STIX Cyber Observable - Credential Dump

"What users have been pwned?"

What is it?

On most of the secretive 'trusted' threat intelligence sharing communities, people share lists of compromised user credentials over encrypted email. Organizations who consume that information have to manually process the lists of thousands of credentials to see if they have any users that reuse those credentials on their service.

Wouldn't be great if we could reduce the time window that these compromised credentials were useful?

Wouldn't it be great if it was possible to share these credentials in a structured format to facilitate recipient organizations automatically and rapidly ingesting these compromised credentials and locking the compromised accounts down? Or other recipients automatically testing their own services for password reuse?

If we implemented this proposal, people would be able to share lists of known compromised credentials, and impacted organizations would be able to lock down the compromised creds, reducing the time window that the attacker can exploit them.

Why do we need it?

Sharing of this information already happens within many threat intelligence sharing trustgroups I am part of. There are people sharing lists of compromised credentials to allow recipients to test their own logins for credential reuse, or to provide the organization that originally hosted the credentials with a list of credentials that need to be investigated for unauthorized access.

Providing a structured way for sharing this data at scale and in an automated fashion will allow a rapid response to compromised credentials, further shrinking the timeframe that miscreants have to exploit the compromised accounts.

How would it work?

I propose that we add a new STIX Cyber Observable (SCO) object to STIX v2.1 - the **Credential Dump** object. This object would provide a list of login credentials that appear to have been compromised.

What benefits would it provide?

- Providing a structured way for sharing this data at scale and in an automated fashion will allow a rapid response to compromised credentials, further shrinking the timeframe that miscreants have to exploit the compromised accounts.

But it's PII! I'll get sued!

Firstly - I am not a lawyer.

Secondly - this information is already shared, traded and sold on the underground by miscreants. It is already shared and traded by those who wish to help stop the miscreants gaining an advantage. This information is already being shared now.

We need a way of helping reduce the time window as close as we can to zero, making compromised credentials worthless.

STIX Cyber Observable object proposal

2.7.Credential Dump Object

Type Name: `credential-dump`

The Credential Dump Object represents credential dump containing username and password information that attackers have gained access to and dumped somewhere on the web in public or traded for money. It is primarily to enable the sharing of credential dump information to allow the remediation of affected users.

2.7.1. Properties

Common Properties		
type, description, extensions		
Webpage Object Specific Properties		
Date, credentials		
Property Name	**Type**	**Description**
type (required)	`string`	The value of this property **MUST** be `credential-dump`.
date (optional)	`timestamp`	Specifies the date/time that the list of credentials were collected by the Object Creator.
credentials (required)	`list` of type `credential`	Specifies a list of credential objects that contain credential objects.

Credential Object (`credential`)

The Credential Object specifies a single credential to capture details for a specific login and password combination. It is only used within the Credential Dump object, and is used to enable the sharing of credential dumps to enable consumers to remediate those affected users or to check for password reuse within their organization.

Properties

Property Name	Type	Description
username (optional)	`string`	Specifies the username of the credential
cleartext_password (optional)	`string`	Specifies the credential's cleartext password.
hashed_password (optional)	`string`	Specifies the credential's password hash.
password_hash_function (optional)	`string`	Specifies the password hashing function used to hash the credential password.
password_salt (optional)	`string`	Specifies the salt used in the hashing function
email_address (optional)	`string`	Specifies the email address associated with the credentials.

Examples

Credential dump

```
}
  "0": {
    "type": "credential-dump",
    "credentials": [
      {
        "username": "user1",
        "cleartext_password": "mysimplepassword"
      },
      {
        "username": "user2",
        "cleartext_password": "mysimplepassword"
      },
      {
        "username": "user3",
        "hashed_password": "b9f621a04b077c9fb742caa1063f50a7"
        "password_hash_function": "MD5"
      }
    }
  }
}
```