Attack Surface Analysis Cheat Sheet

Access Control Cheat Sheet Authentication Cheat Sheet

What is Attack Surface Analysis and Why is it Important?

This article describes a simple and pragmatic way of doing Attack Surface Analysis and managing an application's Attack Surface. It is targeted to be used by developers to understand and manage application security risks as they design and change an application, as well as by application security specialists doing a security risk assessment. The focus here is on protecting an application from external attack - it does not take into account attacks on the users or operators of the system (e.g. malware injection, social engineering attacks), and there is less focus on insider threats, although the principles remain the same. The internal attack surface is likely to be different to the external attack surface and some users may have a lot of access.

Attack Surface Analysis is about mapping out what parts of a system need to be reviewed and tested for security vulnerabilities. The point of Attack Surface Analysis is to understand the risk areas in an application, to make developers and security specialists aware of what parts of the application are open to attack, to find ways of minimizing this, and to notice when and how the Attack Surface changes and what this means from a risk perspective.

Attack Surface Analysis is usually done by security architects and pen testers. But developers should understand and monitor the Attack Surface as they design and build and change a system.

Attack Surface Analysis helps you to:

1. identify what functions and what parts of the system you need to review/test for security vulnerabilities
2. identify high risk areas of code that require defense-in-depth protection - what parts of the system that you need to defend
3. identify when you have changed the attack surface and need to do some kind of threat assessment

Defining the Attack Surface of an Application

The Attack Surface describes all of the different points where an attacker could get into a system, and where they could get data out.

The Attack Surface of an application is:

1. the sum of all paths for data/commands into and out of the application, and
2. the code that protects these paths (including resource connection and authentication, authorization, activity logging, data validation and encoding), and
3. all valuable data used in the application, including secrets and keys, intellectual property, critical business data, personal data and PII, and
4. the code that protects these data (including encryption and checksums, access auditing, and data integrity and operational security controls).

You overlay this model with the different types of users - roles, privilege levels - that can access the system (whether authorized or not). Complexity increases with the number of different types of users. But it is important to focus especially on the two extremes: unauthenticated, anonymous users and highly privileged admin users (e.g. database administrators, system administrators).

Group each type of attack point into buckets based on risk (external-facing or internal-facing), purpose, implementation, design and technology. You can then count the number of attack points of each type, then choose some cases for each type, and focus your review/assessment on those cases.

With this approach, you don't need to understand every endpoint in order to understand the Attack Surface and the potential risk profile of a system. Instead, you can count the different general type of endpoints and the number of points of each type. With this you can budget what it will take to assess risk at scale, and you can tell when the risk profile of an application has significantly changed.

The OWASP Foundation works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

you can tell when the risk profile of an application has significantly changed.

Identifying and Mapping the Attack Surface

You can start building a baseline description of the Attack Surface in a picture and notes. Spend a few hours reviewing design and architecture documents from an attacker's perspective. Read through the source code and identify different points of entry/exit:

- User interface (UI) forms and fields
- HTTP headers and cookies
- APIs
- Files
- Databases
- Other local storage
- Email or other kinds of messages
- Run-time arguments
- …Your points of entry/exit

The total number of different attack points can easily add up into the thousands or more. To make this manageable, break the model into different types based on function, design and technology:

- Login/authentication entry points
- Admin interfaces
- Inquiries and search functions
- Data entry (CRUD) forms
- Business workflows
- Transactional interfaces/APIs
- Operational command and monitoring interfaces/APIs
- Interfaces with other applications/systems
- …Your types

You also need to identify the valuable data (e.g. confidential, sensitive, regulated) in the application, by interviewing developers and users of the system, and again by reviewing the source code.

You can also build up a picture of the Attack Surface by scanning the application. For web apps you can use a tool like the OWASP ZAP or Arachni or Skipfish or w3af or one of the many commercial dynamic testing and vulnerability scanning tools or services to crawl your app and map the parts of the application that are accessible over the web. Some web application firewalls (WAFs) may also be able to export a model of the application's entry points.

Validate and fill in your understanding of the Attack Surface by walking through some of the main use cases in the system: signing up and creating a user profile, logging in, searching for an item, placing an order, changing an order, and so on. Follow the flow of control and data through the system, see how information is validated and where it is stored, what resources are touched and what other systems are involved. There is a recursive relationship between Attack Surface Analysis and Application Threat Modeling: changes to the Attack Surface should trigger threat modeling, and threat modeling helps you to understand the Attack Surface of the application.

The Attack Surface model may be rough and incomplete to start, especially if you haven't done any security work on the application before. Fill in the holes as you dig deeper in a security analysis, or as you work more with the application and realize that your understanding of the Attack Surface has improved.

Measuring and Assessing the Attack Surface

Once you have a map of the Attack Surface, identify the high risk areas. Focus on remote entry points – interfaces with outside systems and to the Internet – and especially where the system allows anonymous, public access.

- Network-facing, especially internet-facing code
- Web forms
- Files from outside of the network
- Backwards compatible interfaces with other systems – old protocols, sometimes old code and libraries, hard to maintain and test multiple versions
- Custom APIs – protocols etc – likely to have mistakes in design and implementation
- Security code: anything to do with cryptography, authentication, authorization (access control) and session management

These are often where you are most exposed to attack. Then understand what compensating controls you have in place, operational controls like network firewalls and application firewalls, and intrusion detection or prevention systems to help protect your application.

Michael Howard at Microsoft and other researchers have developed a method for measuring the Attack Surface of an application, and to track changes to the Attack Surface over time, called the Relative Attack Surface Quotient (RSQ). Using this method you calculate an overall attack surface score for the system, and measure this score as changes are made to the system and to how it is deployed. Researchers at Carnegie Mellon built on this work to develop a formal way to calculate an Attack Surface Metric for large systems like SAP. They calculate the Attack Surface as the sum of all entry and exit points, channels (the different ways that clients or external systems connect to the system, including TCP/UDP ports, RPC end points, named pipes...) and untrusted data elements. Then they apply a damage potential/effort ratio to these Attack Surface elements to identify high-risk areas.

Note that deploying multiple versions of an application, leaving features in that are no longer used just in case they may be needed in the future, or leaving old backup copies and unused code increases the Attack Surface. Source code control and robust change management/configurations practices should be used to ensure the actual deployed Attack Surface matches the theoretical one as closely as possible.

Backups of code and data - online, and on offline media - are an important but often ignored part of a system's Attack Surface. Protecting your data and IP by writing secure software and hardening the infrastructure will all be wasted if you hand everything over to bad guys by not protecting your backups.

Managing the Attack Surface

Once you have a baseline understanding of the Attack Surface, you can use it to incrementally identify and manage risks going forward as you make changes to the application. Ask yourself:

- What has changed?
- What are you doing different? (technology, new approach, ...)
- What holes could you have opened?

The first web page that you create opens up the system's Attack Surface significantly and introduces all kinds of new risks. If you add another field to that page, or another web page like it, while technically you have made the Attack Surface bigger, you haven't increased the risk profile of the application in a meaningful way. Each of these incremental changes is more of the same, unless you follow a new design or use a new framework.

If you add another web page that follows the same design and using the same technology as existing web pages, it's easy to understand how much security testing and review it needs. If you add a new web services API or file that can be uploaded from the Internet, each of these changes have a different risk profile again - see if if the change fits in an existing bucket, see if the existing controls and protections apply. If you're adding something that doesn't fall into an existing bucket, this means that you have to go through a more thorough risk assessment to understand what kind of security holes you may open and what protections you need to put in place.

Changes to session management, authentication and password management directly affect the Attack Surface and need to be reviewed. So do changes to authorization and access control logic, especially adding or changing role definitions, adding admin users or admin functions with high privileges. Similarly for changes to the code that handles encryption and secrets. Fundamental changes to how data validation is done. And major architectural changes to layering and trust relationships, or fundamental changes in technical architecture – swapping out your web server or database platform, or changing the run-time operating system.

As you add new user types or roles or privilege levels, you do the same kind of analysis and risk assessment. Overlay the type of access across the data and functions and look for problems and inconsistencies. It's important to understand the access model for the application, whether it is positive (access is deny by default) or negative (access is allow by default). In a positive access model, any mistakes in defining what data or functions are permitted to a new user type or role are easy to see. In a negative access model, you have to be much more careful to ensure that a user does not get access to data/functions that they should not be permitted to.

This kind of threat or risk assessment can be done periodically, or as a part of design work in serial / phased / spiral / waterfall development projects, or continuously and incrementally in Agile / iterative development.

Normally, an application's Attack Surface will increase over time as you add more interfaces and user types and integrate with other systems. You also want to look for ways to reduce the size of the Attack Surface when you can by simplifying the model (reducing the number of user levels for example or not storing confidential data that you don't absolutely have to), turning off features and interfaces that aren't being used, by introducing operational controls such as a Web Application Firewall (WAF) and real-time application-specific attack detection.

Access Control Cheat Sheet

Authentication Cheat Sheet

Edit on Github

Spotlight: Mercari, Inc.

mercari

Mercari is a C2C marketplace app that makes it easy for people to safely sell and ship their things. Launched in 2013, it's now among the largest peer-to-peer selling platforms globally. From fashion to toys, shoes to electronics and beyond, Mercari's mission is to "create value in a global marketplace where anyone can buy and sell."

Corporate Supporters

WhiteHat SECURITY. DELL Technologies salesforce

RIPSTECH Symantec. GitLab

Security Journey brinqa Allstate.

Become a corporate supporter