Context Aware Authentication Using IoT Services

Daniel Myers*, Larry Rapier†, Wenhui Zhang‡
Department of Computer Science, Northern Kentucky University
Highland Heights, Ky
Email: *myersd4@nku.edu, †lrapier@nku.edu, ‡wzhang2@nku.edu

I. Introduction

The purpose of context aware authentication is to protect against unauthorized access to a system in the event that an attacker acquires a user's credentials. Regardless, if the credentials are obtained through social engineering or compromising the system at a technical level, context aware authentication will make it extremely difficult for an impersonator to log into the system by checking the "context" of the surrounding Internet of Things (IoT) devices and determining if the intended user is the person attempting to log into the system at a given location.

Given the prevalence of networked devices which are pervasive in every facet of the general population's daily interactions, context aware authentication is already viable and becoming more relevant with each passing day. Now that the IoT sensor network envisioned by proponents of ubiquitous computing is coming to fruition, the amount of available context for any given situation is steadily increasing. Therefore, context based authentication can provide enhanced security for a variety of everyday applications such as webmail, credit card transactions, and accessing physical locations. Int this paper, we propose Pinpoint, a context-aware authentication backend to facilitate the use of security enhanced logins.

A. Motivation and Contribution

In the current state of network and systems security, system designers overwhelmingly rely on passwords in order to authenticate users and prevent unauthorized access to their products. While there have been some advancements in this approach, namely the widespread adoption of multi-factor authentication (which could be considered a type of context), relying on passwords for authentication leaves a system open to security breaches through a variety of channels because anyone with the password is assumed to be the correct user. Whether the credentials are leaked through social engineering or security vulnerabilities in the system itself the end result is the same, once credentials are compromised an attacker has the same access to a system as the user they are impersonating. Attacks involving compromised credentials could potentially be thwarted if the system was able to verify that the user requesting access was indeed who they claim to be.

For some time now, biometrics have been available that allow users to positively identify themselves, but these are both expensive and difficult to set up for the average person. Therefore, it seems that users need a simple, transparent, and secure way to log into various systems that can not only prevent unauthorized access, but positively verify a user's identity. Furthermore, any proposed authentication scheme must be easy for the average user as well as cost effective or the adoption of such a method will be in doubt. The Internet of Things (IoT) presents a unique opportunity to rethink and improve traditional IT security by using the sensor/actuator networks that are becoming increasingly available.

B. Approach

In order to keep the initial implementation streamlined and achievable in the allotted time frame, this implementation will consider a positive response from n/2 sources as a confirmation of the user's presence and allow access with a password at that time. The verification method will also be true for any secondary users that the primary user grants access to the system under their account. However, when a temporary user is logging into another user's account, the temporary user will receive a one-time passcode and their access can be revoked by the original user. Furthermore, the implementation will also account for users that have forgotten their password, and use the same location-aware IoT confirmation to either reject or accept a password reset attempt.

Users will be able to securely log into a variety of devices and systems with minimal input since their identity will be confirmed by their presence in the context-aware IoT ecosystem. A context-aware authentication system improves security and makes login easier on the user by not requiring the user to actively confirm their identity using the current accepted standard of two factor authentication, which usually involves entering a code after the username and password are accepted.

In addition, context-aware systems will be able to reconfigure security functions and features based on the user's physical location. Context-aware authentication improves security by restricting access to user accounts if the system is unable to verify that the user is physically present at a given terminal during the login attempt. For instance, if Alice is in Cincinnati, there is probably no valid reason for Bob or anyone else to attempt to access her account from outside of the Cincinnati area (or the building for that matter).

However, there are always scenarios where a user might need access to the data contained in someone else's account, so the system will facilitate those security exceptions by allowing the owner to delegate access to their files and instances. This is an improvement over current security practices that would most likely involve the sharing of a password to facilitate account access for the secondary user. These types of security exceptions will need to be fine tuned in order to minimize both the security vulnerabilities and inconvenience to the users.

C. Previous Work

This paper builds of the work of many other researchers that have explored the application of contextual data as it relates to IoT as well as how to use certain types of context for authentication. In Agadakos et al. [1] the authors propose a location enhanced authentication system called Icelus which leverages IoT devices to act as "beacons" which detect the presence of user devices (referred to as "trinkets") and issue a binary response indicating the user's presence at a given location. While Agadakos et al. provide a complete model for a system which leverages context and IoT to improve upon current authentication schemes, they only consider the possible of location as context and do not address any other type of context than can be gleaned form the considerable amount of data being generated from IoT sensors.

A detailed overview of the scope of context generated by IoT sensors can be found in Habib and Leister [2] in which the authors provide a system for categorization of IoT context. In addition to the preceding contributions, the authors also describe different types of context-awareness for systems that aim the leverage context in order to glean information from the environment.

Strang and Linnhoff-Popien [3] provides an explanation of the benefits that context-aware systems can provide as well as various strategies for data modeling in systems leveraging context.

II. Security

A. General Security Concerns

In any authentication system security is a major concern as authentication is usually one of the major attack vectors that are explored when attackers are attempting to compromise a system. The need for well designed security measures is only increased in the case of context aware authentication systems because the process for authenticating a user becomes distributed. Now, instead of trying to guess a password or crack a weak password hash, attackers may attempt to impersonate devices or use a man in the middle attack to make it appear that the user is in fact present for login attempt. Furthermore, if users begin to think that their passwords are irrelevant due to the presence of context-aware login systems, it could lead many people to choose poor passwords (e.g. "password1") thus reducing the effectiveness of login credentials which need to remain as the first layer of security when logging into a system.

B. Context Aware Security

The idea of context aware security to gather the "context" of the current login attempt from a variety of devices. These devices could include almost any type of sensor including card scanners, thermometers, motion sensors, and microphones. Using the context provided by these devices, PinPoint would then attempt to determine whether or not the person attempting to log into the system is indeed the person they are claiming to be.

C. Secure Communication

Given The distributed nature of context aware authentication systems, providing secure communication between the various components is essential to prevent a number of common attacks such as man in the middle, device spoofing (pretending to be a valid sensor) and packet injection (inserting fake context into a data stream). Most of these security concerns can be mitigated using well known technologies and authentication methods such as TLS to encrypt data in flight as well as whitelisting devices using fingerprints or IP addresses.

D. Pinpoint Security

PinPoint, which is the central component of the centralized authentication system has a number of administrative security concerns that need to be actively managed in order to prevent unauthorized user access to the applications that it would serve. For instance, the ability to define what context constitutes a positive identification of the user must be tightly controlled

in order to prevent an attacker from weakening the requirements for a positive identification. Furthermore, if user's choose weaker passwords because of the presence of context-aware authentication (see first subsection), an attack reducing or eliminating the context requirements could result in highly effective brute force attacks against a login terminal.

III. System Architecture

There are a variety of requirements that must be inherently satisfied due to the nature of authentication systems as well as the unique nature of context aware authentication. By considering these requirements early in the design process and building the system to take these considerations into account, the chance that context aware authentication can become a viable authentication system are greatly increased.

A. Design Patterns

In order to remain as general as possible, PinPoint will utilize an adapter pattern in order to communicate with various IoT devices and api gateways in order to gather user status. For instance, a smart lock and home speaker such as Amazon's Alexa, would have the same method call such as device.get_status(), but the get_status() call will determine the device type and call the appropriate adapter in order to gather the context of the device and determine if the login attempt is valid. Furthermore, all communication classes will be decoupled from the logical implementation of the core authentication framework, this will allow for the addition of communication protocols (e.g. REST and CoAP) with very little change to the core authentication logic. The need to support and swap out communication protocols will be essential when dealing with IoT devices that have no standardized communication protocol.

B. Speed

Due to the critical nature of authentication systems, Pinpoint will utilize a number of strategies to remain highly responsive to requests for user verification. One such strategy will be to gather context from devices and store the context in a way which can be quickly retrieved. For example, when a key card is scanned, the card information as well as a timestamp will be stored in the database with a time to live (TTL). Once the TTL has elapsed and the event can no longer be used as context for a login, the record will be deleted from the database in order to free up room and ensure the speed of lookups remains constant. This assumes that PinPoint will not be used as the source of record for keeping track of events like building access and user interactions with various devices. If PinPoint is expected to perform record keeping duties, a secondary database such as Mongo DB or Dynamo DB could be used to store currently valid context and perform rapid lookups while the main relational database could serve as the source of record for various user interactions.

C. Redundancy

Since authentication systems are generally mission critical and require constant up time, redundancy must be considered during the initial stages of system design. PinPoint lends itself to being implemented in a redundant fashion due to the highly distributed nature of context aware authentication. For instance, the login application programming interfaces (apis), database(s), and context gathering endpoints/tasks could operate on different nodes which could be scaled independently of one another. This means that multiple nodes can be deployed to handle each part of the system, so if any one node goes down the system will still operate. Furthermore, PinPoint will need to consider the possibility of IoT nodes being offline and be able to automatically respond to those situations. One example is when a sensor goes offline either due to hardware failure or a network outage. Given the preceding situation a fallback behavior must be defined so that system continuity is not affected. In that situation, a different sensor or set of sensors could be used to determine the status of the user or the system could start issuing blanket approvals or denials, assuming the username and password entered are correct. Another possibility is that a back second factor comes into play such as SMS codes or phone calls. The course of action depends on the needs of the organization and the capabilities of the system implementing the authentication request.

D. Login Sequence

The diagram below shows the sequence of a user login in three stages. First, the user submits credentials to a login terminal for a given application. This login terminal could be for any application and would only communicate with the PinPoint system in order verify the user's identity and authorization, in much the same way SAML authentication works today. After the user submits their credentials, the application would submit the credentials to PinPoint which would then gather context information from the available IoT s. Based on the context gathered and the defined requirements for a positive identification, Pinpoint will then return an

assertion to the application in order to communicate whether the login is approved or denied as well as any user specific data such as permissions levels (in the event of a successful login).

E. Use Cases

There are a wide variety of use cases in which context aware authentication can be employed to increase security. Some have already been mentioned, such as highly secure localized systems that would be deployed by financial and defense companies. PinPoint could also be used to change the habits of organizations that commonly share passwords between users which reduces accountability in the system. By confirming that the user which is trying to access the system is currently in or around the given terminal, it would eliminate or greatly reduce the practice of password sharing. In fact, PinPoint could be leveraged make cross account more secure by explicitly allowing users to delegate temporary access to their account by issuing one time passwords and confirming that the user receiving the password is confirmed in the context of the system before allowing login.

IV. Implementation

In order to make the reference implementation of PinPoint as simple as possible to build and run, the implementation described in this paper will use common open source web technologies that are readily available and widely supported.

A. Technology Stack

The initial implementation of PinPoint will utilize the Python [4] programming language and leverage the Django Framework [5] to implement the api layer of the application. The data layer will reside on a separate instance and use Postgresql [6] which is a fully ANSI compliant open source database. All of the components for the PinPoint backend will be hosted on Amazon Web Services [7] using an EC2 t3.micro instance (2 vCPU cores and 1 GB RAM) for the web node and a db.t2.micro (1 vCPU core and 1 GB RAM) Postgresql RDS for the data layer.

In addition to the PinPoint backend, an NFC relay is also implmented in Python which receives commands from the GoToTags [8] application when a key card is scanned. The key card scanner is an ACR122U created by Advanced Card Systems [9] and is locally connected to a personal computer to imitate the act of scanning a key card at a door or similar entry point. The cards used contain the Mifare classic 1k chipset which holds one kilobyte of data.

B. Context

One of the main challenges in building any context-aware system is identifying an effective strategy for filtering context which will allow the system to quickly identify relevant context. ”IoT sensors continuously generate enormous amounts of data, which is the digital implication of external environment and system behavior” [10]. Due to the fact that certain sensors generate a continuous stream of data the amount of data needing to be stored can quickly grow to unmanageable levels, especially for systems that cannot implement big-data solutions such as Hadoop due to technical or operational constraints.

During the experiment described in this paper, the context being passed by the nfc relay conforms to the NFC Data Exchange Format(NDEF) and will be in the form of a contact tag. NDEF tags are encoded as Javascript Object Notation(JSON) when being written to the key cards which makes the task of parsing and storing the data trivial. An Example of an NDEF contact tag can be seen in listing 1 below. Some storage space could be saved by trimming down the record contained in the list below but it would then not conform to NDEF standards.

Listing 1. NDEF Contact Tag

```
{
  "uid": "9B418A0A",
  "records": [
  {
    "Note": null,
        "Role": "Student",
        "Email": null,
        "Title": null,
        "FaxWork": null,
        "UrlWork": null,
        "LastName": "Last",
        "UniqueId": null,
        "BirthDate": "",
        "FirstName": "First",
        "PhoneCell": null,
        "PhoneHome": null,
        "PhoneWork": null,
        "ProductId": null,
        "NamePrefix": null,
        "NameSuffix": null,
        "PhoneVoice": null,
        "RecordType": "Contact",
        "UrlPersonal": null,
        "Organization": "NKU",
        "EmailPreferred": "xxxx@nku.edu",
        "AddressHomeCity": null,
        "AddressWorkCity": null,
        "AddressHomeRegion": null,
        "AddressHomeStreet": null,
        "AddressWorkRegion": null,
```

```
        "AddressWorkStreet": null,
        "AddressHomeCountry": null,
        "AddressWorkCountry": null,
        "AddressHomePostalCode": null,
        "AddressWorkPostalCode": null
    }
    ],
    "tagtech": "MifareClassic1K",
    "readonly": false,
    "exception": null,
    "formatted": true,
    "datalength": 166,
    "maxdatalength": 716,
    "canmakereadonly": true
}
```

Once the context is received in the PinPoint backend, it will be stored in the database with the time the context was received as well as the entire NDEF tag. The tag will be stored in a Postgresql specific field called JSONB which is a database field that only stores properly formatted JSON. The advantage to storing the data in this manner is that JSONB fields are indexable as well as searchable which enables the system to perform quick lookups on the field without leveraging any additional technologies. PinPoint's context model is provided in listing 2.

Listing 2. Context Model

```
class Context(models.Model):
    uid = models.CharField(max_length=512)
    device = models.ForeignKey(
        Device,
        on_delete=models.CASCADE
    )
    ttl = models.IntegerField(default=600)
    context = JSONField()
    created = models.DateTimeField(
        auto_now_add=True
    )
    updated = models.DateTimeField(
        auto_now = True
    )
```

C. Context Filtering Strategy

Pinpoint will utilize a combination of temporal constraints as well as a key value store in an attempt to quickly identify a subset of potentially relevant context and check it against the user awaiting login. The checks will happen in real-time when a user submits their credentials to a terminal that is using PinPoint as an authentication backend. Furthermore, context will be retrieved in the order of most to least recent which increases the likely hood that valid context will be found before examining most of the data set. Another consideration when considering the filtering of context in the PinPoint authentication system is that the initial implementation only passively gathers context, meaning that the context generating nodes need to explicitly send their context to the authentication system. In practice this means that any context received by PinPoint is most likely relevant to one of the users registered the system. This method of context gathering will greatly reduce the noise compared to collecting everything in a sensor network for a given location. The code in listing 3 shows the check that is run on all context attached to the configured key card reader.

Listing 3. Context Filtering Checks

```
def get_status(self, **kwargs):
    if 'user' not in kwargs:
        return False

    if 'context' not in kwargs:
        return False

    user_email = kwargs['user'].email
    context = kwargs['context']
    cutoff_time = datetime.datetime.now(pytz.utc) -
        datetime.timedelta(seconds=context.ttl)

    contexts = Context.objects.filter(
        created__gte=cutoff_time,
        context__records__0__EmailPreferred=user_email
    )
    if contexts:
        return True
    else:
        return False
```

V. System Evaluation

The system evaluation involved measuring the time elapsed during a user login Request in order to ascertain how the system would perform given varying levels of context. For this test, the user login was testing with 100, 1000, 10000, 100000, and 10000000 entries in the context table. During each level of testing the system was tested in two scenarios:

1) No valid context - Despite the fact that the context table has the number of items for the given iteration, no valid context exists for that login.
2) One valid context - Only one piece of valid context exists for the given login. This was done in an attempt to provide a worst case lookup since the database will need to find the one matching record in the table.

All of the context levels were run 3 times and the average of those tests appears in the table below:

Context Count	Fail Time	Success Time
100	942.808 ms	984.58 ms
1,000	769.104 ms	1142.147 ms
10,000	822.841 ms	1097.093 ms
100,000	1092.189 ms	1291.088 ms
1,000,000	13463.753 ms	113702.113 ms

From the table above it is apparent that there is a significant performance degradation between 100,000 and 1,000,000 items. This is most likely a limitation of the hardware that this particular implementation is utilizing since Postgresql needs to store the entire index for a table in memory in order to achieve optimal performance. When the size of the table reached 1,000,000 items the table index size around 750mb due to the fact that all of the JSON fields were indexed for searching. It is unlikely that the RDS instance used for this paper was able to store the entire index in memory because it had 1 GB of RAM for the whole database.

VI. Future Work

There are a number of areas that can be explored in more detail as to improve the reference implementation provided in this paper.

- Context searching - The method for searching context in this paper relies on specific implementations written for each device. While this will likely reduce the number of false positives and allow for greater context filtering on the way into the system, it is slower than utilizing a searching tool such as Elastic Search or possibly a high speed cache such as Redis.
- Active context collection - In this implementation the PinPoint backend passively received context in which must be filtered in order to approve or deny a user login. In future implementations, the ability to actively gather context from a set of heterogeneous devices must be explored.

VII. Conclusion

This paper has put forth a system called PinPoint which focuses on using the context created by IoT devices in order to ensure that users logging into a system are actually the person authorized to access the system. Furthermore, a reference implementation of the proposed system has been created an tested against varying levels of context. In conclusion, the paper has demonstrated the viability of context-aware authentication systems and provided a starting point for future implementations which can be leveraged to enhance the security of a wide range of systems.

References

[1] I. Agadakos, P. Hallgren, D. Damopoulos, A. Sabelfeld, and G. Portokalidis, "Location-enhanced authentication using the iot: Because you cannot be in two places at once," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. New York, NY, USA: ACM, 2016, pp. 251–264. [Online]. Available: http://doi.acm.org/10.1145/2991079.2991090

[2] K. Habib and W. Leister, "Context-aware authentication for the internet of things," in *The Eleventh International Conference on Autonomic and Autonomous Systems*, ser. ICAS 2015, 05 2015, pp. 134–139.

[3] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Workshop on advanced context modelling, reasoning and management, UbiComp*, vol. 4, 2004, pp. 34–41.

[4] "Python." [Online]. Available: https://www.python.org/

[5] "Django." [Online]. Available: https://www.djangoproject.com/

[6] "Postgresql." [Online]. Available: https://www.postgresql.org/

[7] "Amazon web services (aws)." [Online]. Available: https://aws.amazon.com/

[8] "Gototags." [Online]. Available: https://gototags.com/

[9] "Advanced card systems - acr122u." [Online]. Available: https://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/

[10] R. Song, Y. Wang, W. Cui, J. Vanthienen, and L. Huang, "Towards improving context interpretation in the iot paradigm: A solution to integrate context information in process models," in *Proceedings of the 2018 2Nd International Conference on Management Engineering, Software Engineering and Service Sciences*, ser. ICMSS 2018. New York, NY, USA: ACM, 2018, pp. 223–228. [Online]. Available: http://doi.acm.org/10.1145/3180374.3181341