Vulnerabilities in the SaaS era

SaaS as the new attack vector
About me

- Noam Liran
  — Researcher
  — Developer
  — Gamer

- Chief Software Architect of Adallom
- Former cyber team leader in the IDF
My second home. More than 25 brothers in arms

Founded in 2012 by former members of Unit 8200

Named after the "Ad Halom" bridge in Israel

Our money is on:
— SaaS adoption will skyrocket
— History is bound to repeat itself
— SaaS is secure – it’s not.
This talk’s purpose:

— Demonstrate how Enterprises use SaaS
— Get you thinking about SaaS security
— Question the transparency of SaaS security

How many of you use SaaS?
Cloud – marketing buzz word

On-premise – “datacenters” in enterprises

SaaS – Software as a Service

Google Apps (Gmail, Drive), Dropbox, Box, Office 365, Salesforce, SuccessFactors, LivePerson, Jive

PaaS – Platform as a Service

— Azure, Force.com, Heroku

IaaS – Infrastructure as a Service

— Azure, Amazon EC2
Story background – Enterprises

- The “old world” – on-premise networks
- Multiple on-premise services:
  - IM & Mails
  - CRM
  - HR management
  - Collaboration (file sharing)
- Very well-defined perimeter
Several types of users:

- Regular users (9-to-5, no home access)
- Power users (home access)
- Travelers (on-the-road access)

Users need remote access to resources

- How to allow access AND keep things secure?
Enterprise users

- **Solution depends on the specific sector**
  - Financial and medical institutions are the strictest

- **Some allow external access to resources**
  - Sometimes coupled with 2-factor auth. (OTP)

- **VPN clients and company policy enforcement**
  - Managed laptops
  - VPN with enforcement of strict OS, AV, FW policies
Perimeters under company policy

Internal
- Workstation
- CRM
- ERP
- Mail
- HR
- Files

DMZ
- Mail services

Al Gore’s Internet

VPN

Traveling Salesman
Troubles in paradise

- **Security often stands in the way of work**
  - People work better with mobile access
  - Multi-site deployment hell
  - Slow response to new needs

- **Ever increasing costs**
  - Skilled IT staff is expensive
  - Hardware, licensing
  - Disaster recovery

- **You are as secure as your IT security skills.**
Introducing SaaS..

- All you need is a browser! IT staff’s dream..
- Predictable costs
  — $ / user / month
- No scaling issues
  — No need to buy more servers to support more users
- “Access anywhere” (+ predictable performance ww)
- Secure
  — SaaS vendors invest a lot in infrastructure security
  — End user security is a different story..
Nothing is without problems

- **Data is out of your sight**
  - It’s somewhere in the “cloud”. Where?
  - What are the backup policies?
  - How do I know if my data was accessed?

- **Availability**
  - Helplessness during technical issues

- **Privacy issues**
  - Some countries (mostly European) have tough restrictions on data residency
New security challenges

- **Access data anywhere**
  - Any location
  - Any computer
  - Any operating system
  - Any browser
  - Any AV (if any)

- **Auditing logs – at the discretion of the SaaS vendor**

- **Effectively a new (and broad) attack vector.**

- **Alerts? SIEM?**
• APT against the SaaS provider
• Physical security
• Data center security
• Side-Channel Attack
• DDoS

Type II Attack – Application layer
• Web vulnerabilities (e.g. XSS)
• SQL injection
• Authentication bypass
• Configuration error vulnerabilities

Type III Attack – End user
• Credential theft
• Data harvesting
• Exfiltration
• Data alteration
• Defamation

SaaS provider responsibility
Enterprise responsibility
Targeted attack – on-premise

1. Break-in: Spear phishing and remote exploits to gain access
2. Latch-on: Malware and backdoors installed to establish a foothold
3. Expand: Reconnaissance and lateral movement to increase access and maintain a presence
4. Gather: Acquisition and aggregation of confidential data
5. Exfiltrate: Data exfiltration to external networks

- Firewall, Patching, External IDS, VPN, Sandboxing, SWG
- Endpoint security, Antivirus, Host FW
- Internal IDS/IPS, DMZ, Compartment, Access Control
- Application FW, IRM, DB FW
- DLP, Proxy, Behavioral analysis, external IDS
- Forensics, Watermarking
Targeted attack - SaaS

1. Break-in
   - Spear phishing and remote exploits to gain access

2. Expand
   - No need to latch-on, all you need is the right credential & internet access

3. Gather
   - No need to expand or maintain a presence, direct access to all resources

4. Exfiltrate
   - No need to exfiltrate to external networks, attacker uses its own computer and network

5. Summary
   - Reconnaissance and lateral movement to increase access and maintain a presence
   - Acquisition and aggregation of confidential data
Authentication in the Cloud

- Starting point: simple username & password
- What if I use 20 applications at Work?
  - Single sign on
  - User (de-)provisioning
- SSO (IdP) providers
  - Cloud: OneLogin, Okta
  - On-premise: Microsoft ADFS, IBM Tivoli, HP IceWall
- Protocol war for SSO
  - SAML 2.0 emerged victorious (unless you ask MS)
Security Assertion Markup Language 2.0

— Celebrated its 9th birthday last week!

Used to exchange claims (assertions) about a user’s identity in signed XMLs.

Instead of presenting a password:

— You presents a claim signed by a trusted IdP.

SAML or similar protocols are used:

— Between SaaS applications
— Within(!) SaaS applications
SAML flow

- **Three parties to every authentication:**
  - Service Provider – the consumer of claims.
  - Browser
  - Identity Provider (IdP) – the producer of claims.

- The browser is pimped around by the SP and IdP.
SAML flow

Security Assertion Markup Language 2.0

Resource Server

App generates auth request

Client (Web Browser)

User accesses URL in app

Authorization Server / IdP

Auth request is passed, verified

HTTP POST to AS w/ Auth Request

User is sent to login page at AS

User logs in

Redirect to app w/ SAML token

SAML token is generated

User is logged in to resource server
A potential Achilles heel

- Really difficult to implement right
  - You can take advantage of _some_ libraries

- No mainstream/standard implementation
  - Shibboleth is closest to that, but it’s far from popular
  - Everybody’s winging it

- Many different implementations
  - Compatibility issues
  - Very few “eyes” (like us) tried to find bugs

- Lots of bugs that are waiting to be discovered
Facebook remote code execution (due to SSO bug!)
— And why defaults are important

The Enemy Within (currently in responsible disclosure)
— And the border between customization and security

Ice Dagger – MS13-104
— Embarrassing Office 365 token theft bug
- Found by Reginaldo Silva in November 2013.
  - Facebook’s highest bounty: $33.5k

- Optional “forgot password” flow:
  - Use Google account to prove ownership
  - Works using OpenID

- Facebook is using libxml to process these XMLs
  - Default settings permit XML External Entity
Basically, you get to open local files and conns.

The fix? Simply add:

```c
libxml_disable_entity_loader(true);
```

The truth?

- These things are quite common.
- Default values aren’t always secure.

Our example: libxmlsec

- Requires user to explicitly disable the option to specify custom certificate during XML sig. check
The Enemy Within

- A vulnerability in one of the Top 10 SaaS apps.
- Currently in responsible disclosure.
- Takes advantage of the paradigm that SaaS apps consider their own domains to be trusted.

- But what happens when users are able to upload custom files or even customize JS?

- Easy (and silent) drive-by theft of token & cookies.
Nicknamed “Ice Dagger” – leaves no trace.

Crafted HTTP response retrieves one’s O365 token

— “The keys” to Office 365 – Microsoft’s cloud platform

Timeline:

— Found in the wild at one of our clients in April 2013
— Temporary fix for the client in place 2 days later
— Reported on May 2013
— Patched on December 2013
Our proxy processed an unusual HTTP request.

Flagged by our heuristics engine due to two strikes:

— Destination host was a known TOR gateway
— The request was performed by Microsoft Word

Scheduled for in-depth review by Adallom Labs.

Our story begins.
Microsoft’s cloud offering for organizations

- Main competitor is Google Apps for Business

Comprised of:

- Exchange Online (hosted email service)
- SharePoint Online (collaboration services)
- OneDrive Pro (file storage, formerly SkyDrive)
- Office 2013 desktop applications (Word and friends)

These are very different products fused together
We’re going to talk mainly about Word

— But it’s the same for PowerPoint, Excel, OneNote

Instead of serial numbers – you sign in to activate.

Must be signed for SharePoint and OneDrive.

There’s a psychological campaign to sign you in:

Office 2013 changes for the cloud
Once you’re signed in

- **Word exchanges your credentials for a token**
  - It is then internally stored.

- **When you try to access SharePoint or OneDrive**
  - Word trades its token for an authentication cookie
  - The cookie has a short life span, the token has a really long one
Back to our case

- We started tracing the request
  - We got to the specific device
  - Questioned the employee
  - Reconstructed his actions with him

- The trigger: a spear fishing email
  - Contained information relevant to his job.

- The link destination was a TOR gateway
  - Using a TOR hidden service
Back to our case – cont’d

- The hidden service was no longer accessible
  — Duh!

- The IP in the email was an anonymous proxy.

- No document to investigate

- No file hash to track
We’re going to use Fiddler

— Web Debugging Proxy

— Available for free from http://www.telerik.com/fiddler

We’re going to do it step-by-step

— Please – slow me down if something is unclear.
We managed to fully reproduce and develop a POC

We contacted MSRC on the 29th of May with:

- Detailed research
- Working POC

We begun our quest for a patch
- **It took over 6 months (!) for the patch to come out**
  - Bypassing MSFT’s bullshit filters took a few weeks
  - Reproduction took them a few weeks too.
    - Even though we supplied a working PoC.
- **Responsible disclosure – or irresponsible one?**
  - Users were vulnerable for a long period of time
  - No pressure on the vendor to fix the issue
  - Some companies could have protected themselves
Why was vulnerability classified as “Important”?

— According to MSFT it’s because “It does not result in remote code execution”

How are they be assessing SaaS vulnerabilities?

— Could it be using metrics from the Windows world?
What if I told you..

- That most of these vulnerabilities are fixed silently?
- That there’s no CVE/NVD for SaaS applications?
- That SaaS vendors are reluctant to have one?
  — And to fix the reported ones

- It’s our shared responsibility
  — Insist on having a CVE for every disclosure
  — Push for a unified disclosure mechanism for SaaS
  — Insist and apply pressure for early patching
THANK YOU

Questions?