Using graph databases to assess the security of thingernets based on the thingabilities and thingertivity of things

Prepared by:
Matt Lewis, Technical Research Director

Table of contents

1. Parlance

We define the following terms that will be used throughout this whitepaper:

- *Thing* – noun – an object, typically of inanimate material but in the context of thingernets and how people interact with things, might also include a living sentient being
- *Thingernet* - noun - a group or system of interconnected things
- *Thingability* – noun – possession by a thing of the means or skill to do something
- *Thingertivity* – noun – the capacity of a thing for interconnectivity and communication with other things due to its thingability when deployed within a thingernet
- *Internet of Things (IoT)* – noun – the interconnection of thingernets via the Internet

2. Introduction

“Security within the Internet of Things (IoT) is currently below par.”

The statement above derives from many observations across our work in IoT (and that of the wider security research community) in addition to a myriad of regular, publicly reported issues and security concerns with IoT devices and their infrastructures. Non-exhaustively, the following common themes surface:

- Some IoT device vendors not following security best practices and/or omitting crucial security controls and features within their devices
- There are a lack of defined and mandated security standards to which IoT devices should adhere
- IoT devices often include insecure default configurations, such as default passwords which aren’t changed by their users
- There are commonly missing secure update mechanisms to rectify security flaws in IoT devices that have been found and publicly disclosed
- The unintended consequence of complexity: As and when multiple IoT devices connect to homogenous networks, the potential attack paths and interoperability between devices within those networks grows almost exponentially

In turn, there is scope for a toolkit and method which allows for reasoning about the security of IoT devices and networks.

For this purpose, we set out an approach using graphs and graph databases to understand IoT network complexity and the impact that different devices and their profiles have on the overall security of the underlying network and its associated data.

When considering the security of “things”, it helps to define the capability (thingability) of that thing and the connectivity (thingertivity) it offers when connected to networks of things (thingernets[1]).

While, in this paper, we use thing-oriented phraseology for no reason other than whimsy, the aim of this paper is to seriously demonstrate the complexity involved with thingernets. We hope to begin a discussion about possible toolkits and methods that might be used by cyber security practitioners to facilitate IoT security assurance.

Who will find this paper useful?

- IoT device vendors who wish to understand the security impact of their thing when connected to thingernets and how/where that security impact can be reduced
- Thingernet maintainers (e.g. home users and their networks, or enterprise network administrators) looking to achieve a holistic view of a thingernet's security posture and where the core risks lie, especially when adding new things to thingernets
- Penetration testers identifying attack paths within thingernets
- IoT security standardisation workers producing IoT device and network security profile definitions and guidelines

3. Security properties (thingabilities) of a thing

Understanding the security properties of a thing allows us to appraise the risk associated with its use.

There are many different properties (and possible values of those properties) that we might want to consider, suppose we capture the following baseline properties of things:

Property	Possible Value	Notes
Name	Alphanumeric string	A name for the thing, e.g. “internet-connected TV”
Position	<Mobile> | <Static>	By this we mean whether the thing is mobile, in that it may physically move within and/or outside of its normal environment (such as an internet-connected toy which might be taken many places by its child owner), or static, meaning the thing remains *in situ* (such as a home router) Mobile things may present more of a security risk given they can connect to untrusted networks or be used in untrusted environments that might result in their compromise
Encryption	<Yes> | <No> | <Configurable>	We capture the encryption at rest property of the thing – either this is enabled, not possible/disabled or is configurable
Data classification	<None> | <Personal> | <Sensitive Personal> | <Corporate Sensitive> | <Mixed Business & Personal> | <Government Classified>	We need to understand the highest possible classification of data that might be stored within a thing – i.e. personal data, mixed business and personal in the case of Bring Your Own Device (BYOD) mobile devices etc.
Environmental effect	<None> | <Audio> | <Visual> | <AV> | <Heat> | <Actuator> | <Movement>	It is useful to understand the environmental effect of a thing, particularly if there’s a safety aspect that needs to be considered. For example, an internet-connected kettle produces heat and steam, while an internet-connected TV has effects on the audio and visual spectrums (AV)
Interfaces	See interface properties below	Things may have one or more interfaces, which allow for interconnectivity with other things and thingernets

Properties of a thing's interface

We capture the following baseline properties for each interface that a thing may possess:

Property	Possible Value	Notes
Name	Alphanumeric string	A name for the interface, e.g. "WiFi"
Connection type	<Wired> | <Wireless>	Whether the interface is wired or wireless
Authentication	<Yes> | <No> | <Configurable>	Whether the interface offers authentication (such as username/password/keys/certificates etc.)
Direction	<TX> | <RX> | <TXRX>	The direction of data flow through the interface, i.e. transmit (TX), receive (TX) or both (TXRX)
Protocol	Alphanumeric string	This is the baseline transport protocol for the interface – suppose the interface is "WiFi", then the baseline protocol would be 802.11
Crypto	<Yes> | <No> | <Configurable>	Whether there's any encryption on the data transport layer

4. Computing thingertivity

Having defined the thingabilities of things, we are able to begin looking at the permutations of their interconnections within a thingernet. For this purpose, we will use a graph database (*neo4j*[2]) which will allow us to visualise thingernets and query properties of things, their thingabilities and thingertivity.

Data structure for things

To automate the graphing of thingernets we first define an appropriate data structure to describe the thingabilities of our things. For this purpose, we use Javascript Object Notation (JSON) and, as an example, the following JSON describes the thingability of an internet-connected Teddy Bear using the property values defined in the previous section:

```
{"thing":[
    {
        "name" : "Teddybear",
        "position" : "Mobile",
        "encryption" : "No",
        "data" : "Sensitive",
        "effect" : "Audio",
        "interfaces" : [
        {
            "interface" : "Bluetooth",
            "conntype" : "Wireless",
            "auth" : "No",
            "direction" : "TXRX",
            "protocol" : "Bluetooth",
            "crypto" : "Yes"
        },
        {
            "interface" : "Speaker",
            "conntype" : "Wireless",
            "auth" : "No",
            "direction" : "TX",
            "protocol" : "Sound",
            "crypto" : "No"
        },
        {
            "interface" : "Microphone",
            "conntype" : "Wireless",
            "auth" : "No",
            "direction" : "RX",
            "protocol" : "Sound",
            "crypto" : "No"
        }
        ]
    }
    ]
}
```

In the example above, we describe our Teddy Bear - which is an internet-connected toy that allows children to speak questions into a microphone on the Teddy Bear.

The device connects via Bluetooth to an app running on a mobile device, which then connects to the internet to retrieve an answer to the question. After this, the question's answer is spoken back to the child via a loudspeaker within the Teddy Bear. This process is invisible to the child, and they are led to believe that the Teddy Bear is real and actually conversing with them.

For the Teddy Bear, we define its thingabilities as:

- Mobile: The toy can be moved around and outside of its normal environment
- Unencrypted: There's limited data storage in the toy but no mechanism to encrypt it at rest
- Sensitive in nature: Children may speak personal and/or sensitive information into the toy
- Audio-effecting: The toy listens (microphone) and speaks (loudspeaker), thus its environmental effect is audio (sound)

We then define three interfaces for the Teddy Bear:

- Bluetooth: This is a bi-directional link between the toy and a mobile handset used to send queries and receive responses. It is wireless and encrypted in transport but has no authentication (no Bluetooth pin pairing)
- Loudspeaker: This is wireless (audio) in transmission direction only. There is no authentication or encryption since it is audio (can be eavesdropped), thus we denote its protocol as 'Sound'
- Microphone: This is wireless (audio) in receive mode direction only. There is no authentication or encryption since it is audio (can be eavesdropped), thus again we denote its protocol as 'Sound'

Note that for our graphical model we also model people - classified as a Person - as things, since they physically interact with other things, such as pushing buttons on devices or speaking commands.

Similarly, from a data received perspective, people can hear audio and see visual feedback from devices. Our Person thing and thingability therefore looks like:

```
{"thing":[
    {
        "name" : "Person",
        "position" : "Mobile",
        "encryption" : "Yes",
        "data" : "All",
        "effect" : "All",
        "interfaces" : [
        {
            "interface" : "Speaker",
            "conntype" : "Wireless",
            "auth" : "No",
            "direction" : "TX",
            "protocol" : "Sound",
            "crypto" : "No"
```

```
        },
        {
            "interface" : "Microphone",
            "conntype" : "Wireless",
            "auth" : "No",
            "direction" : "RX",
            "protocol" : "Sound",
            "crypto" : "No"
        }
        ]
    }
    ]
}
```

A Person is mobile and we say that they are encrypted so far as that they can store secrets within their minds. However, they potentially store all manner of sensitive data.

In the example above, we provide a person with just two interfaces - a microphone and a speaker - to represent hearing and speech, respectively. We could add other interfaces to represent vision and touch (physical button pressing, for example) but, for brevity, we omit these.

5. Connecting the things

Now that we know the thingabilities and interfaces of things, we can automate the possibilities of their interconnectivity. For example, if two things have the same interface which supports the same protocol, and both link bi-directionally, then we can automatically graph a relationship between those two things via the respective protocol.

Similarly, we can marry mutually inclusive supported protocols, such as the 'Sound' protocol, between a microphone and a loudspeaker and vice-versa, while the same protocol on two devices in receive and transmit modes can be directionally-connected. Ultimately, this will allow us to produce a directed graph representing the data flows within a thingernet.

Given an arbitrary number of things, defined according to the JSON structure above, we use a Python script[3] to automate generation of the thingernet. In our example, the output is cypher code[4] – the code used by *neo4j* to define nodes and their relationships.

In a first, simple example, suppose we define four things in our thingernet representing a likely home network: a Home Router[5], a Laptop[6], an internet-connected Teddy Bear[7] and a Person[8].

By running the script, we get the following cypher code output which can be pushed to *neo4j* to produce the respective graph:

```
CREATE (HomeRouter:Thing {name:'HomeRouter', position:'Static', encryption:'No',
data:'Personal', effect:'None'})
CREATE (Person:Thing {name:'Person', position:'Mobile', encryption:'Yes', data:'All',
effect:'All'})
CREATE (Laptop:Thing {name:'Laptop', position:'Mobile', encryption:'Yes', data:'Mixed',
effect:'AV'})
CREATE (Teddybear:Thing {name:'Teddybear', position:'Mobile', encryption:'No',
data:'Sensitive', effect:'Audio'})
CREATE (HomeRouter)-[:Ethernet {conntype: 'Wired', auth: 'No', crypto: 'No'}]->(Laptop)
CREATE (HomeRouter)-[:WiFi {conntype: 'Wireless', auth: 'Configurable', crypto: 'No'}]-
>(Laptop)
CREATE (Person)-[:Sound {conntype: 'Wireless', auth: 'No', crypto: 'No'}]->(Laptop)
CREATE (Person)-[:Sound {conntype: 'Wireless', auth: 'No', crypto: 'No'}]->(Teddybear)
CREATE (Laptop)-[:WiFi {conntype: 'Wireless', auth: 'Yes', crypto: 'No'}]->(HomeRouter)
CREATE (Laptop)-[:Ethernet {conntype: 'Wired', auth: 'No', crypto: 'No'}]->(HomeRouter)
CREATE (Laptop)-[:Sound {conntype: 'Wireless', auth: 'No', crypto: 'No'}]->(Person)
CREATE (Laptop)-[:Bluetooth {conntype: 'Wireless', auth: 'Configurable', crypto: 'Yes'}]-
>(Teddybear)
CREATE (Laptop)-[:Sound {conntype: 'Wireless', auth: 'No', crypto: 'No'}]->(Teddybear)
CREATE (Teddybear)-[:Sound {conntype: 'Wireless', auth: 'No', crypto: 'No'}]->(Person)
CREATE (Teddybear)-[:Sound {conntype: 'Wireless', auth: 'No', crypto: 'No'}]->(Laptop)
CREATE (Teddybear)-[:Bluetooth {conntype: 'Wireless', auth: 'No', crypto: 'Yes'}]->(Laptop)
```

The resultant graph is illustrated below:

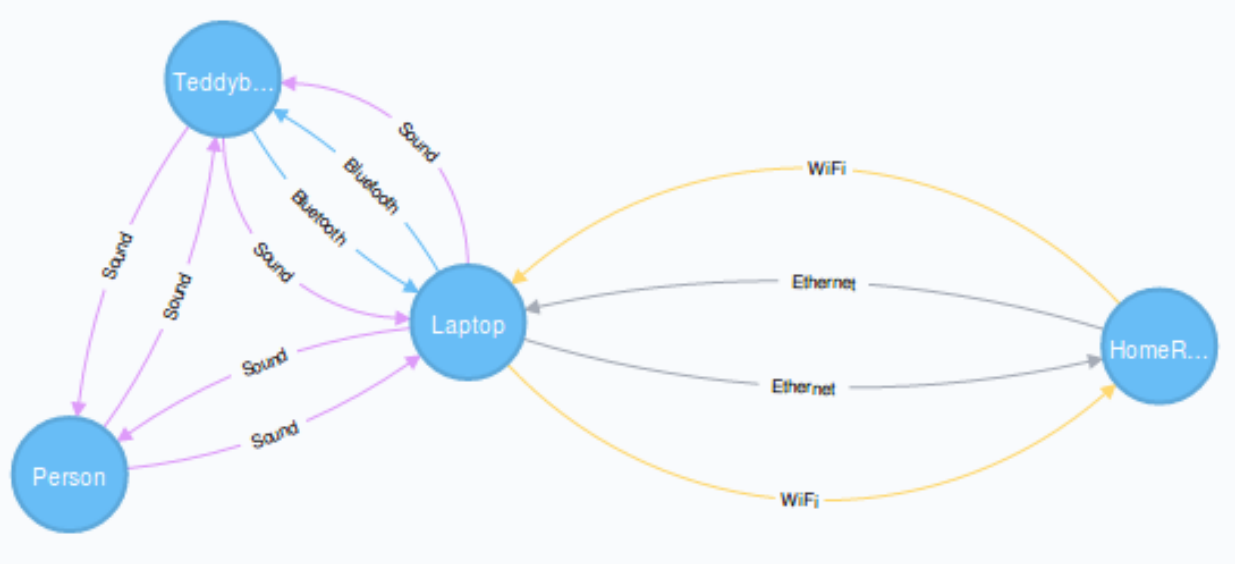

Figure 1 - Thingertivity of our simple thingernet

This simple graph of our thingernet shows all possible communication paths and protocol interactions. In addition, what's hidden from view but captured within the graph are the thingabilities of each thing and the various properties across all thingertivity.

Querying the thingernet

Now that we have created our thingernet graph with edges (thingertivity), we are able to query it and its properties (thingabilities) to answer security-related questions.

Suppose we want to query the simple graph above for all things that are mobile and that do not have encryption at rest:

```
MATCH (t:Thing {position:"Mobile", encryption:"No"}) RETURN t
```

We get one result, which is the Teddy Bear:

```
encryption          No
data                Sensitive
effect              Audio
name                Teddybear
position            Mobile
```

Suppose we want to find out the number of protocols (protocol cardinality) of the thingernet:

```
MATCH (thing)-[r]-() RETURN DISTINCT type(r)
```

We get 4 discrete protocols:

```
Ethernet
WiFi
Sound
Bluetooth
```

To get the shortest path between the Home Router and the Teddy Bear we can use:

```
MATCH (t1:Thing {name:"HomeRouter"}), (t2:Thing {name:"Teddybear"}), p =
shortestPath((t1)-[*]-(t2)) RETURN p
```

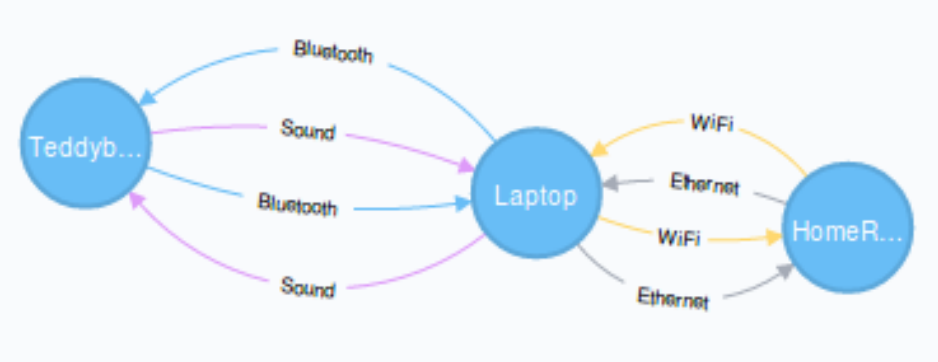

Figure 2 - Shortest path between Home Router and Teddy Bear

6. Thingernet risk metrics

The example above is trivial but serves to show the useful application of graph databases to our reasoning and understanding of thingernet risk and complexity. In the proceeding examples, we will use a more complicated thingernet comprising[9]:

- Home Router
- Internet-connected Teddy Bear
- Internet-connected TV
- Laptop
- Smartphone
- Digital Assistant
- USB Stick
- Person

We see that the resultant graph (below) appears messy and is not necessarily useful in its graphical form. However, we show how the graph nature of the thingernet, and our ability to query it, allows for various risk metrics to be defined that might allow us to reason about the security of the thingernet.

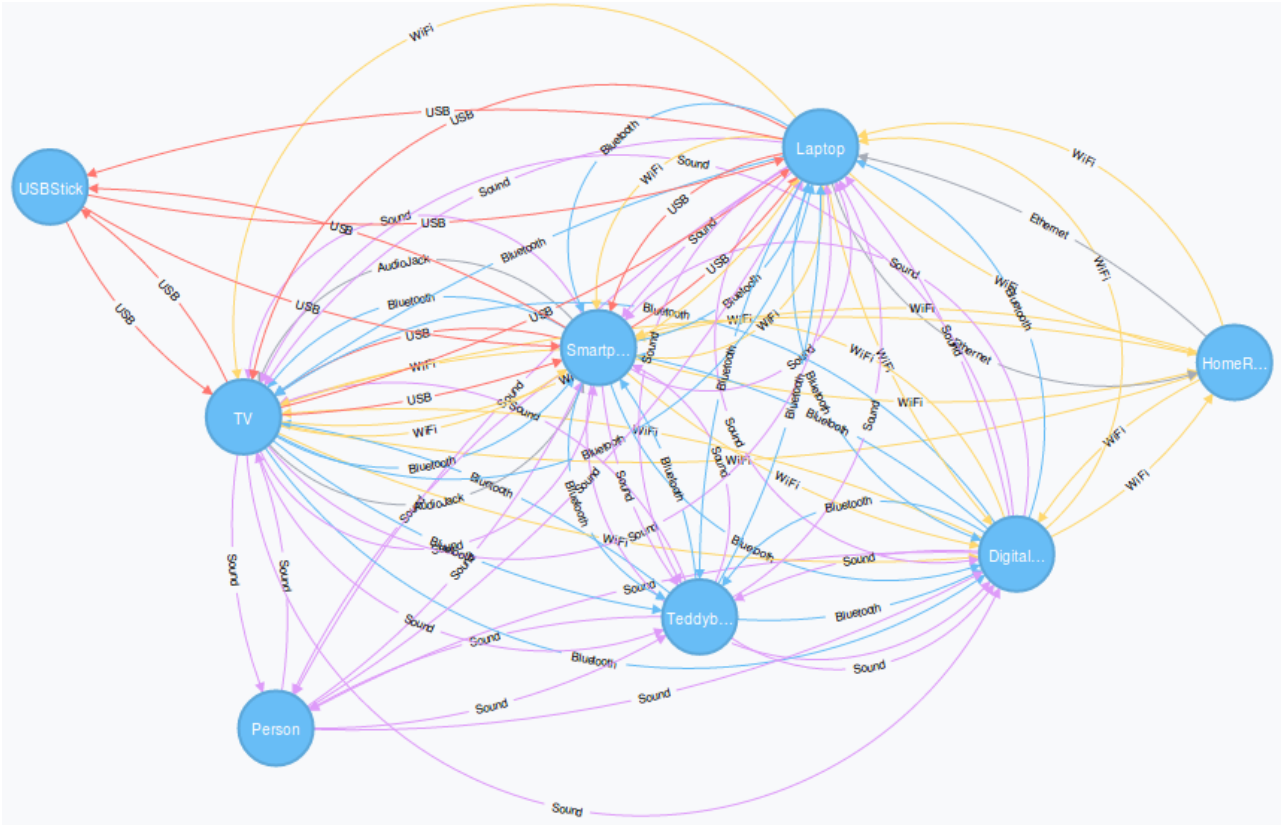

Figure 3 - Few devices, yet complicated thingernet

To begin, we can obtain the total number of relationships in our graph:

```
START r=relationship(*)
RETURN count(r)
```

This returns 86, so, already, we see that with a seemingly simple thingernet of eight things, there is much complexity by way of 86 different relationships between the things.

The protocol cardinality:

```
MATCH (thing)-[r]-() RETURN DISTINCT type(r)
```

Six distinct protocols:

```
WiFi
Ethernet
Sound
USB
Bluetooth
AudioJack
```

The wireless protocol cardinality:

```
match (n)-[r]-() where r.conntype="Wireless"
return distinct type(r)
```

Three distinct wireless protocols:

WiFi
Sound
Bluetooth

We can query the number of mobile devices in our thingernet:

```
MATCH (t:Thing {position:"Mobile"}) RETURN t
```

Five

Figure 4 - There are 5 Mobile Devices in the Thingernet

The number of devices without encryption at rest:

```
MATCH (t:Thing {encryption:"No"}) RETURN t
```

Five

The number of *mobile* devices without encryption at rest:

```
MATCH (t:Thing {position:"Mobile", encryption:"No"}) RETURN t
```

Two (Teddybear and USBStick)

The number of mobile devices without encryption at rest that store personal, sensitive or mixed data:

```
MATCH (t:Thing {position:"Mobile", encryption:"No"}) where t.data = "Personal" or
t.data = "Sensitive" or t.data = "Mixed" RETURN t
```

Two (Teddybear and USBStick)

We can query which things process audio, visual or both (so, potentially environmentally privacy-impacting):

```
MATCH (thing) WHERE thing.effect = "AV" or thing.effect = "Audio" or thing.effect =
"Visual" RETURN thing
```

Six devices:

```
Person
TV
DigitalAssistant
Smartphone
```

```
Laptop
Teddybear
```

We can retrieve things which have a physical effect on the environment (so, potentially safety-impacting):

```
MATCH (thing) WHERE thing.effect = "Heat" or thing.effect = "Actuator" or
thing.effect = "Movement" RETURN thing
```

Zero (none)

Across the thingernet we can query things that process personal or sensitive information:

```
MATCH (thing) WHERE thing.data = "Sensitive" or thing.data = "Personal" or
thing.data = "Mixed" RETURN thing
```

Eight

```
HomeRouter
Person
TV
DigitalAssistant
Smartphone
Laptop
USB Stick
Teddybear
```

We can query the number of relationships that aren't encrypted:

```
START r=relationship(*) where r.crypto="No"
RETURN count(r)
```

66

We can query the number of wireless relationships that aren't encrypted

```
START r=relationship(*) where r.conntype="Wireless" and r.crypto="No"
RETURN count(r)
```

50

The number of relationships each thing has (to determine which is most complex/offers largest potential attack surface and/or risk) :

```
start n=node(*) match (n)-[r]-() return n.name, count(r) as rel_count order by
rel_count desc
```

n.name	rel_count
Laptop	34
Smartphone	34
TV	34
DigitalAssistant	26
Teddybear	18
Person	10
HomeRouter	10
USBStick	6

We can retrieve the number of relationships that are *not* authenticated:

```
START r=relationship(*) where r.auth="No" RETURN count(r)
```

58

We can retrieve the number of relationships that aren't authenticated and aren't encrypted:

```
START r=relationship(*) where r.auth="No" and r.crypto="No" RETURN count(r)
```

50

Thingernet risk profile

We can summarise output from the queries above to gain an understanding of the thingernet's overall security posture and risk profile. To highlight risk we use a fairly crude colouring scheme:

- Medium risk (**Amber**): where 50 per cent to 74 per cent of the thingernet is affected by a particular metric
- High risk (**Red**): where 75 per cent to 100 per cent of the thingernet is affected by a particular metric

Quantification	Amount	% of entire Thingernet	Observation
General Metrics			
Protocol cardinality	6	N/A	There are 6 core protocols in operation on the thingernet. A greater number of protocols will increase the complexity and attack surface of the thingernet
Wireless protocol cardinality	3	**50%**	Half of the protocols supported by the thingernet are wireless. Wireless protocols are generally more at risk of eavesdropping and man-in-the-middle attacks
The most connected thing	Laptop	N/A	The laptop has most inbound and outbound connections, owing to the number of interfaces it possesses

Number of mobile things	5	63%	63% of the things in the thingernet are mobile, meaning they may move outside of their own thingernet and connect to other networks. Thus increasing the potential for their compromise and potential introduction of malware or unauthorised access when returned to their original thingernet
Privacy			
Number of things without encryption at rest	5	63%	63% of the things are not encrypting stored data on the thingernet
Number of mobile things without encryption at rest	2	25%	25% of the thingernet comprises mobile devices do not encrypt data at rest, rendering them vulnerable to data loss through lost or stolen devices when leaving the original thingernet
Number of mobile things without encryption at rest that process personal, sensitive or mixed information	2	25%	25% of the thingernet is comprised of mobile devices that do not encrypt data at rest and where that data is personal, sensitive or mixed in nature
Unencrypted relationships	66	77%	77% of the communication links in the thingernet are not encrypted, rendering them susceptible to eavesdropping and man-in-the-middle attacks
Unencrypted wireless relationships	50	58%	58% of the wireless connections in the thingernet are unencrypted, rendering them susceptible to eavesdropping and man-in-the-middle attacks
Number of things that process sensitive or personal information	8	100%	All devices in the network process sensitive or personal information
Number of relationships that aren't authenticated	58	67%	67% of the communication links are not authenticated, meaning attackers on the thingernet would be able to connect to interfaces without needing to authenticate
Number of relationships that aren't authenticated and aren't encrypted	50	58%	58% of the communication links are unauthenticated and unencrypted, rendering them susceptible to eavesdropping and man-in-the-middle attacks
Environmental privacy impact	6	75%	75% of the thingernet has audio/visual effect, meaning that there's high potential for privacy impact via recording of audio or visuals within the operating environment
Safety			
Physical safety impact	0	0%	None of the things interact with the environment in ways that might compromise human safety

Compilation of the table above allows us to quickly appraise the security and risk profile of our thingernet. The high-level observations are:

- Overall, the thingernet presents a high risk of data exposure and privacy impact (all things in the thingernet process sensitive or personal information)
- There is a broad lack of authentication and encryption of data (in both transit and at rest)
- There is medium to high risk of data exposure due to the use of wireless protocols
- There is a high risk of eavesdropping and man-in-the-middle attacks against the thingernet
- The Laptop poses the biggest risk to the network and thus demands secure lockdown
- There is no risk of physical harm within the thingernet

7. Attack paths

A useful application of graph databases to thingernets is the ability to get all paths between a source and destination. In the example below, we show paths between a Person and the Teddy Bear via the Sound protocol only (limiting results to the first five for brevity):

```
match p = (source)-[:Sound*]-(destination) where source.name='Person' and
destination.name='Teddybear' return extract(n in nodes(p)|n.name) AS Paths limit 5
```

```
[Person, Laptop, DigitalAssistant, Smartphone, Teddybear]

[Person, Teddybear, TV, Smartphone, DigitalAssistant, TV, Laptop, Teddybear,
DigitalAssistant, TV, Teddybear, Smartphone, Laptop, Smartphone, TV, Laptop,
Teddybear, Person, Laptop, DigitalAssistant, Smartphone, Teddybear]

[Person, DigitalAssistant, Person, Teddybear, TV, Smartphone, DigitalAssistant, TV,
Laptop, Teddybear, DigitalAssistant, TV, Teddybear, Smartphone, Laptop, Smartphone,
TV, Laptop, Teddybear, Person, Laptop, DigitalAssistant, Smartphone, Teddybear]

[Person, Smartphone, Person, DigitalAssistant, Person, Teddybear, TV, Smartphone,
DigitalAssistant, TV, Laptop, Teddybear, DigitalAssistant, TV, Teddybear,
Smartphone, Laptop, Smartphone, TV, Laptop, Teddybear, Person, Laptop,
DigitalAssistant, Smartphone, Teddybear]

[Person, TV, Person, Smartphone, Person, DigitalAssistant, Person, Teddybear, TV,
Smartphone, DigitalAssistant, TV, Laptop, Teddybear, DigitalAssistant, TV,
Teddybear, Smartphone, Laptop, Smartphone, TV, Laptop, Teddybear, Person, Laptop,
DigitalAssistant, Smartphone, Teddybear]
```

In the output above, we can see from the first result how the Person could speak to the Laptop's microphone which, in turn, could output sound that would be consumed by the DigitalAssistant's microphone. This could then output sound that would be consumed by the Smartphone's microphone and could, in turn, project sound to be received by the Teddy Bear.

These results show interesting and possibly unconsidered attack paths by IoT device vendors and system owners/implementers.

Further complexity is seen from the other results whereby the paths are cyclic in nature, i.e. we may pass through the same device via sound multiple times. The second result in the output above shows a more complicated traversal of sound around the thingernet. These paths allow us to identify more nuanced attacks[10] that are perhaps unique to thingernets and would otherwise be difficult to enumerate without a querying/automated mechanism as offered through graph databases.

8. Conclusion

There are some limitations to the thingernet reasoning approach described in this whitepaper. However, additional work could undertaken to further enrich the models that we are capturing and to reduce the current limitations – there are likely many other properties of devices and interfaces that we could be capturing (which might be security-impacting).

In reality, protocols such as WiFi will have multiple transport and application layer protocols, each of which may offer their own methods of authentication, authorisation and encryption. Furthermore, when marking relationships or data storage as encrypted, we are not currently capturing the strength and type of encryption employed.

Nevertheless, the aim of this paper has been to demonstrate the utility of reasoning about the security of thingernets and to understand the individual security impacts that different things and their profiles have on homogenous networks.

Using graph databases to describe and query properties of thingernets provides a convenient and repeatable method of reasoning about the security of the underlying network and the data that it processes – the outputs support a risk assessment of thingernets and, when used as a tool during system design and threat modelling, can facilitate concepts of security and privacy by design. As the number of devices in a thingernet increases, so does the thingernet's complexity, which can be trivially appraised and reasoned using a graph database approach.

9. References

[1] Credit to my learned colleague Aaron Adams for coining the term ‘Thingernet’

[2] https://neo4j.com/

[3] https://github.com/nccgroup/thingernet-graph

[4] https://neo4j.com/developer/cypher-query-language/

[5] https://github.com/nccgroup/thingernet-graph/blob/master/router.json

[6] https://github.com/nccgroup/thingernet-graph/blob/master/laptop.json

[7] https://github.com/nccgroup/thingernet-graph/blob/master/teddybear.json

[8] https://github.com/nccgroup/thingernet-graph/blob/master/person.json

[9] https://github.com/nccgroup/thingernet-graph

[10] http://www.bbc.co.uk/news/av/technology-38966285/how-hackers-could-use-doll-to-open-your-front-door

10. About NCC Group

NCC Group is a global expert in cyber security and risk mitigation, working with businesses to protect their brand, value and reputation against the ever-evolving threat landscape.

With our knowledge, experience and global footprint, we are best placed to help businesses identify, assess, mitigate & respond to the risks they face.

We are passionate about making the Internet safer and revolutionising the way in which organisations think about cyber security.

Headquartered in Manchester, UK, with over 35 offices across the world, NCC Group employs more than 2,000 people and is a trusted advisor to 15,000 clients worldwide.

NCCG/SC/WP/THING/V1/06/17