# *VisIBoT: Visualisation of IoT Botnets*

***Daniel Arthur, 2086380A***

## Proposal

### Motivation

*The recent demand for 'Internet of Things' and smart devices has resulted in many manufacturers rushing to release new products. This has resulted in many IoT devices being released with weak default passwords and various other vulnerabilities. As shown through the release of IoT-based botnet malware variants Mirai and Bashlite, Botnet owners have identified and exploited various vulnerabilities found in common IoT devices, allowing for such devices to be infected and controlled through Command & Control (C2) servers. These servers are responsible for coordinating various types of cyber-attacks, most commonly Distributed Denial of Service (DDoS) attacks.*

*The process of identifying, analysing and visualising such botnets and their corresponding C2 servers has long been a primary objective of many Cyber Threat Intelligence specialists, academics and agencies. This is necessary to understand how botnet malware is evolving, as new obfuscation and covert techniques are being employed by botnet owners, making botnet attacks more difficult to mitigate. The primary motivation for my work is to contribute to the field of Cyber Threat Intelligence through proposing a strategy for automated botnet identification and visualisation, which can serve as a basis for future research.*

### Aims

*This project proposes an approach to automated real-time IoT botnet and Command and Control server identification, analysis and visualisation. Using honeypot data provided by BadPackets, this project aims to identify and geo-locate botnet traffic, and additional identify potential command and control servers through automated static and dynamic analysis of malware extracted from honeypot payloads. An additional aim is to identify patterns and similarities between candidate C2 servers and Autonomous Systems.*

## Progress

- *Automatic retrieval, processing and storage of BadPackets API honeypot data*
- *Added MaxMinds Geo2 IP integration for geo-tagging botnet and payload IP addresses*
- *Implemented various payload de-obfuscation techniques for extracting malware URLs*
- *Added LiSa sandbox integration for automated malware scanning (static, dynamic and network analysis)*
- *Modified LiSa source-code to allow for remote binary URL submissions and bash script execution*
- *Added automatic scanning of malware binaries through VirusTotal via LiSa sandbox*
- *Designed and implemented some C2 server identification heuristics for candidate servers using LiSa sandbox analysis and IP blacklist detection.*

- *Implemented frontend website dashboard with interactive map (using nuxt.js and leaflet)*
- *Created backend API for interacting with frontend website dashboard (using express.js)*

## Problems and risks

### Problems

*I had initial struggles with grasping techniques for identifying candidate command and control servers using honeypot data available from the BadPackets API. However, I believe I had made some progress after finding a suitable malware analysis sandbox to work with.*

*As my data-set of C2 information is currently very small, I do not yet have the information available to experiment with ASN pattern identification techniques such as graphing and clustering. However, this will hopefully be easier to accomplish once I obtain a larger data-set of candidate C2 servers.*

### Risks

*I have some concerns with performing dynamic analysis of malware through execution within a LiSa sandbox. I need to investigate routing any incoming and out-going traffic from the sandbox through a VPN or possibly Tor to ensure that the hosting address of the sandbox does not become a target for potential attackers who have detected my attempts of analysis.*

*Additionally, I have concerns regarding my sandboxes potentially contributing to the spread of malware by scanning for vulnerable devices and/or hosting malware binaries. I am currently looking into ways to mitigate this through configuring linux services such as Uncomplicated Firewall (UFW). However, blocking ports may result in also blocking C2 traffic, which would be detrimental to the project. An alternative approach would be to modify the network layer between LiSa sandbox Qemu guests and the internet, but this is potentially too time consuming and complicated to implement.*

*Lastly, I am currently using MongoDB atlas to host the database for my project, however, running my processing script for a couple of weeks may result in the database reaching the 500mb capacity very quickly. I am currently looking into the MongoDB student pack which could yield $200 in MongoDB atlas credits. This would hopefully be enough if my database exceeds the 500mb free-tier capacity.*

## Plan

*Semester 1*

- *Week 13-15: Research and implement methods for mitigating out-going malware traffic when executing in sandboxes, such as LiSa. Additionally, continue to work on and finish VisIBot interactive map features and begin working on report.*
    - *Deliverable: Robust malware analysis process that counters out-going scanning traffic generated by analysed malware binaries (+ possible VPN/Tor traffic*

*routing to hide IP of sandbox server from attackers). Near-completion for frontend interactive map. Initial structure for project report.*

*Semester 2*

- *Week 1-2: Finalise C2 server identification and begin developing statistics/graph interface on frontend web interface.*
    - *Deliverable: Complete VisIBoT processing script and implement a variety of graphs using Chart.js.*
- *Week 3-5: Continue statistics interface and implement C2 and malware-based analysis (ASN analytics, common malware variants, botnet types)*
    - *Deliverable: Finished statistics interface which presents C2 and malware-based information*
- *Week 6: Refactoring, testing and deployment of processing script and LiSa sandbox to remote server.*
    - *Deliverable: A live instance of modified LiSa sandbox and VisIBoT processing script that will run in background and collect botnet information.*
- *Week 7-8: Initial report write-up and final implementation of frontend web interface*
    - *Deliverable: First draft of report and polished version of frontend web-interface*
- *Week 9: Deployment of frontend and backend services to remote server. Finalising first draft of report.*
    - *Deliverable: A website that will publicly host the VisIBoT web application and first draft submitted to supervisor three weeks before deadline.*
- *Week 10: Code clean-up and complete documentation. Improve first-draft pending feedback from the supervisor.*
    - *Deliverable: Final release of VisIBoT repository and near-completion of final report (2 weeks before deadline).*