Agoric
Inter Protocol Assessment

Security Assessment Report

Prepared for Agoric Systems Operating Company
September 21, 2022 (version 1.1)

Project Team:

Technical Testing	Brandon Perry and Loren Browman
Technical Editing	Darren Kemp and Sara Bettes
Project Management	Sara Bettes

Table of Contents

Engagement Overview

Assessment Components and Objectives

Agoric Systems Operating Company ("Agoric") recently engaged Atredis Partners ("Atredis") to perform the Inter Protocol Assessment of the Agoric platform. Objectives included validation that Agoric smart-contract platform was developed and deployed with security best practices in mind.

Testing was performed from May 30 through June 23, by Brandon Perry and Loren Browman of the Atredis Partners team, with Sara Bettes providing project management and delivery oversight. For Atredis Partners' assessment methodology, please see Appendix I of this document, and for team biographies, please see Appendix II. Specific testing components and testing tasks are included below.

COMPONENT	ENGAGEMENT TASKS
Inter Protocol Assessment	
Assessment Targets	• VaultFactory Contract • Automated Market Maker (AMM) Contract • Parity Stability Mechanism Contract • Reserve + Rewards Mechanism • BLD Boost Contract

Assessment Tasks	• Source-Assisted Penetration Testing of the Inter Protocol
	  • Analysis of all processing logic to ensure it meets documented expectations
	  • Manual and automated runtime application testing
	  • Application spidering, fuzzing and fault injection
	  • PoC generation and validation of findings
	  • Development of adversarial unit tests where appropriate

Reporting and Analysis	
Analysis and Deliverables	• Status Reporting and Realtime Communication • Comprehensive Engagement Deliverable • Engagement Outbrief and Remediation Review

The ultimate goal of the assessment was to provide a clear picture of risks, vulnerabilities, and exposures as they relate to accepted security best practices, such as those created by the National Institute of Standards and Technology (NIST), Open Web Application Security Project (OWASP), or the Center for Internet Security (CIS). Augmenting these, Atredis Partners also draws on its extensive experience in secure development and in testing high-criticality applications and advanced exploitation.

Engagement Tasks

Atredis Partners performed the following tasks, at a high level, for in-scope targets during the engagement.

Runtime Analysis

For relevant software targets identified during the course of this engagement, Atredis performed runtime analysis, using debugging and static analysis tools to analyze application flow to aid in software security analysis. Where relevant, purpose-built tools such as custom unit tests and customized clients were utilized to aid in vulnerability identification.

Source Code Analysis

Atredis Partners reviewed the in-scope application source code, with an eye for security-relevant software defects. To aid in vulnerability discovery, application components were mapped out and modeled until a thorough understanding of execution flow, code paths, and application design and architecture were obtained. To aid in this process, the assessment team engaged key stakeholders and members of the development team, where possible, to provide structured walkthroughs and interviews, helping the team rapidly gain an understanding of the application's design and development lifecycle.

Configuration and Architecture Review

Atredis performed a high-level review of available documentation and configuration data with an eye toward the overall functional design and soundness of the implementation. A key aspect of this component was to identify gaps in the architecture and design regarding aspects of design that reduce overall defensibility, aimed at pointing out fundamental issues in the application architecture that should be addressed early in the development cycle as opposed to later when the platform is closer to a full production state.

While specific vulnerabilities may be identified during the architecture and configuration review, the intent was less on finding individual defects and more on how the design of a given target affects its overall defensibility. Outcomes of the architecture review helped to inform testing objectives throughout the rest of the engagement while also helping the client define a long-term platform maturity and security design roadmap.

Executive Summary

Testing targeted the Inter protocol which is comprised of 5 individual smart contracts implementing various economic features. Atredis Partners assessed the Inter protocol as seen in the public `agoric-sdk` repository at the commit beginning with `fca3db8`. The Agoric team provided documentation for testing all contracts and was always available for support when required.

Atredis Partners performed testing from the perspective of a regular system user with access to all publicly exposed facets. Attacks requiring access to internal objects or creator facets were also explored with the understanding that the associated risks are greatly mitigated by the object-capability (OCAP) security architecture.

The focus of the testing was directed to ensuring all contract code paths safely validate user input where possible and do not overly expose sensitive objects or internal state. Atredis also prioritized test cases which seek to exploit the intended order of operations and reentrancy hazards. Lower-level components responsible for translating messages across multiple VATs in a distributed system were covered in a previous Kernel API assessment (delivered April 28, 2022) and are not currently under review.

Key Conclusions

Overall, Atredis Partners found the Inter protocol contracts to be well-thought out and well-documented. The Agoric platform relies heavily on object capabilities which has been carefully implemented throughout with no evidence to suggest any unintended exposure of sensitive object references to untrusted parties.

Across all Inter protocol contracts, Atredis did not identify any issues which would undermine contract logic to benefit an attacker. This includes minting new assets, over collecting collateral, and arbitrary reward distributions.

Atredis recommends imposing restrictions on user supplied input size. Mitigation strategies have been discussed and a solution has already been submitted for internal review. Some code quality issues around linting and error reporting have also been reported as informational.

As in any security assessment, some general areas for improvement were noted, but overall Atredis Partners would rate the tested components of Agoric's platform as sound from a security perspective and well-aligned with modern secure development practices.

Platform Overview

The Agoric platform is designed to enable decentralized finance (DeFi) applications written in secure JavaScript smart contracts. All the contracts on the Agoric platform are executed on top of Agoric's smart contract framework Zoe. Zoe is responsible for handling payments,

minting new assets, and performing trade operations. While Atredis needed to use the Zoe framework during testing, Zoe itself was out of scope for the engagement.

At the time of testing Agoric is the sole publisher of smart contracts with all Inter protocol contracts being registered and publicly accessible. All Inter contracts expose public facets to limit access to sensitive methods which represent a major security boundary. Future deployments will integrate user supplied smart contracts and care must be taken to ensure developers understand the principles of least privilege and the object capabilities paradigm to avoid making simple errors.

Each contract adds to the Inter protocol to provide an incentivized and decentralized digital economy and requires independent consideration with respect the security risks and concerns.

Vault Factory

The Vault Factory contract implements and manages treasury vaults. The Vault Factory owns various vault managers for each collateral type and can mint IST to facilitate collateral backed loans. The Vault Factory creates a vault director and exposes a public facet for user interaction. When a user creates a vault using the available `makeVaultInvitation`, validation is performed to ensure that the collateral type is supported, and that the initial minimum debt is exceeded.

Vaults operate within parameters set by governance which includes the interest rate, liquidity margin, loan fee, and the charging period. The accepted collateral types are decided by the Vault Factory owner. Collateral type string identifiers are validated to contain only ASCII characters and a few allowed symbols, but no length checks are performed as outlined in Vault Factory: Lack of Input Validation When Adding New Vault Collateral Types.

When a loan is created, the `mintAndReallocate` method will mint all IST to satisfy the loan which includes the principal amount plus the loan fee. The loan fee is placed in the rewards pool and the principal amount is made available to the requesting user. Attacks affecting the vault factory's ability to mint IST may include accounting errors or preventing access to the IST mint which would result in a denial-of-service scenario. Atredis did not identify any issues related to minting IST within the vault factory.

Vaults are closed by creating an invitation with `makeCloseInvitation`. Only active or liquidated vaults can be closed and must contain an Electronic Rights Transfer Protocol (ERTP) payment greater than the remaining debt when creating the invitation. If an insufficient payment is included, an assert is raised, a rejected promise is returned, and the vault is not closed.

```
Error{
    message: 'Offer {"brand":"[Alleged: RUN brand]","value":"[0n]"} is not sufficient to
pay off debt {"brand":"[Alleged: RUN brand]","value":"[5250n]"}',
}
```

Error raised when submitting insufficient ERTP payment when closing vault

ERTP collateral payments are handled by Zoe and held in escrow when creating a vault to take out a loan. The vault object must use its Zoe seat to query the current collateral amount. Atredis did not identify any attacks that could confuse accounting records and benefit an attacker.

Adjusting Balances

Vault owners can create an invitation with `makeAdjustBalancesInvitation` to alter collateral and IST balances on active vaults. Care is taken to ensure that vault transfers cannot take place while balances are being adjusted. An ERTP payment is required when submitting proposals to adjust balances and must support the balance adjustment. This means a request cannot be made which would result in an increase in debt beyond the allowed collateralization ratio.

Vault collateral is recovered when adjusting balances and closing vaults manually or due to a triggered liquidation. Atredis did not observe any paths which would result in an attacker recovering additional collateral in these situations.

Potential issues around burning (destroying) incorrect amounts of IST when closing, liquidating, and adjusting balances where also explored. The platform does not rely on user input to calculate debt amounts and instead pulls the debt amount directly from the current contract state using `getCurrentDebt` and subtracts any provided ERTP IST payment to safely calculate the IST to be burned.

Fees

Fees are generated according to governed parameters and include both loan fees and interest fees. Loan fees are a percentage of the requested loan charged when a loan is taken or extended. If the loan is extended, the additional fee is calculated based on the extended amount. Interest fees are a rate that is charged when the `chargingPeriod` elapses. Collecting fees require that the user holds a reference to the Vault Factory `machine` facet which exposes a closely held method `makeCollectFeesInvitation`. Fees can be collected using the invitation to recover all fees across all vaults from the reward pool.

Liquidation

Vault Factory will liquidate assets when the ratio of debt to collateral reaches a defined threshold referred to as the liquidation margin. The liquidation margin is controlled by governance and defines the margin required to maintain a loan. The Vault Manager is responsible for submitting price quotes and triggering liquidations in `processLiquidations` which operates on a map of `prioritizedVaults` that contains all vaults sorted by collateralization ratio. Vaults which have the lowest collateralization are processed first, vaults which exceed the liquidation margin are scheduled for an eventual liquidation. Atredis did not identify any attacks that would allow vaults to be process out of the intended order.

The liquidation happens in `liquidation.js` in the `liquidate` method. A `liquidator` is passed in and used to sell collateral to the AMM in attempt reduce the IST debt to a zero balance. IST proceeds from this sale are subtracted from the remaining vault debt before being burned. Collateral remaining in the vault is left for the user to claim as a payout.

Transfers

Vault owners also have the option to transfer a vault to another user by providing a transfer invitation created by `makeTransferInvitation`. A transfer invitation can be created regardless of the current debt to collateral ratio. Users who wish to buy the vault can inspect the transfer invitation to view the state of the vault and determine if they would like to proceed with the transfer. When a transfer invitation is created, the current state of the vault is saved, collateral is locked, the phase is set to `TRANSFER`, and the vault is no longer available to the original owner. The Vault Holder object helps to enforce vault access during the transition by ensuring the vault has no owner after the transfer is started. The vault can then be claimed later by submitting the transfer invitation with a Zoe offer.

Automated Market Maker (AMM)

The Automated Market Maker (AMM) is central to several of the smart contracts within the Agoric environment. As such, some functions exposed by the contract are discussed in other sections. The AMM consumes two governed parameters, the `PoolFee` and the `ProtocolFee`.

The AMM maintains a constant product ratio between two pools during trading to ensure liquidity. This means that during the operation of trading between pools, the product of the ratio of assets in each pool will always remain constant. For instance, two pools with a ratio of 4:1 will always have a constant product of 4. Two pools with a ratio of 7:3 will always have a constant product of 21 between the two pools.

To protect against arithmetic errors in the AMM when adding and removing liquidity between pools, Agoric strictly uses arbitrary precision natural numbers. The `AmountMath` library performs arithmetic on arbitrarily large integers avoiding the need for the weaker native JavaScript implementations. Agoric also enforces positive natural numbers throughout the code base with a forked library called `isNat` so that negative numbers cannot be accidentally used during computation.

```
Error {
  message: 'value "[-1499999000n]" must be a natural number',
}
```

Example Error Showing Negative Value

The `makeAddLiquidityInvitation`, `makeAddLiquidityAtRateInvitation`, and `makeRemoveLiquidityInvitation` functions enable constant product liquidity and are consumed in the reserves contract. These functions are heavily used by the reserve contract to enable governance and management of liquidity using collateral reserves. They return invitations to manage the liquidity of a given pool using the collateral of a given reserve.

The vault factory contract uses the two swap functions `makeSwapInvitation` and `makeSwapOutInvitation`. `makeSwapInvitation` and `makeSwapInInvitation` are the same method. The swap methods return invitations to perform an asset swap within the pool.

AMM also contains numerous Public Facet methods that are read-only and don't affect state. The only function exposed solely to the AMM creator is `makeCollectFeesInvitation`. This function returns an invitation to collect the fees generated while trading between pools.

Security concerns for the AMM include creating pools that may trick users into choosing a malicious asset pool. While there is always some chance of a user misclicking or misreading, Agoric enforces a small subset of ASCII characters for pool names, preventing similar-looking Unicode characters. Other issues may involve adding or removing liquidity maliciously or stealing fees that only the creator should have access to. During testing, the AMM was resilient potential attacks involving negative or unexpected numbers.

BLD Boost

The BLD Boost contract exposes three functions: 2 public and 1 available to the object creator only.

The public functions `makeLoanInvitation` and `makeReturnAttInvitation` create invitations to exercise a loan contract or attestation. Only the holder of the creator facet can create a loan attestation using `provideAttestationMaker`.

The BLD Boost contract itself handles interest due and ensures that debt does not surpass the loan limit. The `Amount` and `Nat` libraries are consistently used throughout the BLD Boost contract code to ensure correctness of arithmetic during interest or loan agreement calculations

Parity Stability Module

The Parity Stability Module (PSM) provides an interface for users to trade an accepted reference stable token (the anchor) for IST at a defined ratio normally set to parity (1:1). This mechanism facilitates stable token arbitrage by incentivizing users to trade the stable token for IST or vice versa to bring the currencies back to parity and thereby reducing the volatility of the native IST token.

PSM can mint new IST when users provide anchor currency in exchange for IST and the PSM can access the anchor pool when users provide IST in exchange for the anchor currency. IST fees are associated with trading in either direction and are controlled by governed parameters `GiveStableFee` and `WantStableFee`. A limit can also be set on the ceiling of IST the PSM may mint before asserting an error.

The PSM does not expose significant attack surface beyond what is already protected by OCAP. The `makeSwapInvitation` method is exposed by the public facet and allows a user to perform a currency swap without exposing any sensitive internal object references. Atredis did not find any scenarios by which an attacker can cause additional IST to be minted or collect fees intended for the PSM object creator.

Reserve

The reserve contract within the Agoric environment allows governance to keep and remove allotments of assets in the reserve for removing and adding liquidity to a pool. Some issues that may arise are requesting more collateral than is available in the reserve or burning assets to reduce liquidity.

`makeAddCollateralInvitation` creates an invitation to add collateral to a given reserve. At the time of writing, `makeRedeemLiquidityTokensInvitation` was not implemented and was not tested.

`addIssuer` does support Cyrillic and other Unicode characters which may allow an attacker to trick users with similarly named issuers. However, this value is only used for debugging and is not presented to the user in any way. Another mechanism is used on the client-side that allows users to give issuers "pet names".

`addLiquidityToPool` requires a minimum amount of initial liquidity which protects from users maliciously calling the function with arbitrarily small amounts of assets. This minimum amount is set when creating a new pool. During testing, Atredis noticed the error for this scenario was slightly malformed as noted in Malformed Error Message when Pool Minimum Liquidity Insufficient.

```
    message: 'The minimum initial liquidity is [object Object], rejecting [object Object]',
```

Example Error for Insufficient Liquidity

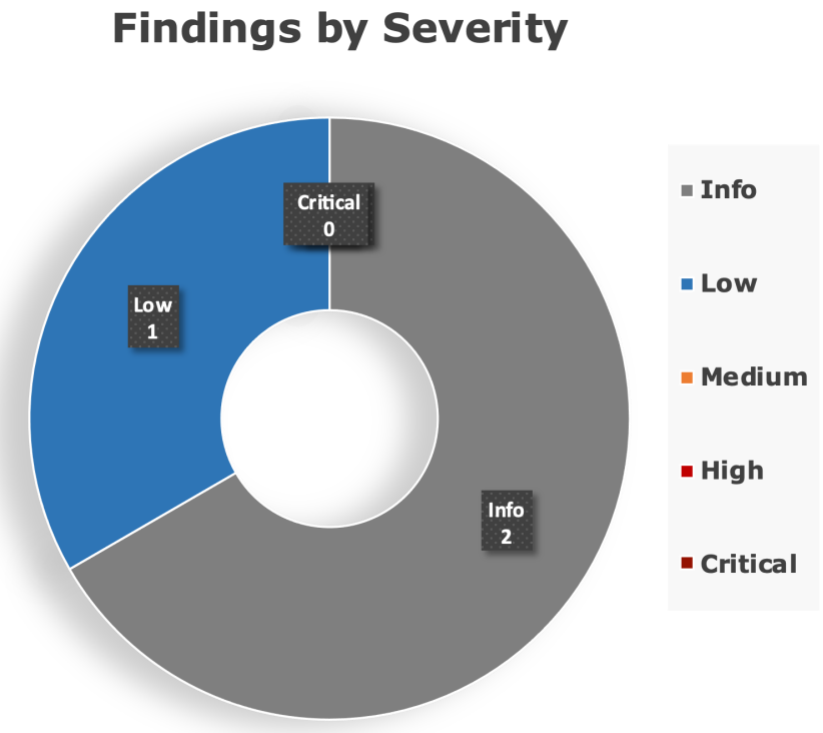
While the exception has some issues printing the values, it prevents using too little liquidity. The reserve contract relies on the `makeAddLiquidityInvitation`, `makeAddLiquidityAtRateInvitation`, and `makeRemoveLiquidityInvitation` AMM functions to govern liquidity.

Findings Summary

In performing testing for this assessment, Atredis Partners identified **one (1) low** severity finding and **two (2) informational** findings. No high or critical severity findings were noted. As stated earlier, none of these issues constitute a potential for direct compromise, and in the case of the low severity vulnerability, other protections in the platform mitigate the issue.

Atredis defines vulnerability severity ranking as follows:

- **Critical:** These vulnerabilities expose systems and applications to immediate threat of compromise by a dedicated or opportunistic attacker.
- **High:** These vulnerabilities entail greater effort for attackers to exploit and may result in successful network compromise within a relatively short time.
- **Medium:** These vulnerabilities may not lead to network compromise but could be leveraged by attackers to attack other systems or applications components or be chained together with multiple medium findings to constitute a successful compromise.
- **Low:** These vulnerabilities are largely concerned with improper disclosure of information and should be resolved. They may provide attackers with important information that could lead to additional attack vectors or lower the level of effort necessary to exploit a system.



Findings and Recommendations

The following section outlines findings identified via manual and automated testing over the course of this engagement. Where necessary, specific artifacts to validate or replicate issues are included, as well as Atredis Partners' views on finding severity and recommended remediation.

Findings Summary

The below tables summarize the number and severity of the unique issues identified throughout the engagement.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	0	1	2

Findings Detail

FINDING NAME	SEVERITY
Vault Factory: Lack of Input Validation When Adding New Vault Collateral Types	Low
Use Linter to Verify let and await Usage	Info
Malformed Error Message when Pool Minimum Liquidity Insufficient	Info

Vault Factory: Lack of Input Validation When Adding New Vault Collateral Types

Severity: Low

Finding Overview

There is no limit imposed on string size when creating new collateral types and specifying a brand keyword. As a result, the string size is constrained by the underlying JavaScript runtime. Specifying extremely large values is allowed which causes a memory exhaustion issue later in execution.

New collateral types are added by calling `addVaultType` which is considered a "closely held" method only accessible to trusted parties. Therefore, this finding is being reported with low severity.

Finding Detail

The following test unit was written to create a new collateral type with a `brand` string size of 100MB in size which resulted in memory exhaustion:

```
test('long keywords', async t => {
  const {
    zoe,
    aethKit: { mint: aethMint, brand: aethBrand },
    runKit: { brand: runBrand },
  } = t.context;

  const services = await setupServices(
    t,
    [15n],
    AmountMath.make(aethBrand, 1n),
    buildManualTimer(t.log),
    undefined,
    500n,
  );
  const { lender, vaultFactory } = services.vaultFactory;

  // long keywords
  const kw = 'A'.repeat(100 * 1000 * 1000);

  // UTF-16 plane 0 keywords not allowed
  // const kw = 'A\u2602';

  const chit = makeIssuerKit(kw);
  const params = defaultParamValues(chit.brand);

  // access to the vaultFactory allows you to create new collateral type
  await E(vaultFactory).addVaultType(chit.issuer, kw, params);
  await E(lender).getCollateralManager(chit.brand);
});
```

Reproduction unit test code

```
<--- JS stacktrace --->

FATAL ERROR: CALL_AND_RETRY_LAST Allocation failed - JavaScript heap out of memory
```

JavaScript heap memory exhaustion

Recommendation(s)

Reduce the acceptable string length to a sane value expected by the platform. Keyword validation in `cleanProposal.js` is part of the Zoe framework but could be extended to perform the additional length check.

The Agoric team has already proposed a deeper fix by implementing a limitation on maximum SwingSet message size, effectively reducing the maximum total message size to 4MB. This solution is preferred and will address the issue outlined in this finding.

References

CWE-20: Improper Input Validation: https://cwe.mitre.org/data/definitions/20.html

Use Linter to Verify let and await Usage

Severity: Info

Finding Overview

The usage of the `let` and `await` keyword can be dangerous in the context of smart contracts in the Agoric ecosystem. A grammar linter can verify safe usage during development.

Finding Detail

Due to the distributed nature of smart contracts, the JavaScript keywords `let` and `await` can introduce security issues. This is heavily documented by Agoric in both online and video documents. A linter can verify that they are used safely while development is occurring.

During the engagement, a security issue was found in the vault factory implementation by Agoric that was the result of a module-level `let` declaration for a variable. At the time of this writing, Agoric is having active discussions regarding incorporating such linter rules into active development practice.

Recommendation(s)

Enabling lint checks for non-top level `await`s and module-level `let`s can prevent from accidental security lapses during development.

References

CWE-1126: Declaration of Variable with Unnecessarily Wide Scope

https://cwe.mitre.org/data/definitions/1126.html

CWE-755: Improper Handling of Exceptional Conditions

https://cwe.mitre.org/data/definitions/755.html

GITHUB: Lint rules for contracts

https://github.com/Agoric/agoric-sdk/issues/4770

Malformed Error Message when Pool Minimum Liquidity Insufficient

Severity: Info

Finding Overview

The error message returned when a pool does not meet the minimum liquidity does not use proper formatting.

Finding Detail

When reproducing the error for not meeting the minimum liquidity for a pool, Atredis noticed the error message did not contain the expected information.

```
message: 'The minimum initial liquidity is [object Object], rejecting [object Object]',
```

Example Error from Insufficient Initial Liquidity

The `[object Object]` in the message should be templated out to retrieve the exact values for the message to be useful for the programmer or user.

Recommendation(s)

The assert does not include the details (X) template.

```
  assert(
    AmountMath.isGTE(centralAmount, minPoolLiquidity),
    `The minimum initial liquidity is ${minPoolLiquidity}, rejecting ${centralAmount}`,
  );
```

Line 156 in `/src/vpool-xyk-amm/addPool.js`

The error string should be changed to.

```
X`The minimum initial liquidity is ${minPoolLiquidity}, rejecting ${centralAmount}`,
```

Using the `X` Templating to Return the Correct Values

References

CWE-755: Improper Handling of Exceptional Conditions:

https://cwe.mitre.org/data/definitions/755.html

Appendix I: Assessment Methodology

Atredis Partners draws on our extensive experience in penetration testing, reverse engineering, hardware/software exploitation, and embedded systems design to tailor each assessment to the specific targets, attacker profile, and threat scenarios relevant to our client's business drivers and agreed upon rules of engagement.

Where applicable, we also draw on and reference specific industry best practices, regulations, and principles of sound systems and software design to help our clients improve their products while simultaneously making them more stable and secure.

Our team takes guidance from industry-wide standards and practices such as the National Institute of Standards and Technology's (NIST) Special Publications, the Open Web Application Security Project (OWASP), and the Center for Internet Security (CIS).

Throughout the engagement, we communicate findings as they are identified and validated, and schedule ongoing engagement meetings and touchpoints, keeping our process open and transparent and working closely with our clients to focus testing efforts where they provide the most value.

In most engagements, our primary focus is on creating purpose-built test suites and toolchains to evaluate the target, but we do utilize off-the-shelf tools where applicable as well, both for general patch audit and best practice validation as well as to ensure a comprehensive and consistent baseline is obtained.

Research and Profiling Phase

Our research-driven approach to testing begins with a detailed examination of the target, where we model the behavior of the application, network, and software components in their default state. We map out hosts and network services, patch levels, and application versions. We frequently use a number of private and public data sources to collect Open Source Intelligence about the target, and collaborate with client personnel to further inform our testing objectives.

For network and web application assessments, we perform network and host discovery as well as map out all available application interfaces and inputs. For hardware assessments, we study the design and implementation, down to a circuit-debugging level. In reviewing source code or compiled application code, we map out application flow and call trees and develop a solid working understand of how the application behaves, thus helping focus our validation and testing efforts on areas where vulnerabilities might have the highest impact to the application's security or integrity.

Analysis and Instrumentation Phase

Once we have developed a thorough understanding of the target, we use a number of specialized and custom-developed tools to perform vulnerability discovery as well as binary, protocol, and runtime analysis, frequently creating engagement-specific software tools which we share with our clients at the close of any engagement.

We identify and implement means to monitor and instrument the behavior of the target, utilizing debugging, decompilation and runtime analysis, as well as making use of memory and filesystem

forensics analysis to create a comprehensive attack modeling testbed. Where they exist, we also use common off-the-shelf, open-source and any extant vendor-proprietary tools to aid in testing and evaluation.

Validation and Attack Phase

Using our understanding of the target, our team creates a series of highly-specific attack and fault injection test cases and scenarios. Our selection of test cases and testing viewpoints are based on our understanding of which approaches are most relevant to the target and will gain results in the most efficient manner, and built in collaboration with our client during the engagement.

Once our test cases are validated and specific attacks are confirmed, we create proof-of-concept artifacts and pursue confirmed attacks to identify extent of potential damage, risk to the environment, and reliability of each attack scenario. We also gather all the necessary data to confirm vulnerabilities identified and work to identify and document specific root causes and all relevant instances in software, hardware, or firmware where a given issue exists.

Education and Evidentiary Phase

At the conclusion of active testing, our team gathers all raw data, relevant custom toolchains, and applicable testing artifacts, parses and normalizes these results, and presents an initial findings brief to our clients, so that remediation can begin while a more formal document is created. Additionally, our team shares confirmed high-risk findings throughout the engagement so that our clients may begin to address any critical issues as soon as they are identified.

After the outbrief and initial findings review, we develop a detailed research deliverable report that provides not only our findings and recommendations but also an open and transparent narrative about our testing process, observations and specific challenges in developing attacks against our targets, from the real world perspective of a skilled, motivated attacker.

Automation and Off-The-Shelf Tools

Where applicable or useful, our team does utilize licensed and open-source software to aid us throughout the evaluation process. These tools and their output are considered secondary to manual human analysis, but nonetheless provide a valuable secondary source of data, after careful validation and reduction of false positives.

For runtime analysis and debugging, we rely extensively on Hopper, IDA Pro and Hex-Rays, as well as platform-specific runtime debuggers, and develop fuzzing, memory analysis, and other testing tools primarily in Ruby and Python.

In source auditing, we typically work in Visual Studio, Xcode and Eclipse IDE, as well as other markup tools. For automated source code analysis we will typically use the most appropriate toolchain for the target, unless client preference dictates another tool.

Network discovery and exploitation make use of Nessus, Metasploit, and other open-source scanning tools, again deferring to client preference where applicable. Web application runtime analysis relies extensively on the Burp Suite, Fuzzer and Scanner, as well as purpose-built automation tools built in Go, Ruby and Python.

Engagement Deliverables

Atredis Partners deliverables include a detailed overview of testing steps and testing dates, as well as our understanding of the specific risk profile developed from performing the objectives of the given engagement.

In the engagement summary we focus on "big picture" recommendations and a high-level overview of shared attributes of vulnerabilities identified and organizational-level recommendations that might address these findings.

In the findings section of the document, we provide detailed information about vulnerabilities identified, provide relevant steps and proof-of-concept code to replicate these findings, and our recommended approach to remediate the issues, developing these recommendations collaboratively with our clients before finalization of the document.

Our team typically makes use of both DREAD and NIST CVE for risk scoring and naming, but as part of our charter as a client-driven and collaborative consultancy, we can vary our scoring model to a given client's preferred risk model, and in many cases will create our findings using the client's internal findings templates, if requested.

Sample deliverables can be provided upon request, but due to the highly specific and confidential nature of Atredis Partners' work, these deliverables will be heavily sanitized, and give only a very general sense of the document structure.

Appendix II: Engagement Team Biographies

Shawn Moyer, Founding Partner and CEO

Shawn Moyer scopes, plans, and coordinates security research and consulting projects for the Atredis Partners team, including reverse engineering, binary analysis, advanced penetration testing, and private vulnerability research. As CEO, Shawn works with the Atredis leadership team to build and grow the Atredis culture, making Atredis Partners a home for some of the best minds in information security, and ensuring Atredis continues to deliver research and consulting services that exceed our client's expectations.

Experience

Shawn brings over 25 years of experience in information security, with an extensive background in penetration testing, advanced security research including extensive work in mobile and Smart Grid security, as well as advanced threat modeling and embedded reverse engineering.

Shawn has served as a team lead and consultant in enterprise security for numerous large initiatives in the financial sector and the federal government, including IBM Internet Security Systems' X-Force, MasterCard, a large Federal agency, and Wells Fargo Securities, all focusing on emerging network and application attacks and defenses.

In 2010, Shawn created Accuvant Labs' Applied Research practice, delivering advanced research-driven consulting to numerous clients on mobile platforms, critical infrastructure, medical devices and countless other targets, growing the practice 1800% in its first year.

Prior to Accuvant, Shawn helped develop FishNet Security's penetration testing team as a principal security consultant, growing red team offerings and advanced penetration testing services, while being twice selected as a consulting MVP.

Key Accomplishments

Shawn has written on emerging threats and other topics for Information Security Magazine and ZDNet, and his research has been featured in the Washington Post, BusinessWeek, NPR and the New York Times. Shawn is a twelve-time speaker at the Black Hat Briefings and has been an invited speaker at other notable security conferences around the world.

Shawn is likely best known for delivering the first public research on social network security, pointing out much of the threat landscape still exists on social network platforms today. Shawn also co-authored an analysis of the state of the art in web browser exploit mitigation, creating the first in-depth comparison of browser security models along with Dr. Charlie Miller, Chris Valasek, Ryan Smith, Joshua Drake, and Paul Mehta.

Shawn studied Computer and Network Information Systems at Missouri University and the University of Louisiana at Lafayette, holds numerous information security certifications, and has been a frequent presenter at national and international security industry conferences.

Brandon Perry, Principal Research Consultant

Brandon Perry's responsibilities include complex web application and web services assessments, software reverse engineering, and source code reviews as well as red team and attack simulation engagements.

Experience

Most recently, Brandon was a Senior Penetration Tester at NTT Security, and contributed to the Foxglove Security blog at http://www.foxglovesecurity.com along with teammates (and now fellow Atredians) Justin Kennedy and Stephen Breen.

Prior to NTT Security, Brandon worked as a developer at Rapid7, supporting the Metasploit Framework and Nexpose vulnerability scanner. Brandon has also worked as a security engineer for Bethesda, working on AAA game titles such as DOOM, Fallout 4, and Elder Scrolls Online.

Brandon's extensive experience as a software engineer across several complex application stacks gives him key, practitioner-level insight into application security from the big picture down to the specifics of implementation and remediation. This development background makes Brandon uniquely suited for roles where interfacing and collaborating with development teams is crucial to a successful engagement.

Key Accomplishments

Brandon is the author of the No Starch book "Gray Hat C#", and the co-author of "Wicked Cool Shell Scripts, 2nd edition", and has spoken at DerbyCon and Infosec Southwest on web application vulnerabilities and exploit writing. Brandon has also heavily contributed to the Metasploit Framework with code and exploits, as well as several other open source security tools.

Loren Browman, Senior Research Consultant

Loren Browman has over 10 years of experience in both consulting and federal law enforcement environments. His experiences range from deep security research in federal government to product and application testing for Fortune 500 corporations. Loren is a recognized subject matter expert (SME) in securing IoT products and advanced hardware testing methodology. Areas of expertise include reverse engineering of hardware, firmware, and communication protocols.

Experience

Loren has conducted numerous large scale product security assessments including challenging black box security assessments and secure design reviews.

Prior to joining Atredis, Loren was an operations supervisor and security researcher for the Royal Canadian Mounted Police (RCMP). This role included providing technical expertise to support police investigations and leading security research efforts in order to circumvent security mechanisms and develop deployable capabilities.

Key Accomplishments

Loren has developed numerous tools for accelerating research on a wide range of products. This includes the development of a fuzzing suite for automotive Electronic Control Units over CAN bus vehicle networks, this led to the discovery of multiple hidden services and exploits. More recently, Loren published nrfsec, a tool for automating firmware recovery vulnerability on secured nrf51 System on Chips.

Loren has studied Electrical and Computer Engineering at the British Columbia Institute of Technology and has attended various specialized training sessions including the Arm IoT Exploit Laboratory, Power Analysis and Glitching and is an Offensive Security Certified Professional (OSCP).

Darren Kemp, Research Consulting Director

Darren Kemp leads and executes highly technical software security, network, and web application assessments and advanced red team assessments, as well as complex reverse engineering and exploit development projects.

Experience

Darren brings over a decade of professional experience in the information security space, spanning a variety of roles and responsibilities. Darren's experience includes penetration testing, application security assessments, malware and vulnerability analysis and reverse engineering.

Most recently, Darren was a security researcher for Duo Security's Duo Labs team, publishing externally facing security research and supporting internal security efforts. Prior to Duo, Darren was a Senior Security Consultant for Leviathan Security Group. Prior to Leviathan, Darren worked as a Threat Analyst for Symantec's DeepSight Research Team.

Key Accomplishments

Darren has several widely-reported and publicly credited vulnerability disclosures in a number of major software products, and his research has been featured on MSN, CNET, Wired, and other media outlets. Darren developed advanced crashdump analysis tools as part of DARPA's CINDER program, and has also made many contributions to the Metasploit Project.

Darren studied Computer Technology at the Southern Alberta Institute of Technology.

Sara Bettes, Client Operations Associate

Sara Bettes assists the creation and completion of projects at Atredis Partners, ranging from the full pre-sales process to project design and management, to final delivery and follow-up. Her goals are to ensure all projects are executed in a way that reaches the goals of the client and assists the consultants at every turn.

Experience

Prior to joining Atredis Partners, Sara led a team that planned international sporting competitions, Olympic and national team qualifying events, as well as supported the mission of multiple non-profits. Her experience includes Live Sports Commentating, Staffing Management, Safety Plan Creation, Event Development, Public Relations, and Marketing efforts.

Key Accomplishments

Sara earned a bachelor's degree in Mass Communications with an emphasis in Broadcast and Public Relations from Oklahoma City University.

Appendix III: About Atredis Partners

Atredis Partners was created in 2013 by a team of security industry veterans who wanted to prioritize offering quality and client needs over the pressure to grow rapidly at the expense of delivery and execution. We wanted to build something better, for the long haul.

In six years, Atredis Partners has doubled in size annually, and has been named three times to the Saint Louis Business Journal's "Fifty Fastest Growing Companies" and "Ten Fastest Growing Tech Companies". Consecutively for the past three years, Atredis Partners has been listed on the Inc. 5,000 list of fastest growing private companies in the United States.

The Atredis team is made up of some of the greatest minds in Information Security research and penetration testing, and we've built our business on a reputation for delivering deeper, more advanced assessments than any other firm in our industry.

Atredis Partners team members have presented research over forty times at the BlackHat Briefings conference in Europe, Japan, and the United States, as well as many other notable security conferences, including RSA, ShmooCon, DerbyCon, BSides, and PacSec/CanSec. Most of our team hold one or more advanced degrees in Computer Science or engineering, as well as many other industry certifications and designations. Atredis team members have authored several books, including *The Android Hacker's Handbook*, *The iOS Hacker's Handbook*, *Wicked Cool Shell Scripts*, *Gray Hat C#*, and *Black Hat Go*.

While our client base is by definition confidential and we often operate under strict nondisclosure agreements, Atredis Partners has delivered notable public security research on improving the security at Google, Microsoft, The Linux Foundation, Motorola, Samsung and HTC products, and were the first security research firm to be named in Qualcomm's Product Security Hall of Fame. We've received four research grants from the Defense Advanced Research Project Agency (DARPA), participated in research for the CNCF (Cloud Native Computing Foundation) to advance the security of Kubernetes, worked with OSTIF (The Open Source Technology Improvement Fund) and The Linux Foundation on the Core Infrastructure Initiative to improve the security and safety of the Linux Kernel, and have identified entirely new classes of vulnerabilities in hardware, software, and the infrastructure of the World Wide Web.

In 2015, we expanded our services portfolio to include a wide range of advanced risk and security program management consulting, expanding our services reach to extend from the technical trenches into the boardroom. The Atredis Risk and Advisory team has extensive experience building mature security programs, performing risk and readiness assessments, and serving as trusted partners to our clients to ensure the right people are making informed decisions about risk and risk management.