the specifics of the case.

There are many solutions to these problems, with automation being only one. Other options generally fall under operational or analyst-driven mechanics; however as a result also require additional analyst time. The solutions, such as documentation and standard operating procedures, still ultimately rely on human execution. Each step that is reliant upon analyst efforts introduces a new point of failure or potential for variation. Furthermore, documentation is often forgotten by the original author and may fail to be updated. Even when updates are applied, analysts already familiar with the process may not realize a minor, yet important change has occurred and overlook it. In many forensic cases, high levels of quality and consistency are required for analysis to withstand legal scrutiny – and tools which can be submitted for review provide a solid definition of what analysis was done and how it was accomplished.

In similar fashion to how a calculator removes sloppy math mistakes, automation can guarantee parts of the analytical process are void of human errors. These errors can be addressed in the raw analysis stages, such as being sure that every item in a list is looked up in a consistent manner, or in review stages by routinely checking for common malicious behaviors. Typographical mistakes can also be reduced by generating output that is easy for the analyst to directly include in reporting. Tools and automation that are part of the analyst's workflow can ensure common processes are followed by different analysts and that output produced follows a standardized format.

Automated reporting which provides “to-do” items further guides manual analysis by informing analysts of actions they need to take. These action items can serve as a checklist to ward off the potential for skipped analysis steps. Unlike documentation or training, updated tools immediately impact analysts and make obvious changes to the process.

2.2.2 Considerations

The types of bugs which impact automation are often very difficult to test against in development. Logical issues in sections of automation designed to do in-depth analysis may be impossible to test for ahead of time, requiring analysts to act as bug testers for new tools using real data. Even when automated output is error free, it can give the impression that all possible checks for a given behavior were done, when in actuality follow-up analysis may still be required. Additionally, analysis driven by guidance provided from tools can be too encompassing or strict and can cause analysts to review data with blinders limiting their view. Care needs to be taken to ensure that automation does not "scale out" errors, as well as analysis.

In many cases errors may only be discovered when tools are used against new data sets, making testing and quality assurance difficult. When errors do occur, logical errors may be tedious or impossible to identify by analysts during production use, depending on what output is provided. Concise views help analysts to process large amounts of information, but may not provide enough detail to allow for thorough vetting of automated results. This results in a situation where both automation and analyst can make mistakes which lead to large scale analysis failures.

A 'realistic' and repeatable testing environment is required for basic quality assurance and error proofing. Logical analysis errors may be hard to test against, but common programming concerns should be identified in development testing. As many errors may still be first identified by analysts in production, developers need to ensure tools provide as much relevant information as possible without cluttering results.

When test data does not exist and development must be done against live data, steps need to be taken to handle the eventuality of automation failures. Error reporting needs to be descriptive and easily communicated back to developers for remediation. Tool output needs to include enough information to validate findings and analysts need to be instructed to trust, but verify automated output. When issues are found, a clear escalation path needs to exist between analysts and developers to allow for questions and concerns to be addressed without friction. Without access to developers, answers to analysts' questions or requested code updates may come long after the analysis need has past. Above all, developers creating and analysts using custom tools need to understand that accountability for the correctness of results rests on their shoulders, not on the software.

Automated reporting needs to have clear definitions of what analysis techniques are included, both to manage customer expectations and to reduce the potential of being blamed for missed findings. Automated analysis is rarely comprehensive and checks for common behaviors leave room for outliers to avoid detection. Assuming reporting represents complete views of the data, analysts and customers alike may discount follow-up review and otherwise obvious security concerns may be overlooked. Inclusive definitions that clearly identify the types of analysis contained in reporting can make limitations of tools more obvious and help avoid dangerous assumptions.

Outside of well defined, operational settings automated analysis needs to be paired with free form review to mitigate the possibility of missing things due to "automated tunnel vision." No amount of automation can currently replace the intuition and ability of a skilled analyst to identify anomalies in data. In many cases, unique situations don't allow for tools to identify unknown-unknowns required to detect a threat. Analysts need to be aware of potential shortcomings in tools and ensure they are always guiding their analysis, rather than simply following the automated bread crumbs left by tools.

2.2.3 Summary

Automation can increase the quality and consistency of analysis by:

- Standardizing review and ensuring common tasks are repeated consistently across different analysts
- Reducing the potential for human mistakes and typos
- Guiding analysts through specific processes and limiting oversight
- Impacting analysis immediately without requiring re-training or updating documentation
- Producing reviewable and definable analysis processes

When using automated solutions to increase the quality and consistency of analysis, the following should be considered:

- ✗ Automation has the potential to "scale out" errors as well as analysis, thereby mass producing failure
  - ✔ Verbose error reporting and detailed output allows analysts the ability to detect and identify issues
- ✗ Logical errors may be hard to test for prior to production use
  - ✔ Provide analysts with a clear escalation path to developers as well as training on how to identify and report problems
- ✗ Reports produced by automation may misrepresent findings, especially in the case of reports with no results
  - ✔ Automation should define specifically what it looks for, and therefor exclude what it does not, clearly identifying what type of review was conducted and what is left for analysts
- ✗ Using automation to guide analysts may reduce creative review done by analysts
  - ✔ Automation should be paired with analyst-driven, proactive review to identify unknown threats

Automation's ability to ensure quality and consistency can allow analysis teams to handle larger workloads on even more disparate sets of data with confidence. Initially this may provide a flexible and dynamic team, but in the long term can result in certain analysts always being required for specific tasks leading to workload imbalances. Using automation, these different sets of skills and knowledge can also scaled up to allow for increased "share-ability" of tactics and techniques.

2.3 Knowledge sharing and re-use

2.3.1 Benefits

Analyst knowledge is the corner stone of conducting proper security review. With the growing array of problems presented to analysts, the amount of knowledge and number of unique skill sets needed to effectively handle incidents is staggering. Teams may include subject matter experts across a variety of domains as well as generalists, capable of quickly learning new topics. The combination allows teams to operate flexibly by simply selecting the right analyst for the task. Depending on workloads, analysts may find themselves needed for specific tasks more frequently than they can support or being asked to do work long since forgotten. Packaging this knowledge into automation lets analysis teams balance and retain knowledge across a wider group of analysts. This use and re-use of analysis techniques can also provide an opportunity to enhance existing automation and processes.

Both specialists and generalists not only have skills that are non-transferable, without large training investments, but also end up with much "point in time" knowledge. With some of this knowledge requiring depth of experience, often in arcane subject matter, training teammates may not be a timely or realistic option. Some situations may further require detailed study by a specialist, in which they dive into a particular topic for a brief time. Much like cramming for a test, this knowledge is generally temporary – even for an expert. Skills learned and then forgotten represent lost productivity for teams and automation can be used as a way to reclaim this effort.

Knowledge sharing already occurs in the form of documentation and training for specific tasks, which allows multiple analysts to complete the same process in a standardized way. Instructions can be written for how to successfully complete analysis; however, ambiguous language and lack of domain specific knowledge still limits what tasks can be reliably taught at a large scale. Training can bridge the gap left by documentation, but also requires an additional effort on the part of multiple analysts to be successful. As with the issues surrounding documentation for quality and consistency purposes, documentation repositories must be constantly updated and referenced to be effective and training may need to be regularly re-conducted.

Automation allows for analysts to package up this personal or time-sensitive knowledge to be stored and re-used later and by other analysts. The use of a programming language rather than a spoken language to define the analytical process removes much of the idiomatic or intrinsic ambiguity. Converting analysis techniques that are difficult to explain into automation decreases the learning curve for other analysts. Using tools reduces the knowledge required to operating the tool and interpreting its output, rather than encompassing all stages of the discrete analysis. All analysts benefit from shared tools, but tool authors in particular have a unique opportunity for continued refinement and enhancement. Re-use of the tool across different data sets provides many use cases for quality testing and enhancement, resulting in robust and well tested tools.

2.3.2 Considerations

Analysts leveraging large collections of tools may find that the tools themselves become a skill set they must learn. Documentation or training can be provided, but takes time to create and for analysts to consume. Both large tool sets and complex tools provide analysts many options when conducting analysis, but may be difficult to use effectively without domain specific knowledge. Tools which require additional learning to use may lead to the analyst simply doing the work by hand, rather than fighting with an unknown tool.

Few analysis situations are identical and as such many tools provide configuration options to help analysts adapt them to the current need. For simple or regularly used tools these options may be straight forward and easy to understand; however, more complicated tools, offering many options, create the potential for misuse. Poorly described configuration options leave analysts to interpret their meaning, often incorrectly, which may impact the accuracy of results. Analysts seeking to configure tools correctly may even find themselves spending time researching the exact task the tool was meant to automate. In situations where repeated analysis is being done, variations in these configuration options may produce output which, while correct, is not consistent with previous reports. To avoid confusion over using tools, clear documentation about the use and impact of configuration options should be provided to analysts. Beyond this, tool output which reports run-time configuration settings allow for easy comparisons to ensure the correct options were used.

Similarly, extremely detailed output can make it difficult for analysts to identify important results. Verbose output, often designed by and for subject matter experts, can make it hard for other analysts to interpret findings. The inclusion of raw data provides context to analysts but cluttered views may obscure visibility into the intended results. This extra detail should be optional such that analysts can view only the most relevant information and expand into more detailed data. Tool designers and analysts often have a fundamentally different view of what output should be which makes designing output difficult. Therefore feedback from analysts is essential producing the right balance between enough detail and too much detail.

Contextual assumptions made about tool output can have the same effect of obscuring findings, but for opposite reasons. Analysts using tools as a means of sharing knowledge may not have a background in the underlying analysis being automated. Without a frame of reference, output which is provided without context may be left open to incorrect interpretation. Additionally, unexpected results which would be obvious to the tool's creator, may not be to an analyst using the tool for the first time. Preventing these issues requires that output from tools be well defined and unambiguous. A clear summary of data, supported by raw or contextual data, helps ensure that analysts focus on the intended findings. For more advanced topics, documentation or training may be required along with proper output to ensure that knowledge, not just information, is shared.

Analysts need to be given freedom to develop the tools they need, but some normalization needs to endure throughout to make tools universally usable. Where possible, using similar interfaces as well as common input and output formats can reduce the impact of many of the issues involved in knowledge sharing. Tools which don't require the analyst to spent effort understanding their use, allow for analysts to focus their full attention on analysis. These familiar tools, when paired with basic documentation and tool-based help menus, allow analysts to potentially solve their own problems without escalating the issue to a developer. Easy to use tools that are quick to learn are more likely to used by analysts.

Large analytical tasks do not easily or quickly lend themselves to functional solutions, much less easy to use tools. When considering which analysis tasks to automate, focusing on small, well defined capabilities makes creating documentation and interfaces less complex. Allowing interoperability between tools, lets analysis and developers tie discrete functions together into larger capabilities based on known and documented components. This reduces time creating similar functionality in multiple tools, by producing one tool and allowing all other tools to 'pipe' data to it for processing. Over time, this growth of this collection of compatible tools gives analysts flexibility to conduct different types of review and can be used to identify larger development needs for consolidated tools. Breaking tasks down into smaller, less encompassing pieces which can be re-used further minimizes the time analysts spend learning new tools.

2.3.3 Summary

Automation can increase knowledge sharing between analysts by:

- Providing a mechanism to "package" knowledge to be used by other analysts
- Maintaining point-in-time knowledge while not in use
- Defining processes in less ambiguous terms than documentation
- Reducing the time required to learning a new discipline to that of learning a new tool
- Refining tools and processes through re-use by different analysts

When using automated solutions to share analyst knowledge, the following should be considered::

- ✗ Large tool sets may become difficult for analysts to use and eventually discourage use
  - ✔ Designing tools with similar interfaces reduces training time for analysts to learn the same thing in multiple tools and makes large tool sets easier to use
- ✗ Complex tools may require more effort to learn (or remember)
  - ✔ Basic documentation can enable analysts to quickly learn (or refresh) how to use a tool
- ✗ Output designed for subject matter experts may be difficult for other analysts to use which may create inconsistencies and produce errors in analysis
  - ✔ Clearly identify potential findings and provide appropriate context to avoid confusion
  - ✔ Tools need to producing only relevant, well defined output, which requires analyst feedback
- ✗ Problems may require complex solutions with many components
  - ✔ Automation does not need to solve whole problems at once, focus on smaller, inter-operable tools which can be used together

Along with the force multiplication, quality and consistency impacts of automation, the ability to share knowledge through tools becomes a base for building a highly scalable, flexible and robust analysis team. A tight connection between analysis and development is needed to allow for quick and effective tools to be created. Analysis teams without dedicated development groups may be left with the need for a lower overhead, more reactive development methodology that is more compatible with analysis workflows.

3. Researching and Developing Automation

Creating new tools and automation requires a significant investment of time, which may distract from analysis team's core responsibility – delivering analysis. Therefore, research and development time needs to not only create effective tools, but also be able to produce usable results as quickly as possible. Lacking full-time developers or time dedicated to research means that analysis teams need to find ways to tie these tasks directly into existing analysis. By letting the natural analysis workflow guide research and development, some of the overhead involved in creating automation can be minimized.

The integration of research and development with analysis is essential to creating effective custom automation for analysis teams. This connection between these processes allow analysts to make progress on new tools, without diverting large amounts of time away from required analysis. Requests for and responses to new tools form a self-generating road map of functional needs and operational problems, which requires very little additional oversight to clearly determine development goals. While reducing overhead creates the opportunity for analysts to spend more time exploring new options, it is not the only motivation for tying research and development into analysis.

Unlike during development where test data can be very difficult to find or create, active investigations allow for realistic testing of new tools. Failed tests allow developers insight into bugs in new tools, and successful tests provide analysts with potential leads. Analyst feedback on these leads and other aspects of using the new tool also provide developers with specifics to further refine the tool for general use.

Research and analysis share common techniques that make this integration a natural extension of the analyst's workflow. The guess-check-revise process of looking for suspicious events and the constant iteration of tool design are at times indistinguishable from each other. This common method of repetition and improvement can then be used to bridge the gap between analysis and development, allowing for truly integrated workflows. Each time an analyst improves a process, suggests an update or identifies a bug based on analysis – tools become more reliable. Every small, targeted change developers deliver quickly directly impacts analysts and can reduce analysis time or produce additional results. The feedback loop between analysis and development combined with regular updates eventually leads to collections of changes that form into tools or new capabilities.

Analysts conducting review on a daily basis can easily put their finger on problems or missing functionality. Leveraging this situational awareness to guide research increases the chances of newly created automation solving real problems. Long running projects, isolated from analyst feedback and real world testing can lead to tools that, while complete and successful, do not fit current analysis needs. Requests and feedback from analysts provide the checks and balances for keeping new research projects in line with actual needs. With experimental ideas being quickly delivered to analysts and used in real situations, problems and shortcomings can be identified at earlier stages in development. This helps small issues be revealed and revised more quickly, and clarifies larger issues which may prohibit overall project success.

A multitude of proven development methodologies already exist covering the full software development life cycle spectrum[i]. These structured methods produce robust and reliable results and are critical to large scale, customer-facing, or production platforms. The detailed tracking and planning involved with projects deliver high quality results; however, may be too rigid or sluggish for the dynamic development needs of analysis teams. Despite the need for a more lightweight process, teams still need to ensure that representative goals drive development and active analysis is an excellent source of this guidance. Tools that begin in analysis may eventually grow into full platforms which require more traditional management. As such, allowing an iterative, integrated research and development methodology to co-exist with traditional SDLCs can create a flexible structure for exploratory development while still providing a long term development path for successful tools.

3.1 Applied research and development

Analyst and developer time is a scarce resource and therefore a dedicated effort must be made to ensure that hours spent in development are hours reduced later in analysis, albeit sometimes much later. Providing direction on what areas to focus research on can be a nebulous process with many different development paths available. Determining the most effective course of action can be difficult due to ever changing environments and countless variables which make direct comparisons between projects impossible. Time and resource limitations prohibit following all potential research leads, making some form of guidance necessary to identify more relevant or useful leads. In addition to these operational issues, the fact that research is by definition experimental means that the results of time invested may be the discovery that a particular tool or technique is not viable.

Analysts already spend time researching unique analysis techniques when answering new questions or dealing with unknown data sets. Tying research and development into these analysis processes can be a way to provide added insight to what capabilities are most needed without requiring large investments of dedicated scoping time. Custom code and new analytic techniques, discovered by interacting with the real data, are ways analysts can identify functional holes in existing tools or platforms in response to actual problems. These scripts, enhancement requests and pieces of documentation form an organic road map to specifically define functionality missing from current tool kits. Large groupings of similar or related requests provide prioritization and clearly identify which lacking functions are most regularly needed. Code and documentation created as part of analysis can be used as functional examples and direction for developers further reducing time spent designing new tools for analysts.

The exploratory nature of research and development means that even when a project is completed, it may no longer be relevant to current analysis needs or may not produce useful results. Using active investigations to drive this type of research creates a feedback loop between actual needs and proactive development. As requests come in for bug fixes, feature requests or new tools the developers gain an understanding for what types of capabilities are lacking. Absence of analyst requests for enhancements or updates of existing capabilities may indicate that current tools cover all existing needs or that analysis related to those tools happens less frequently. Using request queues as guidance, developers can make informed decisions about the prioritization of incoming tasks and focus limited resources on more necessary objectives. Conversely, projects unrelated to any active requests need to be treated with caution as, while the end result may be useful, research and development may be taking time away from addressing a more urgent need. Using real analysis to drive research and development is not a replacement for oversight, but instead can be one of the tools analysts and developers can both use to collect feedback and define goals.

Analysts conducting daily analysis can more readily identify tasks that take unnecessary time or are difficult to review than developers operating in a vacuum, expected to predict the future. Building on these actual use cases allows developers to produce tool sets for common tasks by guiding development to solve problems that actually occur rather than those which might occur. Ingraining the idea that problems like these are not simply to be endured by analysts, but when given proper attention can be solved, not only allows for more flexible analysis but also improves the analyst's quality of life.

Developers are often at a premium within analysis teams and it's unrealistic to expect every analyst to be able to design and build new tools from scratch. A much more realistic goal is to enable analysts to communicate effectively with developers and to provide detailed requests, problem definitions and output examples. This type of information allows developers to quickly identify common needs and focus design efforts on solving these specific analysis problems.

While not all analysts need to be developers, all developers need to be involved in active analysis to avoid isolation from the actual review process. Using the tools as part of on-going investigations, developers can see first hand problems and missing functionality which may not be obvious to front-line analysts. Perspective gained from being part of review also allows developers to have more efficient conversations with other analysts about development requests. This tie between analyst and developer is critical to keeping tool development relevant and research focused.

In order for automation to be truly effective, a universal view of research and development must be bred across the entire analysis team. Analysts should be encouraged to try new things and be part of the research process by providing feedback or ideas to developers. Successful techniques, sample code and test data can then be shared with other analysts to assist with analysis and to gather feedback on new capabilities. Quick and easy communication between analysis and developers which reduces friction when sharing these methods continues to encourage collaboration between the two groups.

This rapid and continuous pattern of scoping, research, design and development does not come without its own concerns. Analysis teams often don't have the luxury of taking a break to proactive work on tools, and as such research and development are done in the margins. The largely decentralized nature of this type of development has the potential to produce silos of development and complicate planning. Many analysis teams do not employ full-time developers, further impacting potential research and development. The combination of these factors results in a situation where traditional development methodologies may end up being too overhead intensive or slow to react to be effective means for producing new tools and automation.

3.2 Highly iterative design

Applying research and development to active analysis can produce extremely targeted and high quality tools; however, the speed at which analysts must operate means actually accomplishing these lofty goals can be difficult. Therefore a focus must be put on reducing the overhead associated with the creation of new tools. Finding as many places to tie research, development and analysis together as possible is one way of creating more efficient operations. As a part of this process, tools that are incomplete or untested may find their way into regular, but supervised use for testing and refinement. Each time the new tool is used can also double as scoping, research, development, or testing as well as analysis. This process results in a highly iterative methodology for building and maintaining tools. Leveraging time already spent working with tools creates a situation where software can evolve through regular use without large amounts of added overhead.

Tools built exclusively for analyst use have fewer requirements than customer-facing platforms and often rely on analysts to manually 'insert' missing functionality in unfinished tools. These prototype tools can produce raw and experimental output and by relying on the analyst to interpret, validate and polish the results can still produce useful findings. This allows for new ideas to be validated before large investments of time are spent refining all facets of a new tool. Bugs are an expected part of this process and as such an escalation path directly to developers needs to be provided. Priority needs to be given to analysis and production issues, especially when impacting or prohibiting analysis, need to be resolved quickly so that analysis can continue. Analysts using the experimental tools can act as a quality assurance and testing team as verifying new capabilities is a result of doing regular analysis. Ultimately developers need to be accountable for a tool's accuracy and results, but analyst use against real data provides an additional low-overhead method of testing reliability.

Ideas generated from active analysis provide developers not only with concepts to test and methods to apply to data, but also help to scope new tools and act as an example of desired output. This allows developers to build tools specific to needs defined by analysts with very little ambiguity attached to the research and definition process. As new tools and capabilities are created, the developer can then use the new tool to provide the types of analysis requested back to the original analyst.

Exposed by means of "software as a service" principles, other analysts can leverage the new capabilities by requesting the developer to run the tool on their data. This allows for the developer to maintain oversight on an untested tool, but still lets it be used in active investigations. Each case where the tool is used provides more opportunity for real world testing while still producing usable analysis results. Brief explanations of the situations a tool can be used in can be provided on team calls and easily spread awareness of new capabilities. In addition to refining the tool, multiple cycles of requests and responses further generate awareness among other analysts of when the a tool may be applicable and what its capabilities are.

Constant iterations on each tool lets developers make small changes frequently, breaking problems down into manageable sizes. Building an entire platform in the margins of analysis is a slow and difficult task, with long time lines leaving analysts without tools during the development process. Projects instead may begin as smaller updates to existing tools or as non-software solutions for much of their early development. These precursors to full automation, shared and exposed in the same way as more complete tools, can also be tested and refined through multiple cycles of use. Over time, theses processes become more robust and can converted into automated solutions.

Ultimately, tools may out-grow this iterative process and find themselves in a more rigid and platform-based model. Large collections of specific solutions may drive overall development towards frameworks that require more overhead and design that a highly iterative approach can provide. Regular use of tools also requires increasing amounts of operational support, which can reduce or completely inhibit analyst-developers from creating new tools or conducting analysis. While projects may need to transition to development operations teams, testing and quality assurance can still be integrated directly into analysis to provide real world data and relevant feedback. The challenge of identifying this transition is as much an organizational and business politics problem as it is a technical consideration.

4. Opportunities for Automation

At its core – automation is any place where a task done by a human can be replaced or augmented, even partially, by machine time. In some cases this requires custom tools to be created while in others existing tools can be used or slightly modified to increase the productivity of analysts. The concepts of automation and where it fits in to the overall analysis process needs to be a clear part of the way a team operates. Most processes can be augmented by automation, many in multiple distinct ways, making the choice of which opportunities to pursue a difficult one.

The ability to identify processes that can benefit from automation comes largely from experience and an understanding of existing tools, workflows and platforms. These factors along with other unique variables make listing all of these opportunities, even for a limited situation, an unrealistic goal. In order to better understand the different ways automation can be applied to an analyst's operations, specific scenarios can be used to highlight aspects of designing these enhancements. Using examples as a baseline, common components of designing all automation, such as iterative development or non-traditional definitions of 'automation', can be identified and applied to other related situations.

The automation created in each situation may be distinct, but the process by which they are designed stays generally consistent. Automation created in reaction to analysis needs is best built piece by piece, rather than as large, monolithic projects. This allows for changes to quickly reach analysts and for feedback to be provided at early stages of development. These smaller pieces may not produce the entire solution, but can still reduce workloads by addressing the most time consuming steps involved in analysis first. Solutions will vary between scenarios and require different implementations; however, a broad view of what qualifies as automation ties each option together with a common goal of bettering analysis. Whether changes are made to tools, platforms or workflows the measure of success is always connected to analyst output. This key metric allows for the results of drastically different projects to be compared in meaningful ways.

Using automation already available is just as important as creating new tools. Refining the way existing tools can often provide more benefit, for less investment, than developing new tools. Furthermore allowing custom tools to be post processed by analysts using existing tools can save programmers hours of development re-building features that already exist elsewhere. Unique tools usually come into existence when more complex filtering or correlation is required than currently is available.

Distinct to each situation are the problems and motivations driving the need for automation. While all three examples touch on multiple benefits provided by automation, each also shows a specific, critical need. This specific need, either for greater scale, reliability or knowledge transfer, guides research and development towards relevant tools. By tying research and development into existing analysis operations, attention to problems with tools and missing capabilities can be clearly focused. The constant iteration at each stage of development provides a built-in mechanism to allow for changes in needs to be identified quickly. Developers and management can then use this information easily redirect efforts and adjust road maps accordingly.

4.1 Managed security monitoring

One place where automation's role as a force multiplier can be seen most clearly is in the managed security industry or at large-scale security operations centers. Processing large amounts of information and quickly identifying malicious behaviors requires that at least some facets of review must be automated[j]. Analysis will always have a manual component, which is typically the largest bottleneck in the process, and when operating at extreme scales or with under-staffed analysis teams automation needs to focus on making analysts' time as efficient as possible.

Automation can take many forms, ranging from platforms that ingest raw events to tools designed to look for specific behaviors. Tedious and time consuming tasks like parsing, organizing and enriching large data sets are generally handled by dedicated platforms[k] before any analyst is ever presented the data. Built-in whitelisting or correlation in these tools can reduce the time required to prioritize events and instead allows analysts to spend time doing analysis. With mature and well maintained platforms in place, actual analysis can be converted into custom tools which handle baseline review steps and other common forms of simple analysis. At every level, tasks previously accomplished by analysts can be offloaded to automation resulting in an overall increase in analyst efficiency. This holistic view of automation allows for development flexibility when involved in large-scale analysis.

Platforms importing, correlating and visualizing data make up the majority of what automation does for analysts in large-scale environments. Therefor, these functions need to be effective for analysis, and not simply act as data warehouses. In many cases complicated dashboards, cluttered with useless information, impede in-depth analysis. Slow tools further force analysts with limited time to work with information at hand and often prohibit follow-up analysis that may result in additional findings.

When working with the volume of alerts that most managed security solutions have to handle[l], custom tools and automation may offer less benefit than changes to the core platforms consuming events. Just as software requires upkeep and support, rules and indicators feeding these tools also need regular attention beyond simply adding new entries. Large lists of indicators and overly comprehensive rule sets can result in analysts spending much of their time eliminating false positives. If analysis is to be effective, curating of existing rules and indicators is a required part of making automation valuable.

Automation can range from analysts leveraging common tools like ‘grep’ or Microsoft Excel to custom programs designed to process large collections of data in very specific ways. When handling large data sets analysts often turn to the tools at hand to perform reductions on large sets of data. Interfaces which provide query and filter capabilities can be used to remove or highlight important items for review and to bring more relevant information into view. Tools not designed and supported by analysis teams have lower overhead requirements related to their maintenance and leave more time for analysis. In many cases, existing tool suites are more than adequate for analysts daily needs, but for specific cases additional automation may be required.

From the perspective of an analyst, the force multiplicative effects of well-designed automation are felt in the form of a more enjoyable workflow and less overall repetition. Analysis platforms fed by curated rules and paired with robust whitelisting produce less false positives and let the analyst spend time on events that are valuable and interesting. Records that are automatically enhanced with information which would commonly be looked up, such as reputation data or geographical locations, saves analysts additional time moving between tools. Correlating list items with other activities, when supported by clean, functional interfaces, allows for a quick, yet detailed review of potential threats. Each piece of added or updated automation can result in more analysis, better analysis or both. These effects not only produce higher quality analysis, but also increase the quality of life of an analyst.

4.2 Incident response workflows

Preforming an incident response is one of the most dynamic[m] and least forgiving[n] situations an analysis team may have to deal with. Compromised environments can provide a wide variety of information to be reviewed in a very short time. Ensuring nothing is missed and that results are forensically viable requires that review be standardized across data sets and analysts. Standard operating procedures and proper documentation are a mainstay of this type of repeatable work, but leave the potential for human error at any number of points in the process. With high tensions and higher stakes, analysts need reliable tools supporting them. Introducing automation into an incident response workflow can help ensure a timely, repeatable method for reviewing data and may free analyst resources to pursue other investigative leads.

In many cases, the platforms used to review incident response data are not created by the incident response team. Instead, third party platforms and available data are typically incorporated into a general analysis workflow and 'glued' together with custom automation. These tools provide a consistent analysis baseline that can be built upon and updated as the incident progresses. These baselines might include common tasks like indicator look-ups, checks for dangerous misconfigurations or detection methods for specific, known threats. Each baseline can then be reliably re-used to produce consistent analysis. The comprehensiveness of these tools is less important than the definition of capabilities they provide. Even when empty, well-defined reports can be effective at ruling out known threats and steering analysis towards other aspects of the data. Each new tool created gives analysts another way to consistently review incident data and provides confidence in the quality of results.

Forensic investigations involving law enforcement put a premium on repeatable, documented correctness. Chain of custody, data handling and processing can all impact the usability of findings in court proceeding or other legal actions. An incident response team may find themselves acting as crime scene investigators, documenting data points that must be complete and accurate for later review. This process and documentation associated with data collection may be equally important to the accuracy of the results and a miss-step may result in legally useless data[o]. Interfaces and output from tools can be used to help guide investigators through the current process and help ensure consistency across analysts. Automated reports can guide even the most junior analysts through complicated processes by producing output in delivery-ready formats. Sections left empty or explicitly defined steps act as a “to-do list” of necessary actions and make obvious when required details are forgotten.

Using tools created for automated analysis not only mitigates the potential for human error, but also allows for objective analysis methods. Even when understandings of behaviors are consistent, the actual analysis that takes place from analyst to analyst and judgments they make can affect the quality of findings. Large sets data, such as failed authentication logs, may be broken up across multiple analysts to quickly identify suspicious behaviors. Looking for failed authentication attempts in logs is a straightforward process; however, when scaled across different analysts inconsistent results may make follow-up analysis ineffective. Tools defining factors like which failure types to consider and what minimum thresholds to use help remove variations introduced by distributed workloads.

When handling incident responses, findings need to be not only correct, but also presented consistently. Inconsistent results are very difficult to correlate, due to variations in reporting structure or analysis, which puts an emphasis on formalizing reporting processes. In many cases results are passed along to other analysts, law enforcement or external security teams who must then decipher the differences within the data before being able to proceed with follow-up actions. Automation and tools that guide analyst workflows help remove these analysis variations and produce common results every time. Tools can also be used to mitigate assumptions made of reporting data, as source code can be shared as a static definition of what analysis actually occurred.

4.3 Proactive review and research

Proactive review can be defined in a number of ways, but ultimately comes down to looking for unknown threats by reviewing data without rigid guidance. Investigations often turn to proactive review when no indicators or obvious signs of threats have been found, but when the need to identify maliciousness still exists. This proactive review may result in leads for new analysis capabilities which, as part of the analysis process, have already undergone some testing and improvement. Early stages of proactive review lack a defined process and instead follow intentionally exploratory analysis paths. Research paths which have a high potential for discarding the results, may not merit real development until they can be validated; however, once capabilities have been tested and verified, more formal research and development can be applied to refine them for general use.

Front-line analysts and developers have a unique view of their analysis environments and the types of tasks they typically cannot accomplish based on tool limitations. Vendors have tools, intelligence sets and other security capabilities are designed for every problem[p], but over generalization of these platforms leave gaps that must be filled by analysts. Long-term issues require analysts to creatively improve aspects of their workflow each time they encounter the problem. This process of multiple iterations, on small fixes, to regular problems becomes a method of manufacturing solutions.

New tools often begin as documented analysis processes derived from this proactive review. Notes analysts keep to allow for ease of repetition or to track new techniques are the first step in creating automation. Over many uses, these processes may be refined similarly to software and produce an end result that is well defined. These documented processes, however explicit, may still rely on intrinsic knowledge the original analyst forgot to include or cover processes which are too detailed to fully describe adequately. While the analysis method may continue to improve for the author, it still may not be readily sharable with other analysts. Leveraging this documentation as design schematics, developers can begin to convert manual processes into automated tools. Knowledge transfer from the source analyst may still be required to help the developer fully understand the process, but once transformed into a tool, additional analysts can quickly employ the tactic with no need for in-depth training.

Flexible tool kits derived from proactive review become powerful assets for analysts. With functionality and interfaces designed specifically for their needs, different analysts are able to conduct dynamic review based on intuition and leads. Tools that can adapt to these specific needs let analysts review data "their way." This allows for unique analysis from different analysts, even when using the same tool.

5. Conclusion

Automation is not the singular solution for solving all security problems; however, in the context of modern analysis it is quickly becoming a fundamental component of holistic review. A multitude of external factors affect the analysis types teams may be asked to conduct. The number of breaches[q] and size of budgets[r] also seem only to be on the rise – proving there more need than ever for analysis. While new players enter the industry every day and handle some of this increased workload, there is still a surplus of analysis required[s]. Ultimately analysis needs to be made more efficient, effective and accurate to cope with this growing need. Automation's ability to reliably scale analyst knowledge makes it perfect for the task.

Creating successful automation requires that a wide range of needs and concerns must first be identified and addressed. Many of these issues are difficult to predict or solve and direct feedback from analysis and analysts can act as a guide. A research and development methodology that quickly allows new ideas to be tested and improved upon can let teams safely explore new possibilities without fear of wasted resources. Relying on constant iterations and real world use to drive goals creates an environment where time is devoted primarily to relevant projects with minimal need for oversight.

With analysts and developers working together, meaningful research can occur and the feedback loop created prevents tools from growing stale. Constant attention and regular curating of automation and its supporting components is required to avoid problems introduced by this added complexity. When tools and automation are properly maintained and driven by analyst feedback, capabilities of teams conducting analysis can be increased.

Human analysis is most likely never going away, even as automation and tools are made more capable they still lack the fundamental creativity of a dedicated attacker[t]. Intuition is required for identifying unknown-unknowns and to ensure automated tasks remain accurate. The most critical aspect of automation is to remember that it is an extension of and not a replacement for the analyst.

Appendix A – Definitions

Analyst-developer – An analyst with a development background, allowed to leverage the skill-sets in combination for the benefit of providing better and more comprehensive analysis.

Application program interface / API - The set of available functionality exposed by a tool or platform for interfacing from external applications.

Automation - Any tool or platform that allows for a human task to be replaced or augmented, even partially, by machine time

Incident response / IR – An organized and planned approach to responding to successful malicious activity, with the goal being to mitigate damage and begin remediation.

Indicators (of compromise) / IOC(s) – Artifacts used to identify potential malicious activity, typically based on previously observed use. These artifacts tend to be signature based, such as IP addresses, MD5 hashes or domain names, but also can include behavioral descriptions of malicious activity.

Intrusion detection system / IDS – A platform that monitors a network or systems for malicious activity or policy violations, typically based on signatures.

Malware – "Malicious Software" or programs intended to damage, disrupt or otherwise manipulate computers

Proactive review / proactive analysis - Analysis based on looking for unknowns by reviewing data without the constraints of templated analysis, with the focus being on identifying new analysis paths.

Software as a Service/SaaS – A product delivery methodology where tools are present on-demand, rather than hosted or run by customers.

Security Analysis – Any sort of technical analysis on data relating to potential or actual security breaches. Typically focusing on logs and events collected from network appliances, host-based monitoring and other security specific tools such as web proxies or IDS

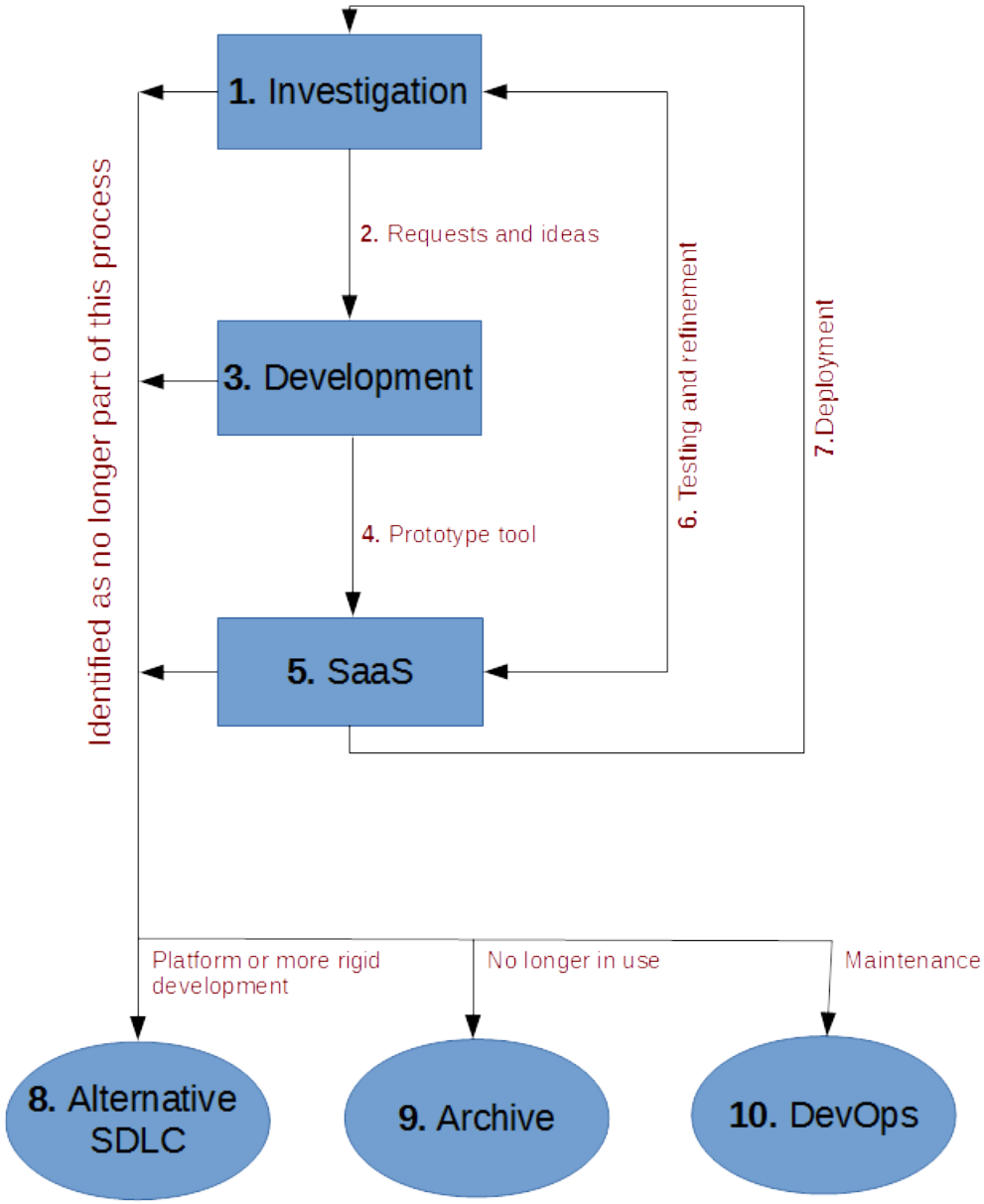
Security operations center / SOC – A centralized and often highly operational team dedicated to monitoring security events in an organizational and technical capacity. In many cases these teams are 24/7/365 and represent the front-line of human defense against security threats.

Security incident and event manager / SIEM – Tools designed to collect and combine information from a variety of security related sources, such as logs or events.

Software (Systems) development life cycle / SDLC – The process of planning for, creating, developing and testing a tool or platform.

Appendix B – Highly Iterative Design Workflow Diagram

Provided below is a workflow diagram detailing the process of integrating analysis with research and development.



1. Investigation – The process where tools, tactics and procedures are used during analysis. This can be ad-hoc, monitoring or proactive analysis.

2. Requests and ideas - Bug reports, feature requests, enhancements and ideas generated from using (or desiring) tools during analysis

3. Development – Scoping, design, development and initial testing of new tools

4. Prototype tool – A new tool which can be used for analysis, when guided by its author

5. Software-as-a-Service – The prototype tool is presented to the analysis team as a "service"

6. Testing and refinement – A bi-directional process of requests from analysts to developers for assistance using the prototype tool. Testing and refinement can be done on live data prior to general release, allowing unfinished tools to still benefit analysts

7. Deployment – Production ready tools are released from the developer to analysts for personal use

8. Alternative SDLC – Larger tools or platforms may need a more rigid development methodologies which fall outside this process

9. Archive – Unsuccessful or no longer used tools should be saved, as pieces may be useful in future projects

10. DevOps – Active tools require operational support which may fall outside this process.

Appendix C – Malware Detection Design Example

Designing tools to detect malware variants is a good example of the highly iterative development process being used with a proactive review. The different and constantly changing malware ecosystem means that a single rule, indicator or behavioral check will never catch all possible malicious activities. Furthermore, malware authors intentionally change their tactics and monitor detection methods to increase the lifespan of infected hosts. This cat and mouse game leaves analysts always a step behind when trying to identify malware. The absence of live samples or test data can make creating and validating new methods even more difficult difficult. Leveraging real use cases when available is an opportunity to create and validate new methods of detection.

Each time a new malware variant is identified, whether through active investigation or comparison with intelligence sources, further review is required to determine which behaviors can be used with the available tools. The focus of this research is to identify unique behaviors or indicators that, when seen together, specifically identify the malicious traffic. This process may not be straightforward, with analysts attempting multiple combinations of logical structures around malware qualities before potentially finding a reliable detection method. Each test and set of results helps the analyst become a temporary subject matter expert. If the collection of these traits and their relationships produces a potential processes for identifying the malware, the process can then move on to development.

As the detection method is created the developer has opportunities to further validate the accuracy through presenting its use to other analysis. In this way, the new malware detection method can be released to the wider team, without requiring extensive training or interface code to be written. In some cases, these enhancements may take the simple forms of indicators added to automated tagging, whitelisting or new pre-built queries in tools, which can easily be delivered to analysts. Some threats are more difficult to detect by static indicators, and instead require procedural or behavioral approaches which are more suited to automation, initially requiring the developer's oversight. These methods use more advanced techniques such as correlating behavior across multiple events and even when packaged into tools can require additional documentation or modification to be sharable. Repeated use of the new tool from different perspectives allows the creator to refine any remaining issues and make the tool more universally usable.

Now available in the analysis toolkit, the new malware detection method can enter regular analyst use and the refinement cycle continues. The overall quality of the tool along with how frequently it is used will affect the number of bug reports and feature requests submitted. These numbers can then be compared against other tools, old and new, providing a very lightweight method for identifying important research and development tasks. Tools used frequently will naturally improve at a faster pace than those rarely needed. With only minimal overhead required, this method of prioritizing work can be extended to build longer-term road maps.

Over time, more malware detection methods are also added the analysis toolkit to increase its overall coverage. Despite using different tactics to detect each piece of malware, the post processing and follow-up analysis are generally very similar for findings produced by each distinct tool. Using the specific examples of each detection method to guide the a development of a generalized framework, new methods to be easily added with less effort required. This ultimately allows for the custom malware detection solutions to truly compliment traditional detection sources, such as IDS or anti-virus, by making deploying new methods an easy process.

Appendix D – References

a. https://gcn.com/articles/2014/12/09/dhs-ease.aspx
b. http://www.livescience.com/2493-mind-limit-4.html
c. http://www.darkreading.com/too-much-security-data-or-not-enough/d/d-id/1140635
d. http://www.networkworld.com/article/3110452/security/the-pressing-need-for-network-security-operations-automation.html
e. http://www.computerworld.com/article/3083264/it-careers/automation-not-cheap-labor-is-reshaping-outsourcing.html
f. http://www.apa.org/research/action/multitask.aspx
g. https://www.sans.org/reading-room/whitepapers/analyst/2015-analytics-intelligence-survey-36432 (page 13, figure 7)
h. http://www.csoonline.com/article/2154861/average-us-business-fields-10000-security-alerts-per-day-damballa-analysis-finds.html
i. https://en.wikipedia.org/wiki/List_of_software_development_philosophies
j. http://www.securityweek.com/incident-response-becoming-more-difficult-survey
k. http://www.drdobbs.com/siem-a-market-snapshot/197002909
l https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5112783/
m. https://www.cert.org/incident-management/csirt-development/csirt-staffing.cfm
n. https://www.scmagazineuk.com/1-in-3-businesses-have-no-incident-response-plan/article/541358/
o. https://www.hg.org/article.asp?id=5127
p. http://www.zdnet.com/article/security-landscape-plagued-by-too-many-vendors-cisco/
q. http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf (page 8, figure 4)
r. https://www.sans.org/reading-room/whitepapers/analyst/security-spending-trends-36697 (page 6, table 4)
s. https://www.hexiscyber.com/news/hot-topics/automation-%E2%80%93-force-multiplier-security-teams
t. http://arstechnica.com/security/2016/02/clever-bank-hack-allowed-crooks-to-make-unlimited-atm-withdrawals/